MUZAFER SARAČEVIĆ

OBJEKTNO - ORIJENTISANO PROGRAMIRANJE I MODELOVANJE

Java i UML : zbirka rešenih zadataka

Novi Pazar, 2011.

ISBN 978-86-84389-22-2

MUZAFER SARAČEVIĆ

OBJEKTNO - ORIJENTISANO PROGRAMIRANJE I MODELOVANJE JAVA i UML

-Zbirka rešenih zadataka-

NOVI PAZAR,
2011.

Autor

Muzafer Saračević

Naziv udžbenika

Objektno - orijentisano programiranje i modelovanje, JAVA i UML

-Zbirka rešenih zadataka-

Izdavač

Univerzitet u Novom Pazaru

Za izdavača

Prof.dr Mevlud Dudić, rektor Univerziteta u Novom Pazaru

Recezenti

Prof. dr Zoran Lovreković

Prof. dr Ćamil Sukić

ISBN: 978-86-84389-22-2

Odlukom Senata Univerziteta u Novom Pazaru, odobreno je korišćenje zbirke za praktičan deo predmeta – Objektno orijentisano programiranje

С IP - Каталогизација у публикацији
Народна библиотека Србије, Београд

004.42.045(075.8)(076)(0.034.2)
004.438JAVA(075.8)(076)(0.034.2)
004.438UML(075.8)(076)(0.034.2)

САРАЧЕВИЋ, Музафер, 1984-
Objektno-orijentisano programiranje i modelovanje [Elektronski izvor] : Java i UML : zbirka rešenih zadataka / Muzafer Saračević. - Novi Pazar : Univerzitet, 2011 (Novi Pazar : Univerzitet). - 1 elektronski optički disk (CD-ROM) ; 12 cm
Sistemski zahtevi: Nisu navedeni. - Nasl. sa etikete na disku. - Tiraž 100. - Sadrži bibliografiju.

ISBN 978-86-84389-22-2
a) Објектно оријентисано програмирање -Задаци
b) Програмски језик "Java" - Задаци
c) Програмски језик "UML" - Задаци

COBISS.SR-ID 181797388

SADRŽAJ

PREDGOVOR

Zbirka "*Objektno orijentisano programiranje i modelovanje – Java i UML*" je namenjena studentima treće godine Departmana za prirodno-tehničke nauke, Univerziteta u Novom Pazaru, za praktičan deo predmeta *Objektno-orijentisano programiranje* i može se koristiti i za jedan deo vežbi za predmete *Dizajn aplikativnog softvera* i *Softversko inženjerstvo.*

Početak nastajanja ove zbirke se vezuje za moj početak angažovanja na Univerzitetu u Novom Pazaru za navedeni predmet, tako da pisanje ove zbirke traje oko četiri godine (od 2007 do 2011 godine). Navedeni zadaci su uglavnom rađeni i testirani na vežbama u protekle 4 godine.

Iskreno se zahvaljujem svojim kolegama Enisu Ujkanoviću, Seadu Mašoviću, Ulfeti Harčinović Marovac i Esadu Međedoviću, čija je saradnja doprinela objavljivanju ove zbirke.
Posebno se zahvaljujem svojoj porodici na razumevanju i podršci.

Recenzetima, **prof.dr Ćamilu Sukiću i prof.dr Zoranu Lovrekoviću**, koji su mi pomogli svojim sugestijama u poboljšanju kvaliteta ove zbirke, se najsrdačnije zahvaljujem za trud koji su uložili.

U Novom Pazaru, 2011

Autor

UVOD

Ova zbirka se sastoji od 4 dela i dva dodatka.

Prvi deo predstavlja uvod u java programski jezik. Cilj ovog poglavlja zbirke je da uvede studenta i upozna ga sa Java programskim jezikom. Prvi, uvodni zadaci pomažu da upoznate osnovnu sintaksu ovog programskog jezika. Zatim sledeći zadaci se vezuju za osnovne koncepte u javi, odnosno na objekte i klase. Naveo sam više primera kako bi studenti bolje upoznali ovaj objektno-orijentisani jezik. Zatim slede zadaci koji navode još jedan osnovni koncept jave a to je nasleđivanje. Poslednji segment ovog poglavlja čine složeni zadaci koji implementiraju sve do sad pomenuto. Na kraju ovog poglavlja postoje kontrolna pitanja i testovi za vežbanje i proveru znanja.

Drugi deo čine zadaci iz oblasti Java apleta. Na početku su navedeni osnovni zadaci koji se vezuju za rad sa komponentama za izgradnju grafičkog korisničkog interfejsa (GUI) zvanu *Abstract Window Toolkit* (AWT). Zatim sledi par zadataka koji ilustruje rad sa grafičkim elementima (metoda paint). Ako student savlada prethodna dva tipa zadataka onda slede zadaci za rad sa događajima (metod ActionPerformed). Na kraju ovog poglavlja postoji uporedni pregled- java apleti i aplikacije, kako bi se uvidela razlika između ova dva dela. Takođe, na kraju ovog poglavlja postoje kontrolna pitanja i testovi za vežbanje i proveru znanja.

Treći deo je Java-grafički korisnički interfejs. Ovaj deo se nadovezuje na drugi deo. Uključuju se AWT i Swing komponente i rad sa događajima. Takođe postoje zadaci za rad sa događajima KeyListener i MouseListener (funkcionalnost na tastere ili na klik misa). Takođe navedeni su i neki primeri kako programirati neke komponente koje su nam potrebne za izradu konkretnog informacionog sistema. Na kraju ovog poglavlja postoje kontrolna pitanja za proveru znanja.

Dodatak A predstavlja sintezu prva tri dela i prikazuje studentu kako može implementirati svoja znanja u izradi informacionog sistema. Naveo sam informacioni sistem *Kursevi Engleskog jezika*. Inače ovo je moj projekat koji sam radio na osnovnim studijama. IS sadrži opcije unošenja podataka, modifikacije, pretrage i brisanja.

Četvrti deo je UML modelovanje. Objektno-orijentisana analiza i dizajn omogućava svim učesnicima u razvoju aplikacije da na jednostavan i sveobuhvatan način steknu uvid u analizu i implementaciju konkretnog problema. UML (Unified Modeling Language) je univerzalni jezik za modelovanje koji služi za specifikaciju, vizuelizaciju, konstrukciju i dokumentaciju razvoja sistema. Koristi se u različitim fazama razvoja, od specifikacije zahteva do testiranja završenih, gotovih sistema. Za izradu UML dijagrama korišćeni su alati *STAR UML i Visual Paradigm for UML.*

Dodatak B je primena UML alata u dizajnu i analizi informacionog sistema. U ovom dodatku je predstavljen modul Web aplikacije baziran na poslovanju On-line Knjižare, sa svim objašnjenim procesima počevši od pretrage, nabavke, brisanja, izmena kao i mogućnost On-line kupovine. Na kraju ovog dodatka sam naveo uporedni pregled UML dijagrama i ekranskih formi koje su programirane u Java programskom jeziku, a sve u cilju da bi uočili povezanost između ova dva alata.

1 DEO

JAVA OSNOVNI KONCEPTI

1 JAVA OSNOVNI KONCEPTI

Java je objektno-orijentisani programski jezik, koji je razvila kompanija Sun Microsystems početkom devedesetih godina. Mnogi koncepti Jave su bazirani na jeziku Oberon. Izbacili su koncept modula i uveli pakete kakve danas znamo, koji se oslanjaju na fajl sistem i uveli formalno koncept klasa iz objektno-orijentisane paradigme. Osim toga jezik ima sintaksu iz C i C++-a, ali je mnogo stroži pri prevođenju, dizajniran tako da bude nezavisan od platforme, i sa pojednostavljenim upravljanjem memorijom.

1.1 Uvodni zadaci

Najpre ćemo početi sa nekim uvodnim zadacima kako bi se upoznali sa ovim programskim jezikom.

1.Napisati program koji na standardni izlaz ispisuje „ovo je prvi zadatak iz jave“.

REŠENJE:

```
public class primer1 {
        //izvršna metoda
   public static void main(String[] args) {
      //ispis na standardni izlaz
System.out.println("ovo je prvi zadatak iz jave !");
   }
}
```

Možemo uočiti da najpre imamo klasu ***primer1 (*public class primer1)** , koja poseduje glavnu izvršnu metodu, koja ispisuje na standardni izlaz navedenu poruku. System.out.print je naredba za ispis u jednom redu, a *println* ispisuje u novom redu. U primerima možemo koristiti dva načina:

Prvi način: System.out.println("string"+i);

Drugi način: System.out.printf("string", i);

NAPOMENA
Komentari u Javi:

- Jednoredni komentari obeležavaju se sa // i završavaju se na kraju reda
- Dvosmerni komentari obeležavaju se sa /* i završavaju se sa */. Ovi komentari mogu se prostirati preko više redova. Ne mogu biti ugniježđeni.
- Komentari za dokumentaciju obeležavaju se sa /** i završavaju se sa */. Ovi komentari mogu se prostirati preko više redova. Ne mogu biti ugniježđeni.

Sada ćemo malo proširiti prošli zadatak pa pored standardnog sistemskom ispisa uvešćemo neke promenljive i odraditi neku operaciju nad njima.

2.Napisati program koji sabira dva broja i ispisuje na standardni izlaz

REŠENJE:

```
public class zbir {
   public static void main(String[] args) {
      int a=5;
      int b=6;
      int sum=a+b;
      System.out.println("Suma broja 5 i 6 je = " + sum);
   }
}
```

Znači ovaj zadatak ispisuje sumu dva broja (tipa int).

Za razdvajanje, java koristi nekoliko znakova:

- () (obične zagrade) - Služe za odvajanje liste parametara od poziva metode. Koriste se i za naglašavanje prioriteta izraza, za grupisanje izraza u upravljačkim naredbama kao i za određivanje tipova podataka pri konverziji.
- {} (vitičaste zagrade) - Služe za ograđivanje vrednosti automatski inicijalizovanih nizova, za definisanje blokova naredbi, klasa, metoda kao i za definisanje lokalnog opsega važenja promenljivih.

- [] (uglaste zagrade) - Koriste se za izdvajanje vrednosti članova nizova i za deklarisanje samih nizova.
- ; (tačka i zarez) - Zaključuje naredbu.
- , (zarez) - Razdvaja identifikatore u deklaraciji promenljive i koristi se za povezivanje naredbi unutar petlji.
- . (tačka) - Služi za razdvajanje naziva paketa od potpaketa i klasa kao i za razdvajanje promenljivih ili metoda od imena objekata.

3.Napisati program koji množi dva broja i ispisuje na standardni izlaz proizvod.

REŠENJE:

```
public class proizvod{
   public static void main(String[] args) {
      int a=4;
      int b=6;
      System.out.println("Proizvod  brojeva 4 i 6 je = " + a*b);
   }
}
```

NAPOMENA

Operatori i operacije:
• Svaki operator izvršava neku funkciju
• Unarni, binarni i ternarni operatori
• Rezultat izvršavanja operatora zavisi od tipa operanada

Sada ćemo prikazati malo složeniji zadatak u odnosu na prethodne,i uvešćemo neke petlje (for, if naredbu i sl.)

4. Napisati program koji računa faktorijal broja 5.

REŠENJE:

```
public class faktor{
   public static void main(String[] args) {
      int product = 1;
      for (int tekuci = 1; tekuci <= 5; tekuci++) {
         product *= tekuci;
      }
```

```
        System.out.println("Faktorijal broja 5 je  " + product);
    }
}
```

Videli smo da primenom for petlje imamo ponavljanje neke operacije onoliko puta koliko smo definisali u petlji.

NAPOMENA
Sve do J2SE5, java je sadržavala 48 rezervisanih reči. Uz sintaksu separatora i operatora one predstavljaju definiciju programskog jezika Java.

Rezervisane reči Jave su:

Abstract,boolean, break, byte, case, catch, char, class, const, continue, default, do, double, else, extends, final, finally, float, for, goto, if, implements, import,instanceof, int, interface, long, native, new, package, private, protected, public, return, short, static, strictfp, super, switch, synchronized, this, throw, throws, transient, try, void, volatile i while.

Osim ovih, rezervisane su i vrednosti: true, false i null.

5.Napisati program koji sabira prvih 10 brojeva (od 1 do 10).

REŠENJE:

```
public class suma10 {
    public static void main(String[] args) {
        int sum = 0;
        for (int tekuci = 1; tekuci <= 10; tekuci++)
        {
            sum += tekuci;
        }
        System.out.println("Sum = " + sum);
    }
}
```

Sada ćemo uvesti mogućnost sistemskog unosa, tako da imamo mogućnost da proizvoljno unosimo vrednosti za odgovarajuće promenljive.

6. Sabrati četiri broja, omogućiti unos podataka.

REŠENJE:

```
import java.util.Scanner;
public class zbir2 {
public static void main(String[] args) {
int br1;
int br2;
int br3;
int br4;
int suma;
Scanner ulaz=new Scanner(System.in);

System.out.println("uneti prvi broj");
br1=ulaz.nextInt();
System.out.println("uneti drugi broj");
br2=ulaz.nextInt();
 System.out.println("uneti treci broj");
br3=ulaz.nextInt();
 System.out.println("uneti cetvrti broj");
 br4=ulaz.nextInt();

 suma=br1+br2+br3+br4;
System.out.println("rezultat je",suma);
}
```

Čitanje podataka sa konzole u programskom jeziku Java vrši se iz objekta System.in kombinaciji sa odgovarajućim klasama reader-a (najčešće se koriste baferovani ulazni tok – BufferedReader sa svojom potklasom – InputStreamReader)

```
BufferedReader br=new BufferedReader(new InputStreamReader(System.in));
```

Prilikom korištenja Reader-a potrebno je importovati paket IO, tj.

import java.io.*;

7. Napisati program koji uporedjuje dva uneta broja.

REŠENJE:

```
import java.util.Scanner;

public class MaxDvaBroja {
public static void main(String[] args) {
 Scanner unos= new Scanner (System.in);

int x1, x2;
System.out.println("uneti prvi broj");
x1=unos.nextInt();
 System.out.println("uneti drugi broj");
 x2=unos.nextInt();
 if(x1>x2)
 {System.out.printf("veci je", x1); }
 else {System.out.printf("veci je", x2); }

        }

}
```

Jednostavnije čitanje podataka sa konzole može se obaviti korištenjem klaseScanner. Konstruktor klase Scanner koji se koristi u ovu svrhu ima opšti oblik:

```
Scanner(InputStream is)
```

kojim se pravi objekat tipa Scanner koji tok *is* koristi kao izvor podataka. Sledeći iskaz pravi objekat tipa Scanner koji podatke čita sa standardnog ulaza, što je podrazumijevano tastatura:

```
Scanner konzUlaz=new Scanner(System.in);
```

Napomena:

System.in je objekat tipa InputStream.

Prilikom upotrebe klase Scanner po pravilu se treba pridržavati sledeće procedure:

- *Pozivom jedne od hasNextX metoda klase Scanner, gdje X predstavlja tip željenih podataka, utvrdi se da li je određeni tip ulaza dostupan.*

- *Ukoliko je ulaz dostupan, pročita se tako što se pozove pozivom jedne od nextXmetoda klase Scanner.*
- *Postupak se ponavlja sve dok se ne učitaju svi podaci sa ulaza.*

8. Napisati program koji oredjuje koji je broj najveci(3 broja).

REŠENJE:

```
import java.util.Scanner;

public class Max3Broja {
public static void main(String[] args){

Scanner unos=new Scanner(System.in);
 int x, y, z;
 System.out.println("uneti prvi broj");
x=unos.nextInt();
System.out.println("uneti drugi broj");
 y=unos.nextInt();
System.out.println("uneti treci broj");
 z=unos.nextInt();
 if(x>y) {if(x>z) {System.out.println("najveci je %d", x); }
else {System.out.printf("najvece je %d", z); } }
 else {if(y>z) {System.out.printf("najveci je", y);}
else if(z>x) {System.out.printf("najveci je", z);
                        } }
            }
}
```

9. Napisati program koji konvertovanje iz float u int.

REŠENJE:

```
import java.util.Scanner;

public class konvertovanje {
public static void main(String[] args) {
 int x, y, z; float a, b, c;

Scanner unos=new Scanner(System.in);
 System.out.println("ZADATAK1");
```

```
 System.out.println("uneti prvi celi broj");
 x=unos.nextInt();

 System.out.println("uneti drugi celi broj");
 y=unos.nextInt();
 System.out.println("uneti prvi realan broj");
 a=unos.nextFloat();
System.out.println("uneti drugi realan broj");
b=unos.nextFloat();
 c=(float)(x+y);
 z=(int)(a+b);

System.out.printf("rezultat je", x, y, z);
System.out.printf("rezultat je", a, b, c);

          }

}
```

10.Napisati program koji izračunava proizvod n brojeva.

REŠENJE:

```
import java.util.Scanner;

public class zadatak {
public static void main(String[] args) {
 Scanner unos=new Scanner(System.in);
 int n, i; i=1;
System.out.println("uneti n");
 n=unos.nextInt();
 for(int k=1;k<=n;k++) { i=i*k; }
System.out.println("proizvod je"+ i); }
}
```

11.Napisati program koji izlistava brojeve izmedju m i n.

REŠENJE:

```
import java.util.Scanner;
public class brojevi {
public static void main(String[] args) {
```

```
Scanner unos=new Scanner(System.in);
 int n, m;
 System.out.println("uneti prvi celi broj");
 n=unos.nextInt();
System.out.println("uneti drugi celi broj");
 m=unos.nextInt();
 if(n>m) { for(int i=m; i++<n;){
 System.out.printf("%d", i);} }
 else {
for(int i=n; i++<m;)
{
System.out.printf("%d", i);}
                        }
            }
}
```

12.Napisati program koji ispisuje informatika dok ne unesemo 1.

REŠENJE:

```
import java.util.Scanner;

public class ispis {
public static void main(String[] args) {
 int n; Scanner unos = new Scanner(System.in);
 System.out.println ("Uneti broj");
 n=unos.nextInt();
while(n!=1) {
System.out.println("informatika");
 System.out.println ("Uneti broj");
 n=unos.nextInt(); }

            }
 }
```

13.Napisati program koji pokazuje koliko imaš bodova u zavisnosti od ocene.

REŠENJE:

```
import java.util.Scanner;
public class bodovi {
public static void main(String[] args) {
```

```
Scanner unos=new Scanner(System.in);
 int n; System.out.println("unesi ocenu");
 n=unos.nextInt();
switch(n){
 case 1: System.out.println("imao si od 0-29"); break;
case 2: System.out.println("imao si od30-49"); break;
case 3: System.out.println("imao si od 49-69"); break;
case 4: System.out.println("imao si od 69-80"); break;
case 5: System.out.println("imao si od 80-100"); break;
default: System.out.println("nema vise ocena"); break;
                        }
                }
}
```

14.Korisnik unosi n brojeva program govori koliko ima parnih i neparnih i koliko pozitivnih i negativnih.

REŠENJE:

```
import java.util.Scanner;
public class par_nepar {
public static void main(String[] args) {
Scanner unos=new Scanner(System.in);

                        int x,y,n,i,s,p;
                        int[]c;
                        System.out.print("unesi broj brojeva");
                        n=unos.nextInt();
                        c=new int[n];
                        for(i=0; i<n; i++)
                        {System.out.print("unesi broj");
                                c[i]=unos.nextInt();
                        }
                        x=0;
                        y=0;
                        s=0;
                        p=0;
                        for(i=0;i<n;i++)
                        {
                                if(c[i]<0){x++;}
                                if(c[i]>0) {y++;}
```

```
                    if(c[i]%2==0){s++;}
                    if(c[i]%2==1){p++;}
        }
            System.out.println("ima %d pozitivnih i %d negativnih,.",x,y);
            System.out.println("ima %d parnih i %d neparnih",s,p);
}
}
```

15.Drugi način za program za izračunavanje zbira dva broja sa sistemskim unosom (preko izuzetaka i InputStreamReader(System.in));)

REŠENJE:

```
import java.io.*;
public class Add{
int x,y;
public static void main(String args[]){
try {
   Add sum1= new Add();
   int sum;
   BufferedReader object= new BufferedReader
(new InputStreamReader(System.in));
   System.out.print("Unesi prvi broj x:="+ "");
   sum1.x=Integer.parseInt(object.readLine());
   System.out.print("Unesi drugi broj y:="+ "");
   sum1.y=Integer.parseInt(object.readLine());
   sum= sum1.x+sum1.y;
System.out.println("Zbir je:"+ sum);
}
   catch(Exception e){}
   }
  }
```

Izuzetak predstavlja neuobičajeno stanje koje se može javiti u toku izvršavanja programa, tj. greška pri izvršavanju. Izuzetak je u Javi objekat koji opisuje određeno vanredno stanje koje nastaje u dijelu koda.

Obrada izuzetaka u Javi sprovodi se pomoću 5 rezervisanih reči: try, catch, throw, throws i finally. Prilikom definisanja sopstvenih izuzetaka nasleđuje se klasa Exception (paketjava.lang).

Pisanje sopstvenih izuzetaka i obrada istih predstavlja zgodan način izveštavanja o greškama nastalim u toku izvršavanja programa.

NAPOMENA
Niz je imenovani set promenljivih istog tipa

- Svaka promenljiva u nizu se naziva element niza
- Deklarisanje niza Double [ ] xKoordinata;
- Definisanje niza xKoordinata = new double [10];

Inicijalizacija niza

- double [ ] xKoordinata [1.2, 3.4, 1.8, 2.6, 7.1, 2.4, 0.0, 4.2, 9.3, 3.5];

Ako se želi inicijalizacija samo nekih elemenata

- xKoordinata [0] = 1.2; xKoordinata [2] = 1.8;

16. Program za ispisivanje jednodimenzionalnog niza (10).

REŠENJE:

```
public class niz1{
  public static void main(String[] args) {
    int[] a2 = new int[10];
    for (int i=0; i<a2.length; i++)
     {
       a2[i] = i;
       System.out.print(" " + a2[i]);
     }
     System.out.println("");
   }
  }
```

Ako ste savladali prethodni zadatak onda ćemo sada razmotriti dvodimenzionalan niz.

17. Program za ispisivanje dvodimenzionalnog niza (10,5).

REŠENJE:

```
public class niz2{
  public static void main(String[] args) {
```

```
    int[][] a2 = new int[10][5];
    for (int i=0; i<a2.length; i++) {
      for (int j=0; j<a2[i].length; j++) {
        a2[i][j] = i;
        System.out.print(" " + a2[i][j]);
      }
      System.out.println("");
    }
  }
}
```

18. Program za uporedjivanje dva niza tipa Char.

REŠENJE:

```
import java.util.Arrays;
public class niz2cahr {
  public static void main(String[] args) {
    char[] c1 = new char[] { 'a', 'b', 'c', 'd' };
    char[] c2 = new char[] { 'a', 'b', 'c', 'e' };
    System.out.println(Arrays.equals(c1, c2));
    char[] a1 = new char[] { 'a', 'b', 'c', 'd' };
    char[] a2 = new char[] { 'a', 'b', 'c', 'd' };
    System.out.println(Arrays.equals(a1, a2));
  }
}
```

19. Program za sortiranje niza.

REŠENJE:

```
import java.util.*;
public class  array{
  public static void main(String[] args){
    int num[] = {50,20,45,82,25,63};
    int l = num.length;
    int i,j,t;
    System.out.print("Navedeni clanovi niza: ");
    for (i = 0;i < l;i++ ){
     System.out.print("  " + num[i]);
    }
    System.out.println("\n");
```

```
   System.out.print("Sortiran niz: ");
   Arrays.sort(num);
   for(i = 0;i < l;i++){
     System.out.print("  " + num[i]);
   }
 }
}
```

20. Program za izračunavanje determinante matrice.

REŠENJE:

```
public class Determinant {
    public int determinant(int[][] arr) {
        int result = 0;
        if (arr.length == 1) {
            result = arr[0][0];
            return result;
        }
        if (arr.length == 2) {
            result = arr[0][0] * arr[1][1] - arr[0][1] * arr[1][0];
            return result;
        }
        for (int i = 0; i < arr[0].length; i++) {
            int temp[][] = new int[arr.length - 1][arr[0].length - 1];

            for (int j = 1; j < arr.length; j++) {
                for (int k = 0; k < arr[0].length; k++) {
                    if (k < i) {
                        temp[j - 1][k] = arr[j][k];
                    } else if (k > i) {
                        temp[j - 1][k - 1] = arr[j][k];
                    }
                }
            }
            result += arr[0][i] * Math.pow(-1, (int) i) * determinant(temp);
        }
        return result;
    }
```

```
public static void main(String[] args) {
            int array[][] = { { 5, 6 }, { 8, 9 } };
            Determinant d = new Determinant();
            int result = d.determinant(array);
            System.out.println(result);
      }
}
```

21. Program za sabiranje dve matrice.

REŠENJE:

```
class MatrixZbir{
   public static void main(String[] args)  {
     int array[][]= {{4,5,6},{6,8,9}};
   int array1[][]= {{5,4,6},{5,6,7}};
   System.out.println("Broj redova= " + array.length);
   System.out.println("Broj kolona= " + array[1].length);
   int l= array.length;
   System.out.println("Matrica 1 : ");
     for(int i = 0; i < l; i++) {
     for(int j = 0; j <= l; j++) {
       System.out.print(" "+ array[i][j]);
     }
       }
   int m= array1.length;
   System.out.println("Matrica 2 : ");
     for(int i = 0; i < m; i++) {
     for(int j = 0; j <= m; j++) {
       System.out.print(" "+array1[i][j]);
     }
       }
   System.out.println("Sabiranje : ");
     for(int i = 0; i < m; i++) {
     for(int j = 0; j <= m; j++) {
       System.out.print(" "+(array[i][j]+array1[i][j]));
     }
   System.out.println();
       }
   }
}
```

22. Program za množenje dve matrice.

REŠENJE:

```
import java.util.*;

class MatrixMnozenje{
   public static void main(String[] args){
   Scanner input = new Scanner(System.in);
   int[][] A = new int[3][3];
int[][] B = new int[3][3];
int[][] C = new int[3][3];
System.out.println("Unesi elemente matrice A : ");
for (int i=0 ; i < A.length ; i++)
for  (int j=0 ; j < A[i].length ; j++){
A[i][j] = input.nextInt();
}
System.out.println("Unesi elemente matrice B : ");
for (int i=0 ; i < B.length ; i++)
for  (int j=0 ; j < B[i].length ; j++){
B[i][j] = input.nextInt();
}
System.out.println("Matrica A: ");
      for (int i=0 ; i < A.length ; i++){
         System.out.println();
         for  (int j=0 ; j < A[i].length ; j++){
            System.out.print(A[i][j]+" ");
              }
      }
System.out.println("Matrica B: ");
      for (int i=0 ; i < B.length ; i++){
         System.out.println();
         for  (int j=0 ; j < B[i].length ; j++){
            System.out.print(B[i][j]+" ");
              }
      }
System.out.println("Rezultat je: ");
for(int i=0;i<3;i++){
for(int j=0;j<3;j++){
for(int k=0;k<3;k++){
C[i][j]+=A[i][k]*B[k][j]; }
```

```
        }
}
for(int i=0;i<3;i++){
for(int j=0;j<3;j++){
System.out.print(+C[i][j]+" ");
                                }
                        }
        }
}
```

1.2 Klase i metode

U objektno-orijentisanom programiranju osnovnu ulogu imaju objekti koji sadrže i podatke i funkcije (metode). Program se konstruiše kao skup objekata koji međusobno komuniciraju. Podaci koje objekat sadrži predstavljaju njegovo stanje, dok pomoću metoda on to stanje menja i komunicira sa drugim objektima. Java je potpuno objektno-orjentisan jezik i zato je sav izvorni kod u Javi podeljen u klase (koje se nalaze u istoj ili različitim izvornim datotekama).

Dakle, u osnovi svega je objekat koji:

- ima podatke
- ima metode
- ima jedinstvenu adresu u memoriji
- predstavlja instance neke konkretne klase

Klasom se opisuju :

- objekti sa istim karakteristikama (podaci članovi)
- ponašanjem (funkcionalnostima –metode)

Podaci članovi (atributi):

- svaki objekat ima sopstvene vrednosti podataka članova
- trenutne vrednosti podataka objekta čine trenutno stanje objekta

Funkcije članice (metodi):

- njima su definisana ponašanja objekta
- poziv metoda jednog objekta –slanje poruke
- obrada zahteva tj. odgovaranje na poruku

Metodi se definišu samo kao deo klase. Pozivi pogrešnih metoda za neki objekat se registruju pri kompajliranju.

1. Formirati klasu pravougaonik koji omogućava rad sa pravougaonicima.

PODACI/ **ATRIBUTI**	*x –kordinata i* *y-kordinata gornjeg levog ugla* *širina (s) i visine (v)*	*-svi podaci su (int).*
METODE	*- konstruktor sa inicijalnim vrednostima (formira pravaugaonik iz temena (0,0) širine 10 i visine 10)* *-konstruktor sa datom x-kord., y-kord., širinom i visinom*	

REŠENJE

// Na samom početku definisemo glavnu klasu. Zatim definisemo atribute koji su tipa ***int.***

```
class Pravougaonik{
        int x;
        int y;
        int s;
        int v;
```

// Zatim navodimo konstruktor klase Pravougaonik, uvek prvo navodimo konstruktor sa inicijalnim vrednostima, navodimo vrednosti koje su date u zadatku. ***Konstruktor mora imati isto ime kao i klasa.***

```
Pravougaonik(){
                x=0;
                y=0;
                s=10;
                v=10;
        }
```

// Sada navodimo konstruktor sa referencama(pokazivačima), navodimo argument u zagradi, i njihove tipove podataka.Ovaj konstruktor sadrži pokazivače na atribute.

```
        Pravougaonik(int x,int y,int s,int v){
                this.x=x;
                this.y=y;
                this.s=s;
                this.v=v;
}
```

NAPOMENA
Konstruktori
**metode za kreiranje objekata date klase*
**isto ime kao i klasa, bez tipa povratne vrednosti*
**različiti oblici – različite signature (API)*
Default konstruktor
Java platforma ga automatski kreira ako klasa ne deklariše nijedan konstruktor eksplicitno
– na taj način, svaka klasa ima bar jedan konstruktor
- nema parametre (argumente) i ne radi ništa

2. Odraditi metod za odredjivanje koordinata donjeg desnog temena

desna_donja_X()-> (int) i

desna_donja_Y()-> (int)

REŠENJE:

```
int desna_donja_X()
        {
         return (x+s);
         }

int desna_donja_Y(){
         return (y+v);
         }
```

NAPOMENA

Metode i njihovi parametri

Sintaksa definisanja metoda (u smislu navođenja njihovog imena, liste parametara i tipa rezultata) je nalik sintaksi u jeziku C++.

Metode kao i atributi mogu biti tipa **String**, **int,double, boolean**, **void itd.**

3. Odraditi metod za odredjivanje duzine dijagonale($d^2=s^2+v^2$) *dijagonala()-> (double)*

REŠENJE:

```
double dijagonala(){
        double d;
        d=Math.sqrt(Math.pow(s,2)+Math.pow(v,2));
    return(d);
    }
```

4. Odraditi metod za odredjivanje površine (s*v).

povrsina() -> (int)

REŠENJE:

```
 int povrsina(){
        return(s*v);
        }
}
```

Izvršna klasa za navedenu klasu – Pravougaonik i način pozivanja navedenih metoda je:

```
class Test{

  public static void main(String[] Arguments){
  Pravougaonik p1=new Pravougaonik(10,10,20,30);
  Pravougaonik p2=new  Pravougaonik(20,30,20,30);
```

```
        int x=p1.desna_donja_X();
         int y=p1.desna_donja_Y();
  System.out.print("Koordinate p2 pravougaonika su: x="+p2.x+" y="+p2.y);
  System.out.print("Dijagonala pravougaonika je "+p1.dijagonala());
System.out.print("Površina pravougaonika je "+p1.povrsina());
  }
}
```

p1 i p2 predstavljaju instance (objekte,primerke) klase pravougaonik.

NAPOMENA
Odnos klase i objekta:

- Klasa je prototip objekta (mustra)
- Objekti se kreiraju instanciranjem klasa
- Klasa određuje tip objekta

5. Formirati klasu Datum koja omogućava rad sa datumima.

PODACI	*- d-dan,m-mesec i g-godina*	*svi podaci su celi brojevi.*
METODI	*- konstruktor sa inicijalnim vrednostima (postavlja datum na 01.01.2011.)* *- konstruktor sa datim vrednostima d-dan,m-mesec i g-godina* *metod za odredjivanje dana ,meseca I godine*	*d()->(int)* *m()-> (int)* *g()->(int)*

REŠENJE:

```
class Datum{
        int d;
        int m;
        int g;

Datum(){
                d=1;
                m=1;
                g=1;
        }
```

```
        Datum(int d,int m,int g){
                this.d=d;
                this.m=m;
                this.g=g;
        }
int dan(){
        return (d);
        }
int mesec(){
         return (m);
        }
 int godina(){
        return (g);
        }
```

NAPOMENA

Tip podataka je skup vrednosti koje podatak može da ima**,** memorijski prostor potreban za smeštanje podatka**,** operacije koje mogu da se vrše nad podatkom.

6. Odraditi metod za odredjivanje da li je godina prestupna i metod za određivanje da li je ispravan datum.

prestupna()-> (int) ispravan() -> (int)

REŠENJE:

```
int prestupna()
{
        int p;
        p=(g%4!=0 || (g%100==0 && g%400!=0))?0:1;
      return(p);
      }

int ispravan(){
        int i;
        if (g<0) return 1;
        if (d<0) return 1;
        switch(m){
                case 1:
                case 3:
                case 5:
```

```
                case 7:
                case 8:
                case 10:
                case 12: if(d<=31)
                            return 0;
                         else
                            return 1;
            case 2: if(d<=(28+this.prestupna()))
                        return 0;
                    else
                        return 1;
            case 4:
            case 6:
            case 9:
            case 11: if(d<=30)
                        return 0;
                     else
                        return 1;
           default: return 1;
        }
        }
}
```

7. Napisati program koji izračunava površinu pravougaonika i ispisuje kordinate X i Y (omogućiti unos i iskoristiti već definisanu metodu za površinu).

REŠENJE:

```
class Test{
public static void main (String[] args) {

System.out.print("Unesite koordinatu X: ");
Scanner x1 = new Scanner(System.in);
int x = x1.nextInt();

System.out.print("Unesite koordinatu Y: ");
Scanner y1 = new Scanner(System.in);
 int y = y1.nextInt();

System.out.print("Unesite sirinu S: ");
```

```
Scanner s1 = new Scanner(System.in);
 int s = s1.nextInt();

System.out.print("Unesite visinu V: ");
Scanner v1 = new Scanner(System.in);
int v = v1.nextInt();
Pravougaonik p1=new Pravougaonik(x,y,s,v);
System.out.println("Povrsina pravougaonika je "+s*v);
System.out.println("Koordinate pravougaonika su "+ p1.x +" i "+ p1.y);
  }
}
```

Metode

Dat je uporedni prikaz metoda tipa **String**, **int** i **void,**i način njihovog pozivanja u izvršnoj klasi.

INT METODA	**VOID METODA**	**STRING METODA**

Int Popust(){
If(Kolicina>=30 && Cena-Nabavna_cena>=2500)
return 0;
Else return 1;
}
	Void Popust(){
	If(Kolicina>=30 && Cena-Nabavna_cena>=2500)
	System.out.println("artikal zadovoljava uslov");
	Else
	system.out.println("artikal ne zadovoljava uslov");
	}
		String Popust(){
		If(Kolicina>=30 && Cena-Nabavna_cena>=2500)
		return ("zadovoljava");
		Else return ("ne zadovoljava");
		}

Način pozivanja u test klasi
If(a.popust()==0)
System.out.println("artikal zadovoljava uslov za popust");
Else
system.out.println("artikal ne zadovoljava uslov za popust");
	Način pozivanja u test klasi
	
	A.popust ();
		Način pozivanja u test klasi
		
		System.out.println("proizvod uslov za popust "+a.popust());

1.3 Nasleđivanje

Nasleđivanje predstavlja jedan od osnovnih koncepata objektno-orijentisanog programiranja. Pomoću koncepta nasleđivanja, klasa koja je izvedena iz osnovne klase, nasleđuje sve atribute i metode osnovne klase.

Java osnovnu klasu nazivamo nadklasa (superclass) dok izvedenu klasu nazivamo podklasa (subclass).

U Javi klase se nasleđuju korišćenjem ključne reči extends. Opšti oblik definicije nasleđivanja klasa je:

class Podklasa extends Nadklasa // Klasa Podklasa nasledjuje klasu Nadklasa

Podklasa može da nasledi samo jednu klasu (za razliku od C++ koji dozvoljava višestruko nasleđivanje klasa).

Kod nasleđivanja i nadklasa i podklasa mogu da imaju static metodu main. U zavisnosti od imena datoteke, u kojoj se nalaze klase, izvršava se static metoda main klase, koja ima isti naziv kao i ime datoteke.

8. Nadovezati se na zadatak (klasa Pravougaonik) . Formirati podklasu puni_pravougaonik koja nasledjuje klasu pravougaonik i sadrži sledeće:

-PODACI	*boja*	*tipa string*
predefinisati:	*- konstruktor sa inicijalnim vrednostima (boja="bela")* *-konstruktor sa referencama(x,y,s,h,boja)*	

REŠENJE:

// Navodimo klasu Puni_Pravougaonik koja nasleđuje klasu Pravougaonik, dodajemo nove attribute,na isti način navodimo konstruktor sa inicijalnim vrednostima i sa referencama za klasu puni_pravougaonik. Prilikom nasledjivanja navodi se metoda super(),kako bi klasa nasledila i ostale attribute od glavne klase.

```
class Puni_pravougaonik extends Pravougaonik{
        String boja;
         Puni_pravougaonik(){
                super();
                boja="bela";
        }
Puni_pravougaonik(int x,int y,int s,int v,String boja){
                super(x,y,s,v);
                this.boja=boja;
        }
}
```

9. Formirati podklasu kvadrat koja nasleđuje klasu Pravougaonik i ima sve podatke iste, ali su visina i širina uvek jednake.

predefinisati:	*- konstruktor sa inicijalnim vrednostima (super)* *-konstruktor sa referenacma(x,y,s) koji postavja s i v na s*

REŠENJE:

// Navodimo klasu Kvadrat koja nasledjuje klasu Pravougaonik,dodajemo nove attribute, na isti način navodimo konstruktor sa inicijalnim vrednostima i sa referenacma za klasu kvadrat.

```
class Kvadrat extends Pravougaonik {
        Kvadrat(){
                super();  }
        Kvadrat(int x,int y,int s){
                super(x,y,s,s);    }
}
```

10. Nadovezati se na zadatke (klasa Pravougaonik) i napisati klasu test koja će biti izvršna i u njoj:

-formirati p1 pravaougaonik(10,10,20,30)
-formirati p2 pravougaonik (12,12,16,26,crveni)
-formirati k1 kvadrat (14,16,14)
-formirati p3 pravaugaonik iz donjeg desnog temena prvog pravougaonika visine 14 i širine 15.

Na standardni izlaz ispisati: duzinu dijagonale p2 , površinu kvadrata k1 i koordinate gornjeg levog temena p3.

REŠENJE:

```
class Test{
  public static void main(String[] Arguments){
  Pravougaonik p1=new Pravougaonik(10,10,20,30);
  Puni_pravougaonik p2=new     Puni_pravougaonik(12,12,26,36,"crvena");
  Kvadrat k1=new Kvadrat(14,16,14);
// operator new *rezerviše memorijski prostor za objekat
//vraća referencu na objekat
  int x=p1.desna_donja_X();
  int y=p1.desna_donja_Y();
  Pravougaonik p3=new  Pravougaonik(x,y,20,30);
  System.out.print("Koordinate p3 pravougaonika su: x="+p3.x+" y="+p3.y);
  System.out.print("Dijagonala pravougaonika je "+p2.dijagonala());
  System.out.print("Povrsina kvadrata je "+k1.povrsina());
  }
}
```

NAPOMENA
Modifikatori pristupa
U Javi postoje sledeća tri modifikatora pristupa:
• **public**: označava da su atribut ili metoda vidljivi za sve klase u programu
• **protected**: atribut ili metoda su vidljivi samo za klase naslednice
• **private**: atribut ili metoda su vidljivi samo unutar svoje klase
• nespecificiran (tzv. *friendly*): atribut ili metoda su vidljivi sa klase iz istog Paketa

Nadovezati se na zadatak (klasa Datum) i formirati podklasu p_n_e koja sadrži:

-PODACI	*era (“nove ere” ili “pre nove ere”)*	*tipa string*
predefinisati:	*konstruktor sa inicijalnim vrednostima (era=”nove ere”)* *- konstruktor sa referencama(d,m,g,era)* *- metodu ispis()->void za ispisivanje datuma na standardni izlaz.(d.m.g.era).*	

REŠENJE:

```
class P_n_e extends Datum {
        String era;
         P_n_e(){
                super();
                era="nove ere";
         }
         P_n_e(int d,int m,int g,String era){
                super(d,m,g);
                this.era=era;
         }
         void ispis(){
         System.out.println("Datum:"+d+". "+m+". "+g+". "+era+"\n"); }
}
```

12. Formirati podklasu datum2010 koja ime sve podatke kao datum ali joj je godina uvek 2010.

predefinisati:	*- konstruktor sa inicijalnim vrednostima (g=2010)* *-konstruktor sa referencama(d,m)* *-metodu za ispisivanje datuma na standardni izlaz*

REŠENJE:

```
class Datum2010 extends Datum {
        Datum2010(){
                super();
                g=2005;
        }
        Datum2010(int d,int m){
                super(d,m,2010);
        }

         void ispis(){
                System.out.println("Datum: "+d+". "+m+". 2010."+"\n");
         }
}
```

NAPOMENA
Redefinisanje metoda

Redefinisanje metoda (*method overriding*) je postupak kada klasa naslednica redefiniše telo metode nasleđene od roditeljske klase. U Javi se to specificira prostim navođenjem nove definicije metode u klasi naslednici.

13. Napisati klasu test koja će biti izvršna i u njoj:

-formirati datume D1(1,1,2000) i D2(29,2,2010)
-formirati datume D3(1,2,3,"pre nove ere") i D4()
-formirati D (11,6) ove godine
-Na standardni izlaz ispisati:
-datum D1,D2
-da li je godina prestupana u datumu D2
-da li je ispravan D1
-datume D3 , D4
- datum D

REŠENJE:

```
class Test{
  public static void main(String[] Arguments){

  Datum d1=new          Datum(1,1,2000);
  Datum d2=new          Datum(29,2,2005);
  P_n_e d3=new          P_n_e(1,2,3,"pre nove ere");
  P_n_e d4=new          P_n_e();
  Datum2005 d=new       Datum2005(11,6);
  System.out.println("Datum d1:"+d1.dan()+". "+d1.mesec()+". "+d1.godina()+"\n");
  System.out.println("Datum d2:"+d2.dan()+". "+d2.mesec()+". "+d2.godina()+"\n");
  if(d1.ispravan() == 0)
   System.out.println("d1 je ispravan datum!");
  else
   System.out.println("d1 nije ispravan datum!");
  if(d2.ispravan() == 0)
    System.out.println("d2 je ispravan datum!");
  else
    System.out.println("d2 nije ispravan datum!");
  d3.ispis();  d4.ispis(); d.ispis();
          }
}
```

1.4 Složeni zadaci

14. Formirati klasu Tačka koja omogućava rad sa tačkama.

-PODACI	*- x –kordinata i y-kordinata*	*svi podaci su realni brojevi.*
-METODI	*- konstruktor sa inicijalnim vrednostima (tačku sa koordinatama (0,0))* *- konstruktor sa datom x-kord., y-kord.* *- metod za odredjivanje koordinata* *- metod za odredjivanje rastojanja izmedju dve tacke ($d^2=(x_1+x_2)^2+(y_1+y_2)^2$)* *- metod za odredjivanje rastojanja od centra ($d^2=x^2+y^2$)*	*x ()-> (int)* *y ()-> (int)* *rastojanje(Tacka t)-> (double)* *do_centra() -> (double)*

Formirati podklasu Imenovana_tacka koja sadrži dodatno

PODACI	*ime*	*tipa string*
predefinisati:	*- konstruktor sa inicijalnim vrednostima (ime="tacka")* *- konstruktor sa zadatim vrednostima(x,y,ime)* *- metodu za ispisivanje imena tačke na standarni izlaz*	*zovem_se()->string*

Formirati podklasu Tacka3D koja sadrži dodatno

-PODACI	*-z-koordinata*
predefinisati:	*- konstruktor sa inicijalnim vrednostima (z=0)* *-konstruktor sa referencama(x,y,z)* *-metodu za odredjivane rastojanja od centra($d^2=x^2+y^2+z^2$)*

Napisati klasu test koja će biti izvršna i u njoj:

-formirati tačke A(1,2) i B(3,9)
-formirati tačku D(12,12,moja_tacka) i E()
-formirati C tačku (1,2,3)

Na standardni izlaz ispisati:

-rastojanje AB
-rastojanje A do koordinatnog pocetka
-rastojanje A do koordinatnog pocetka
-kordinate tacke C
- "A projekcija C" ako su im x i y koordinate iste
-ispisati ime za E i D koriteci metodu zovem_se

REŠENJE:

```
import java.math.*;

class Tacka{
        float x;
        float y;

        public Tacka(){
                this.x=0;
                this.y=0;
        }
        public Tacka(float x, float y){
                this.x = x;
                this.y = y;
                }
        public float x(){
                return this.x;
        }
        public float y(){
                return this.y;
                }
        public double Rastojanje(Tacka t){
                double d;
                d=Math.pow(this.x - t.x,2) + Math.pow(this.y - t.y,2);
                Math.sqrt(d);
```

```
                return(d);
                }
                public double DoCentra(){
                        double d;
                        d=Math.pow(this.x,2) + Math.pow(this.y,2);
                        Math.sqrt(d);
                        return(d);
                        }
}
class ImenovanaTacka extends Tacka{
        String ime;
        ImenovanaTacka(){
                super();
                this.ime = "tacka";
                }
                public ImenovanaTacka(float x, float y, String ime){
                        super(x,y);
                        this.ime = ime;
                        }
                public void  ZovemSe(){
                        System.out.println("Ime tacke je : " + ime);
                }
}
class Tacka3D extends Tacka{
        float z;
        public Tacka3D(){
                super();
                this.z = 0;
        }
        public Tacka3D(float x, float y, float z){
                super(x,y);
                this.z = z;
                }
        public double OdCentra(){
        double d;
        d = Math.pow(this.x,2) + Math.pow(this.y,2) + Math.pow(this.z,2) ;
                        Math.sqrt(d);
                        return(d);
                        }
        }
```

```
public class Test{
public static void main(String[] args){
        Tacka A = new Tacka(1,2);
        Tacka B = new Tacka(3,9);
        ImenovanaTacka D= new ImenovanaTacka(12,12,"Moja_Tacka");
        ImenovanaTacka E= new ImenovanaTacka();
        Tacka3D C = new Tacka3D(1,2,3);
System.out.println("Rastojanje izmedju tacaka A i B je : "+A.Rastojanje(B));
System.out.println("rastojanje do koordinatnog pocetka tacke A : " + A.DoCentra());
        System.out.println("X koordinata tacke C :" + C.x);
        System.out.println("Y koordinata tacke C :" + C.y);
        System.out.println("Z koordinata tacke C :" + C.z);
        if(A.x()==C.x() && A.y()==C.y())
        System.out.println("A je projekcija tacke C");
        E.ZovemSe();
        D.ZovemSe();
        }
}
```

Polimorfizam je koncept koji omogućava objektima da ispolje različito ponašanje, zavisno od njihove klase, bez obzira što se oni koriste kao instance nekog zajedničkog roditelja.

15 . Formirati klasu SemaforRG sa sledećim podacima:

Podaci	-crveno -zeleno	*svi podaci su (int).*

-METODI	- *konstruktor sa inicijalnim vrednostima* (crveno=0 i zeleno=1)
	- konstruktor sa referencama
	- metod za odredjivanje crvenog I zelenog svetla,
	- metod za odredjivanje koje je svetlo aktivno,
	-metod za promenu boje na semaforu,
	-metod za ispis koje je svetlo uključeno,
	-metod za odredjivanje da li je stanje semafora ispravno.
		int Crveno();
		int Zeleno();
		int Aktivno();
		void Promeni_boju();
		void Ispisi();
		String Ispravan ();

(b) Formirati podklasu SemaforRGY koja sadrži dodatne podatke

Podatak	-žuto	*Podatak je (int).*
predefinisati:	- *konstruktor sa inicijalnim vrednostima* (zuto=0) -konstruktor sa zadatim vrednostima (crveno,zeleno,zuto) -metod za ispis koje je svetlo uključeno.	Ispisi_zuto(); **metoda koja proverava da li je uključeno zuto svetlo*

Napisati klasu test koja ce biti izvršna i u njoj formirati objekte(instance):

-formirati s1 semafor sa vrednostima(1,1)
-formirati s2 semafor sa vrednostima(0,1)
-formirati s3 semafor sa vrednostima(0,0,1)

Na standardni izlaz ispisati:

-Vrednosti crvenog i zelenog svetla za semafor s1
-Koje je svetlo aktivno za semafor s2
-Ispravnost semafora s2
-Promeniti boju za semafor s1

-Ispisi za semafor s3 koje je svetlo aktivno,takođe i za s1.

REŠENJE:

```
class SemaforRG{
        int crveno;
        int zeleno;
        SemaforRG(){
                crveno=0;
                zeleno=1;}

        SemaforRG(int crveno,int zeleno){
                this.crveno=crveno;
                this.zeleno=zeleno;}
        int Crveno(){
                return crveno;}
        int Zeleno(){
```

```
                return zeleno;}
        int Aktivno(){
                if(crveno==1 && zeleno==0) return 1;
                if(zeleno==1 && crveno==0) return 2;
                else return 0;}
        void promeniBoju(){
         if (crveno==1) zeleno=1; crveno=0;
         if (zeleno==1) crveno=1; zeleno=0;
}
void ispisi(){
                if(crveno==1) System.out.println("Aktivno je crveno svetlo");
                if(zeleno==1) System.out.println("Aktivno je zeleno svetlo");}
String ispravan(){
                if((crveno==1) && (zeleno==0)) return("ispravan");
                if((crveno==0) && (zeleno==1)) return("ispravan");
                if((crveno==1) && (zeleno==1)) return("neispravan");
                if((crveno==0) && (zeleno==0)) return("neispravan");
                else return("neispravan");
        }}
```

class SemaforRGY extends SemaforRG{

```
        int zuto;
        SemaforRGY(){
                super();
                zuto=0;}

        SemaforRGY(int crveno,int zeleno,int zuto){
                super(crveno,zeleno);
                this.zuto=zuto;}

        void promeni3(){
                if(zeleno==1) zuto=1; zeleno=0;
                if(zuto==1) crveno=1; zuto=0;
                if(crveno==1) zeleno=1; crveno=0;
                }
        void ispisi_zuto(){
                this.ispisi();
                if(zuto==1) System.out.println("Aktivno je zuto svetlo");}
        }
```

```
class Test{
        public static void main(String [] Args){
        SemaforRG s1=new SemaforRG();
        SemaforRG s2=new SemaforRG(0,1);
        SemaforRGY s3=new SemaforRGY();
        SemaforRGY s4=new SemaforRGY(1,0,0);
System.out.println("Vrednost crvenog svetla s1 je:"+s1.Crveno());
System.out.println("Vrednost zelenog svetla s1 je:"+s1.Zeleno());

if(s2.Aktivno()==1) System.out.println("Aktivno je crveno svetlo kod s2");
if(s2.Aktivno()==2) System.out.println("Aktivno je zeleno svetlo kod s2");
if(s2.Aktivno()==0) System.out.println("Nepravilno stanje kod s2");
        s2.promeniBoju();
           s2.ispisi();
                System.out.println("s2 je "+s2.ispravan());
                s4.promeni3();
                s4.zuto;
                s4.ispisi();}
}
```

Klasa *String* se nalazi u paketu *java.lang* i predstavlja string kao tip podatka (sećamo se da stringovi nemaju odgovarajući primitivni tip u Javi). U tom smislu, klasa String je kao i svaka druga Java klasa osim što ima donekle specijalan tretman od strane kompajlera. U tom smislu, "tekst" u Java programu predstavlja objekat klase *String* čija je vrednost inicijalizovana na dati tekst.

Zbog toga je sasvim ispravno pisati:

String x = "tekst"; što ima isti efekat kao i

String x = new String("tekst");

16. Formirati klasu Vozilo :

PODACI	p-potrošnja po predjenim kilometrima v-prosecna brzina gorivo-vrsta goriva(“benzin”, “nafta”)	P,v-(ceo broj) gorivo- (string)

METODI	- *konstruktor sa inicijalnim vrednostima* (postavlja p=10 , v=100 i gorivo=”nafta”) - konstruktor sa datim vrednostima p-potrosnja,v-pr.brzina ,gorivo - metode za odredjivanje potrošnje i brzine, - metod za odredjivanje koliko je goriva potrebno za n kolometara (n*p) - metod za ispisivanje podataka o vozilu(“Prosecna brzina 100 km/h, potrošnja 10 l/km gorivo: nafta”)	potrosnja()->(int) brzina()-> (int) potrebno_goriva(int n)-> (int) podaci()->(void)

Formirati klasu Dizel koja nasledjuje Vozilo a vrsta goriva mu je uvek nafta, predefinisati:

- *konstruktor sa inicijalnim vrednostima* ()
- konstruktor sa zadatim vrednostima(p,v)
- metodu za odredjivanje koje je vozilo bolje

(0 ako je bolje vozilo na kome pozivamo metodu od onoga sa kojim se uporedjuje,inace 1. Bolje je ono koje je brze a ako su im brzine iste onda ono koje ima manju potrosnju). *bolje(Dizel vozilo)->(int)*

-metodu za izracunavanje koliko je potrebno vremena za n kilometara voznje(n/v) *potrebno_vremena(double n)->double*

Napisati klasu test koja će biti izvršna i u njoj:

-formirati vozila v1(), v2(13,110,”benzin”)
-formirati dizel vozila d1(), d2(7,90)
-Na standardni izlaz ispisati:
- podatke o vozilu v1 koristeci metode
- potrosnja() i brzina()
- koliko v2 potrosi goriva za 30km puta

- koliko je potrebno vremena vozilu d1 za 50km
- podatke o v2 pomoću metode podaci()
- koje je vozilo bolje d1 ili d2.

REŠENJE:

```
class Vozilo{
        int p;
        int v;
        String gorivo;

        public Vozilo(){
                p=10;
                v=100;
                gorivo="nafta";
        }
        public Vozilo(int p,int v,String gorivo){
                this.p = p;
                this.v = v;
                this.gorivo=gorivo;
                }
        public int potrosnja(){
                return p;
        }
        public int brzina(){
                return v;
                }
        public int potrebno_goriva(int n){
                return(n*p);
                }
        public void podaci(){
                System.out.println("Prosecna brzina je:"+this.v+"\n Potrosnja
                je:"+this.p+"\nGorivo:"+this.gorivo);
        }
}
class Dizel extends Vozilo{
                Dizel(){
                super();
                        }
                public Dizel(int p,int v){
                        super(p,v,"nafta");
```

```
                }
        public int bolje(Dizel vozilo){
        if(this.v> vozilo.v)
        return 0;
        else if(this.v== vozilo.v && this.p < vozilo.p)
        return 0;
        else return 1;
        }
        public double potrebno_vremena(double n){
        return(n/this.v);
        }
}

public class Test{
public static void main(String[] args){

        Vozilo v1= new Vozilo();
        Vozilo v2= new Vozilo(13,110,"benzin");
        Dizel   d1=new Dizel();
        Dizel   d2=new Dizel(7,90);
System.out.println("Prosecna brzina je:"+v1.brzina() +"\n Potrosnja
je:"+v1.potrosnja());
System.out.println("Vozilo v2 porosi za 30km: " +v2.potrebno_goriva(30));
System.out.println("Vozilo d1 predje 50 km za vreme :" + d1.potrebno_vremena(50));
        v2.podaci();
        if(d1.bolje(d2)==0)
        System.out.println("Bolje je vozilo d1");
        else
        System.out.println("Bolje je vozilo d2");
        }
}
```

17. Napisati program "Racunari", koji omogućava simulaciju prodaje računarske opreme,sa sledećim podacima:

Artikal (tip String) *Cena* *(tip int)*

Nabavna_cena (tip int) *Porez* *(tip double)*

Kategorija *(tip int) *1-Hardware,*2-Software*

1)Formirati konstruktor,za inicijalne vrednosti postaviti:
(Monitor,14500,12800,0.08,1),

2)Formirati konstruktor sa referencama,

3)Formirati metodu void Provera_Kategorija(),
koja omogucava proveru koja je kategorija artikla,
4)Formirati metodu void Ispis(),
koja omogucava kompletan ispis na standardni izlaz,
5)Formirati metodu void snizenje(),koja smanjuje cenu za 10%,za dati artikal.

Formirati klasu Hardware (atribut *Kategorija* se ne navodi), koja sadrži:

Brend *(tip String)* , *Sifra* *(tip int)*

1)Formirati konstruktor, za inicijalne vrednosti novog podatka postaviti:

(Brand="Samsung",Sifra=24),

2)Formirati konstruktor sa pocetnim vrednostima,

3)Pomocu naredbe *switch-case ili if naredbe* realizovati metodu

int Provera_Sifre(), **ispravne sifre za kategoriju Hardware su u opsegu 1-99, sve ostale su neispravne,*

4)Formirati metodu void Promena_PDV, koja menja porez za artikal,I ispisuje na standardni izlaz.

Formirati klasu Veleprodaja koja je predefinisana i koja sadrži novi podatak:

Kolicina *(tip int)*

1)Formirati konstruktor, za inicijalne vrednosti novog podatka postaviti:

(Kolicina=10),

2)Formirati konstruktor sa referencama,

3)Formirati metodu int Popust(),koja proverava sledece: *ako je količina artikla veca od 30 i razlika izmedju cene i nabave cene veca od 2500 onda artikal zadovoljava uslov za popust* (return 0),

4)Koristeći već definisanu metodu Ispis,definisati novu metodu

void Ispis_Veleprodaja().

Formirati izvrsnu klasu Test i sledeće instance:

-Računari : R1-(Procesor,3460,2850,0.08,1)
-Računari: R2-(Windows Vista,12450,8750,0.18,2)
-Hardware: H1-(WebCam,3450,2850,0.08,Genius,98)
-Veleprodaja: VP-(Tatature,840,680,0.08,15,1)
1)Na standardni izlaz ispisati:

-kompletan ispis za instance R1 I VP
-u kojoj kategoriji se nalazi artikal R2
-promeniti cenu za artikal R1 (za 10%)
-Proveriti ispravnost šifre za artikal H1
-Promeniti PDV za artikal H1 (0,18 -> 0,08)
-Proveriti da li artikal VP zadovoljava uslov za popust

REŠENJE:

class Racunari{

```
        String Artikal;
        int Cena;
        int Nabavna_cena;
        double Porez;
        int Kategorija;
Racunari(){
                Artikal="Monitor";
                Cena=14500;
                Nabavna_cena=12800;
                Porez=0.08;
                Kategorija=1;}
Racunari(String Artikal,int Cena,int Nabavna_cena,double Porez,int Kategorija){
                this.Artikal=Artikal;
                this.Cena=Cena;
                this.Nabavna_cena=Nabavna_cena;
                this.Porez=Porez;
                this.Kategorija=Kategorija;}
```

void Provera_Kategorija(){

```
if(Kategorija==1) System.out.println("Artikal:" + Artikal + " pripada kategoriji-
HARDWARE");
if(Kategorija==2) System.out.println("Artikal:" + Artikal + " pripada kategoriji-
SOFTWARE");
else System.out.println("Artikal ne pripada nijednoj kategoriji");}

void Ispis(){
System.out.println("Artikal je :" +Artikal+"\n njegova cena je: "+Cena+" nabavna
cena je: "+Nabavna_cena+" Porez je: "+Porez);
}

void Snizenje(){
        double s;
        s=Cena*0.90;
System.out.println("Artikal za koji se radi snizenje je: "+Artikal+ "\n njegova stara
cena je : "  +Cena+" a cena posle snizenja je: " +s);}
}

class Hardware extends Racunari{
                        String Brend;
                        int Sifra;
Hardware(){
                        super();
                        Brend="Samsung";
                        Sifra=24;}
Hardware(String Artikal,int Cena,int Nabavna_cena,double Porez,String Brend,int
Sifra){
                        super(Artikal,Cena,Nabavna_cena,Porez,1);
                        this.Brend=Brend;
                        this.Sifra=Sifra;}
int Provera_Sifre(){
                                if (Sifra>=1 && Sifra<=99) return 0;
                                else return 1;
        }
void Promena_PDV(){
        if (Porez==0.08) {
Porez=0.18;      System.out.println("Artikal za koji se menja porez: "+Artikal+
" novi porez je :" +Porez);}
else      {
        Porez=0.08;      System.out.println("Artikal za koji se menja porez:
"+Artikal+
```

```
        " novi porez je :" +Porez);}
        }
        }

class Veleprodaja extends Racunari{
        int Kolicina;
Veleprodaja(){
        super();
        Kolicina=10;
        }
Veleprodaja(String Artikal,int Cena,int Nabavna_cena,double Porez,int Kategorija,int
Kolicina){
        super(Artikal,Cena,Nabavna_cena,Porez,Kategorija);
        this.Kolicina=Kolicina;
        }

int Popust(){
        if(Kolicina>=30 && Cena-Nabavna_cena>=2500) return 0;
        else return 1;
        }

void Ispis_Veleprodaja(){
this.Ispis();
System.out.println("Kolicina ovog artikla je : " + Kolicina);
}
}

class Test{
public static void main(String [] Args){

        Racunari R1=new Racunari("Procesor",3460,2850,0.08,1);
        Racunari R2=new Racunari("WindowsVista",12450,8750,0.18,2);
        Hardware H1=new Hardware("WebCam",3450,2850,0.08,"Genius",98);
        Veleprodaja VP=new Veleprodaja("Tatature",840,680,0.08,15,1);
                        R1.Ispis();
                        VP.Ispis_Veleprodaja();
                        R2.Provera_Kategorija();
                        R1.Snizenje();
if (H1.Provera_Sifre()==0) System.out.println("Artikal:" +H1.Artikal+ " ima ispravnu
sifru");
else  System.out.println("Artikal:" +H1.Artikal+ " nema ispravnu sifru");
```

```
H1.Promena_PDV();
if(VP.Popust()==0) System.out.println("Artikal:" +VP.Artikal+ " ZADOVOLJAVA
uslov za popust");
else
 System.out.println("Artikal:" +VP.Artikal+ " NE ZADOVOLJAVA uslov za
popust");
                }
        }
```

18. Napisati program "Stanovnik", koji omogućava evidenciju stanovnika, sa sledećim podacima:

ime (tip String) *JMBG (tip String)*

prezime (tip String) *ime_roditelja (tip String)*

*pol (tip int)*1-muški,*2-zenski* *mesto_rodjenja (tip String)*

1)Formirati konstruktor, za inicijalne vrednosti postaviti:

(Amer,Hasic ,1203984543678, Anel, muski, Novi Pazar)

2)Formirati konstruktor sa referencama,

3)Formirati metodu void Provera_mesta(),koja omogućava proveru stanovnika Novog Pazara,

4)Formirati metodu void Ispis(),koja omogućava kompletan ispis na standardni izlaz,

5)Formirati metodu void Promena_mesta(), koja menja mesto

(Novi Pazar->Novi Sad)

Formirati klasu Stanovnik_Np (atribut *Mesto_rodjenja* se ne navodi) koja sadrži :

Dan,Mesec,Godina *(tip int)*

1)Formirati konstruktor, za inicijalne vrednosti novog podatka postaviti:

(Dan=29,Mesec=2,Godina=1984)

2)Formirati konstruktor sa referencama,

3)Pomoću naredbe *switch-case* formirati metodu

int Provera_Datuma(),koja proverava ispravnost
za ispravan datum vratiti vrednost 0,u suprotnom je vrednost 1,

4)Formirati metodu void Promena_Godine,koja menja godinu za jednu unazad.

Formirati klasu Stanovnik_Radnik koja je predefinisana, koja sadrži nove podatke:

Radno_mesto (tip String),
*Strucna_sprema (*1-srednja,2-visa,3-visoka),(tip int),*
Radni_staz (tip int)

1)Formirati konstruktor, za inicijalne vrednosti novog podatka postaviti:

(Radno_mesto="programer",Strucna_sprema=3,Radni_staz=2),

2)Formirati konstruktor sa referencama,

3)Formirati metodu int Konkurs(),koja proverava da li kandidat zadovoljava sledeći uslov: **radni staž više ili jednako od 3 godine,i visoka stručna sprema ,*

4)Koristeci već definisanu metodu Ispis,definisati novu metodu

void Ispis_Radnik().

Formirati izvrsnu klasu Test i sledeće instance:

-Stanovnik: S1-(Sead,Dulic,080598432415,Esad,muski,Novi Pazar)
-Stanovnik: S2-(Darko,Markovic,1305880345562,Dejan,muski,Novi Pazar)
-Stanovnik_Np: S3-(Emina,Eminovic,010298122245,Kemal,zenski,31,2,1981)
-Stanovnik_Radnik: S4-
(Milan,Milovanovic,080898245553,Dejan,muski,Beograd,menadzer,3,4)

Na standardni izlaz ispisati:

-kompletan ispis za instance S1 I S4
-dali je stanovnik S2 iz Novog Pazara
-promeniti mesto stanovanja stanovniku S1 (Novi Pazar->Novi Sad)
-Proveriti ispravnost datuma za stanovnika S3
-Promeniti godinu rodjenja za jednu unazad za stanovnika S3
-Proveriti da li stanovnik S4 zadovoljava uslov za konkurs.

REŠENJE:

```
class Stanovnik{
String ime; String prezime ; String JMBG; String ime_roditelja; int pol; String
mesto_rodjenja;
Stanovnik(){
                                ime="Anel"; prezime="Hasic";
                                JMBG="1203984543678";
                                ime_roditelja="Amer";
                               pol=1;mesto_rodjenja="Novi Pazar";
        }

Stanovnik(String ime,String prezime,String JMBG,String ime_roditelja,int pol,String
mesto_rodjenja){
        this.ime=ime;this.prezime=prezime;
        this.JMBG=JMBG;
        this.ime_roditelja=ime_roditelja;
        this.pol=pol;this.mesto_rodjenja=mesto_rodjenja;}

void Provera_mesta(){
if(mesto_rodjenja=="Novi Pazar") System.out.println("Stanovnik:" + ime + " " +
prezime + " je rodjen u Novom Pazaru");
else System.out.println("Stanovnik:" + ime +" " + prezime  + " nije rodjen u Novom
Pazaru");
}

void Ispis(){
        System.out.println("Stanovnik :" +ime +" " + prezime+"\n njegov JMBG je:
"+JMBG+" ime roditelja je: "
+ime_roditelja+" mesto rodjenja je: "+mesto_rodjenja);
}
void Promena_mesta(){
        String novo_mesto;
        novo_mesto="Novi Sad";
        System.out.println("Stanovnik za kojeg se menja mesto je: "+ime +" " +
prezime+ "\n njegovo staro mesto je : " +mesto_rodjenja+" a sada je: "
+novo_mesto);
        }

class Stanovnik_NP extends Stanovnik{
        int d; int m; int g;
```

```
Stanovnik_NP(){
                super();
                d=29;m=2;g=1984;
        }

Stanovnik_NP(String ime,String prezime,String ime_roditelja,String JMBG,int pol,int
d,int m,int g){
                super(ime,prezime,ime_roditelja,JMBG,pol,"Novi Pazar");
                this.d=d; this.m=m; this.g=g;
        }

int prestupna(){
 int p;
        p=(g%4!=0 || (g%100==0 && g%400!=0))?0:1;
           return(p);
        }

int Provera_datuma(){
                int i;
                if (g<0) return 1; if (d<0) return 1;
                switch(m){
        case 1:case 3:case 5:case 7:case 8:case 10:case 12:
        if(d<=31) return 0; else return 1;
        case 2: if(d<=(28+this.prestupna())) return 0; else return 1;
        case 4:case 6:case 9:case 11: if(d<=30) return 0;elsereturn 1;
        default: return 1;
         }
        }

void Promena_godine(){
        int nova_godina;
        nova_godina=g-1;
System.out.println("Stanovnik za kojeg se menja godina rodjenja je: "+ime +" " +
prezime+ "\n stari podatak je : " +g+" a sada je: " +nova_godina);
}
}

class Stanovnik_Radnik extends Stanovnik{
        String Radno_mesto; int Strucna_sprema; int Radni_staz;
Stanovnik_Radnik(){
        super();
```

```
        Radno_mesto="programer";Strucna_sprema=3; Radni_staz=2;
        }
Stanovnik_Radnik(String ime,String prezime,String ime_roditelja,String JMBG,int
pol,String mesto_rodjenja,String Radno_mesto,int Strucna_sprema,int radni_staz){
        super(ime,prezime,ime_roditelja,JMBG,pol,mesto_rodjenja);
        this.Radno_mesto=Radno_mesto;
        this.Strucna_sprema=Strucna_sprema;
        this.Radni_staz=Radni_staz;
        }

int Konkurs(){
if(Radni_staz>=3 && Strucna_sprema==3) return 1;else return 0;
}

void Ispis_Radnik(){
this.Ispis(); System.out.println("Radno mesto je : " + Radno_mesto);}
}

class Test{
public static void main(String [] Args){

Stanovnik S1=new Stanovnik ("Sead","Dulic","080598432415","Esad",1,"Novi
Pazar");
Stanovnik S2=new Stanovnik("Darko","Markovic","1305880345562","Dejan",1,"Novi
Pazar");
Stanovnik_NP S3=new
Stanovnik_NP("Emina","Eminovic","010298122245","Kemal",2,31,2,1981);
Stanovnik_Radnik S4=new
Stanovnik_Radnik("Milan","Milovanovic","098245553","Dejan",1,"Beograd","menadz
er"3,3);

S1.Ispis();
S4.Ispis_Radnik();
S2.Provera_mesta();

if (S3.Provera_datuma()==0)
        System.out.println("Stanovnik:" +S3.ime+ " ima ispravan datum");
else
        System.out.println("Stanovnik:" +S3.ime+ " nema ispravan datum");

S3.Promena_godine();
```

```
if(S4.Konkurs()==0)
        System.out.println("Stanovnik:" +S4.ime+ " ZADOVOLJAVA USLOV ZA
KONKURS");
else
        System.out.println("Stanovnik:" +S4.ime+ " NE ZADOVOLJAVA USLOV
ZA KONKURS");

        }
}
```

KONTROLNA PITANJA

1.Definiši sledeće pojmove:
Klasa,metoda,atribut,objekat,polimorfizam,instanca.
2.Kako se definiše konstruktor neke klase i šta on radi?
3.Koja je naredba u Javi za ispisivanje na standardni izlaz?
4. Koja je naredba u Javi za sistemski unos?
5.Kako se definišu nizovi u Javi?
6.Kako se definiše izvršna metoda i sta ona radi?
7.Koji su modifikatori pristupa u Javi?
8.Šta radi super() metoda?
9.Koja je službena reč za nasledjivanje?
10.Šta je izuzetak i kako se definiše?
11.Koja je razlika između void i int metoda i kako se one pozivaju ?
12.Šta radi službena reč import?
13.Kako se definiše instanca u Test klasi?
14.Šta je predefinisana metoda?
15.Šta je super klasa a šta podklasa?

ZADACI ZA VEŽBANJE - PRIMER TESTA BR.1

1. *Realizovati klasu ***e-ucenje*** koja sadrzi:
   -promenljive ***redni_broj, kurs, polaznik, profesor***,
   (podaci su tipa string, osim rednog broja-int)
   -konstruktore (inicijalni i sa pokazivacima)
   -metode:
   a) ***proveri ()*** - koja proverava da li je polaznik na kursu „objektno programiranje“
   b) ***ispis ()-*** koja ispisuje sve podatke na standardni izlaz
   c) ***promeni()*** - koja menja naziv kursa u „java programiranje“

   ***U „test** klasi“ navesti jednu instancu :
   (redni broj kursa=7, kurs = objektno programiranje, polaznik = Amel Suljic, profesor = Zoran lovrekovic)
   Nad ovom instancom izvrsiti sledece:
   -proveri kurs, ispisi sve na standardni izlaz i promeni kurs

2. Data je klasa **Trougao**
   sa atributima: *Stranica1,stranica2,stranica3* i
   metodama: **obim(), ispis().**
   *Realizuj klasu **Jednakokraki_Trougao** koji nasledjuje datu klasu,koja je predefinisana i koja ima:
   Atribute: *Stranica1,Stranica2,ugao*
   Metode: **obim(),** koristeci vec definisanu metodu **ispis()** odradi ispis za klasu koja nasledjuje.

3. Dat je izvorni kod u Javi.Pronadjite nepravilnosti i ispravite.

```
class SemaforRG () {
        int crveno;
        int zeleno;
        SemaforK(){
                crveno=0;
                zeleno=1;}
        int Aktivno(){
                if(crveno==1 && zeleno==0) return 1;
                if(zeleno==1 && crveno==0) return 2;
                else System.out (return 0);}
        void ispisi() {
```

```
                    if(crveno==1) System.out.println("Aktivno je crveno svetlo");
                    if(zeleno==1) return ("Aktivno je zeleno svetlo");
}
class Test{
                    SemaforRG s2=new SemaforRG(0,1);
                    System.println("Vrednost crvenog svetla s2 je:"+s1.Crveno());
                    System.out.println("Vrednost zelenog svetla s2je:"+s2.Zeleno());
                    s2.Aktivno() ;
                   s2.ispisi();
                    s2.stampaj();
                    }
```

4. Realizuj klasu **izraz** koja za date pozitivne brojeve x,y,z radi sledeće:
   -ako je x>y računa sledeće x^y / (2^z+y^4)
   -ako je y>z računa sledeće y^z / (3^x+z^6)
   -ako je z>x racuna sledece z^x/ (1+y)

   Ovo možete realizovati pomoću if-else naredbe.

5. Realizovati metodu **int** ako je data sledeca metoda:

```
void Provera_Kategorija() {
if(Kategorija==1) System.out.println("Artikal:" + Artikal + " pripada
kategoriji-HARDWARE");
if(Kategorija==2) System.out.println("Artikal:" + Artikal + " pripada
kategoriji-SOFTWARE");
                    else System.out.println("Artikal ne pripada nijednoj kategoriji");
}
```

Objasniti kako se pozivaju ove metode u Test klasi, koja je razlika izmedju ove dve metode?

ZADACI ZA VEŽBANJE - PRIMER TESTA BR.2

1. *Realizovati klasu ***e-prodavnica*** koja sadrzi:
   -promenljive ***sifra, artikal, kupac, prodavac***, (podaci su tipa string, osim sifre-int)
   -konstruktore (inicijalni i sa pokazivacima)
   -metode
   a) ***ispis ()-*** koja ispisuje sve podatke na standardni izlaz
   b) ***proveri ()*** - koja proverava da li je kupac kupio artikal „intel procesor“
   c) ***promeni()*** - koja menja naziv artikla u „intel pentium procesor“

   ***U "test** klasi“ navesti jednu instancu :
   (redni broj kursa=12, artikal = intel procesor, kupac = Emina Hasic, prodavac= Dejan Simic)
   Nad ovom instancom izvrsiti sledece:
   -proveri artikal, ispisi sve na standardni izlaz i promeni artikal

2. Data je klasa **Romb**
   sa atributima: *Stranica1,stranica2* i
   metodama: **obim(), ispis().**
   *Realizuj klasu **Trougao** koja je predefinisana, i koja nasledjuje datu klasu i koja ima:
   Atribute: *Stranica1,Stranica2,Stranica3*
   Metode: **obim(),**koristeci vec definisanu metodu **ispis()** odradi ispis za klasu koja nasledjuje.

3. Dat je izvorni kod u Javi.Pronadjite nepravilnosti i ispravite.

```
class SemaforRG{
        int crveno;
        SemaforRG(int crveno,int zeleno){
                this.crveno=crveno;
                this.zeleno=zeleno;}
int ispisi(){
                if(crveno==1) System.out.println("Aktivno je crveno svetlo");
                if(zeleno==1) System.out.println("Aktivno je zeleno svetlo");}
}
class SemaforRGY extends SemaforK{
        int zuto;
```

```
        SemaforRGY(){
                super();
                }
        SemaforRGY(int crveno,int zeleno,int zuto){
                this.zuto=zuto;} }
class void main Test(){
                SemaforRG s2=new SemaforRGY(0,1);
                SemaforRGY s3=new SemaforRGY();
                s2.ispisi();
                System.out.println("s2 je "+s2.ispravan()); }}
```

4. Realizuj klasu **izraz** koja za dati pozitivan jednocifreni broj x radi sledeće:
   -ako korisnik unese paran broj računa sledeće x^2 / (2^x+x^4)
   -ako korisnik unese neparan broj računa sledeće x^3 / (3^x+x^6)
   -ako korisnik unese 0 onda da racuna sledece x^0/(1+x)
   -ostale slučajeve onemogućiti (sve negativne brojeve isključiti kao i dvocifrene brojeve)

   Ovo možete realizovati pomoću switch-case ili if-else naredbe.

5. Realizovati metodu **int** ako je data sledeca metoda:

```
void ispisi_Humansticke(){
    if(smer=="PEDAGOGIJA") System.out.println("Student studira smer na Humanistickim naukama");
    if(smer=="VASPITAC") System.out.println("Student studira smer na Humanistickim naukama");
    else System.out.println("Student NE studira na ovom departmanu");
}
```

Objasniti kako se pozivaju ove metode u Test klasi, koja je razlika izmedju ove dve metode?

ZADACI ZA VEŽBANJE - PRIMER TESTA BR.3

1.Objasniti source kod u JAVI , i objasniti sta radi:

```
public class Primer1 {
public static void main(String[] args) {
int sum = 0;
for (int current = 1; current <= 10; current++) {
sum += current;
}
System.out.println("Sum = " + sum);
}
}
```

2.Definisati klasu Student (atributi : ime,prezime,br_indexa – svi su tipa string) a zatim definisati potrebne konstruktore.

3.Ispraviti navedeni kod,i objasniti:

```
class Krug {
   int x, y, radius;
public Krug(){
x=10;
y=5:
radius=2.06;
public Tacka (int x, int y, int radius)
        this.x = x;
        this.y = y;
        this.radius = radius;
}
```

4.Objasniti sta je navedeno sledecim kodom a zatim primeniti metodu Upali( ),i dati konkretne parametre sledecim redosledom (naziv_automobila,tip,marka,godina,km).

```
Automobil a = new Automobil();
Automobil b = new Automobil();
```

5. U Javi postoje sledeća tri modifikatora pristupa,objasniti:

- public: _____
- protected: _____
- private: _____

6.Ubaciti sledece pojmove na odgovarajuca mesta (nadklasa,podklasa,metoda,atribut)

class Avion { _____
Krilo levo, desno; _____
void poleti() { ... } _____
void sleti() { ... } _____
}
class BorbeniAvion extends Avion { _____
}

7.Sledecu int metodu definisati kao void:
int Konkurs(){

if(Radni_staz>=3 && Strucna_sprema==3) return 1;else return 0;}

8.Definisati podklasu Stanovnik_radnik (atributi: preduzece,radni_staz) koja nasledjuje glavnu klasu Stanovnik (atributi: ime, prezime, datum_rodj, grad) a zatim definisati konstruktore za klasu koja nasledjuje).

9.Definisati izvrsnu klasu i sledece instance:

Tacka A = new Tacka(1,2);
Tacka B = new Tacka(3,9);

a zatim izvrsiti metode:

- *rastojanje (tipa void) i*
- *provera (tipa int,vraca vrednosti 0-ispravan i 1-neispravan)*

10.Obajsniti sledece delove koda:

- *System.out.println* _____
- *public class HelloWorld extends Auto* _____
- *double result* _____
- *switch (s) { }* _____
- *public static void main(String[] args)* _____
- *return nickName;* _____
- *a.radi()* _____

ZADACI ZA VEŽBANJE - PRIMER TESTA BR.4

1.Objasniti source kod u JAVI (navesti koji rezultat daje na standardni izlaz):

```
public class ArithmeticDemo {
   public static void main(String[] args) {
      int i = 37;
      int j = 42;
      double x = 27.475;
      double y = 7.22;
      System.out.println("   i = " + i);   System.out.println("   j = " + j);
      System.out.println("   x = " + x);  System.out.println("   y = " + y);
      System.out.println("   i + j = " + (i + j));
      System.out.println("   x + y = " + (x + y)); }
```

2.*Definisati klasu Univerzitet (atributi : fakultet,odsek,smer – svi su tipa String) a zatim definisati potrebne konstruktore.*

3.Ispraviti navedeni kod i objasniti:

```
class Datum{
        int d;
        int m;
        int g;
        }
        Datum {
                d=1;
                m=1;
                g=2009;
        Datum1( ){
                this.d=d;
                this.m=mesec;
                this.g=godina;
        }
```

4.Objasniti sta je navedeno sledecim kodom a zatim primeniti metodu povrsina( ),i dati konkretne parametre sledecim redosledom (10,10,20,30).

```
Pravougaonik    p1=new          Pravougaonik( );
```

5. Definisati sledece pojmove:

Klasa _____

Objekat_____

Metoda_____

Tri modifikatora pristupa su:_____

6.Ubaciti sledece pojmove na odgovarajuca mesta: *klasa,metoda,atribut,objekat,standardni izlaz,izvrsna klasa ,instanca, dodela vrednosti*

class AutomatNovca{ _____

String ImeKlijenta; _____

double Stanje; _____

public static void main(String args[]){ _____

AutomatNovca k = new AutomatNovca (); _____

k.ImeKlijenta = "Pera Peric"; _____

k.Stanje = 1200; _____

System.out.println("Klijent: " + k.ImeKlijenta + ". Stanje: " + k.Stanje) ___

} }

7.Sledecu int metodu definisati kao void:

int Popust(){

if(Kolicina>=30 && Cena-Nabavna_cena>=2500) return 0; else return 1; }

8.Definisati podklasu Veleprodaja*(atributi: magacin,kolicina) koja nasledjuje glavnu klasu Prodaja(atributi: Prodavnica,artikal,tip - String) a zatim definisati konstruktore za klasu koja nasledjuje.*

9.Definisati izvrsnu klasu i sledece instance koje su tipa Student:

S1*-(Sead,Dulic,080598432415,Esad,muski,Novi Pazar)*

S2*-(Darko,Markovic,1305880345562,Dejan,muski,Novi Pazar)*

a zatim izvrsiti metode:

- *Smer_Ispis (tipa void) i*
- *Provera_brIndexa (tipa int, vraca vrednosti 0-ispravan i 1-neispravan)*

10.Obajsniti sledece delove koda:

- *Import* _____
- *public class Tacka extends Oblik* _____
- *double result* _____
- *if-else* _____
- *public static void main(String[] args)* _____
- *return result ;* _____
- *aValue = 8933.234;* _____

2 DEO

JAVA APLETI

2 JAVA APLETI

Apleti su posebna vrsta Java programa koji su namenjeni za ugrađivanje u HTML stranice. U okviru stranice aplet dobija na raspolaganje određenu pravougaonu površinu čije su dimenzije date u pikselima. U tom smislu, aplet se u HTML stranicu ugrađuje na sličan način kao i slika.

Kada Web čitač pristupi stranici koja sadrži aplet, automatski će preuzeti i programski kod apleta (prevedene Java klase), pokrenuti Java virtuelnu mašinu i početi izvršavanje apleta.

Aplet je Java program koji na raspolaganju ima gotovo sve mogućnosti klasičnih Java aplikacija, izuzev dva bitna ograničenja, uvedena iz bezbednosnih razloga:

- apleti ne mogu da pristupe fajl-sistemu računara na kome se izvršavaju;
- apleti ne mogu da uspostave mrežnu konekciju sa bilo kojim računarom osim sa Web serverom sa koga su preuzeti.

Aplet nema metodu main, tako da ga ne možemo pokrenuti na do sada poznat način – iz komandne linije. Potrebno je ovakav aplet ugraditi u HTML stranicu u okviru koje će biti prikazan. Sledi primer ovakve HTML stranice.

```
<html>
<head>
<title>Test stranica sa apletom</title>
</head>
<body>

<applet code = "AppletTest" width = 100 height = 50>
</applet>

</body>
</html>
```

public class Drugi extends Applet { }	*Definicija svih potrebnih elemenata pre init metode.*
public void init() { }	*Metoda init:* *Spajanje – element.funkcionalnost*
public void paint(Graphics g) { }	*Metoda Paint:* *Unose se svi elementi koji se iscrtavaju na Panel*
public void actionPerformed(ActionEvent e){ }	*Metoda ActionPerformed:* *Unose se svi elementi kojima se dodaje funkcionalnost*

Metode apleta i način definisanja

2.1 Rad sa komponentama i grafičkim elementima

Programski jezik Java je, u svojoj inicijalnoj verziji, posedovao biblioteku komponenti za izgradnju grafičkog korisničkog interfejsa (GUI) zvanu ***Abstract Window Toolkit*** **(AWT).**

1. Aplet koji realizuje formu na kojoj se nalaze dva dugmića.

REŠENJE:

```
import java.awt.*;
import java.applet.*;

public class Dugmici extends Applet {
   Button d1= new Button("Moje dugme");
//Kreiramo dugme sa datim tekstom kao oznkom.

   Button d2= new Button();
//Kreiramo dugme bez oznake.

public void init() {
        //Postavljamo labelu na drugo dugme.
        d2.setLabel("Drugo moje dugme");

        //Dodajemo dugmice
```

```
        add(d1);
        add(d2);
        }
}
```

U pitanju je biblioteka koja se zasniva na korišćenju komponenti korisničkog interfejsa koje su dostupne na platformi na kojoj se program pokreće (*Windows*, *Motif*, *Macintosh*, itd).

To znači da je implementacija AWT komponenti različita za svaki operativni sistem. Java klase koje predstavljaju AWT komponente koriste su u velikoj meri *native* programski kod koji je vršio interakciju sa operativnim sistemom. Svaka klasa mora da pripada nekom paketu. Ako se ne navede kom paketu pripada data klasa, podrazumeva se da pripada tzv. korenskom ili implicitnom paketu. Taj korenski paket nema posebno ime. On može da sadrži klase i potpakete, koji sa svoje strane takođe mogu da sadrže klase i potpakete.

2. Aplet koji realizuje formu na kojoj se nalaze tri dugmića sa oznakama.

REŠENJE:

```
import java.awt.*;
import java.applet.*;

public class Dugme1 extends Applet {

/*Formiramo dugmice*/
   Button crveno= new Button("CRVENO");
   Button plavo= new Button("PLAVO");
   Button zuto= new Button("ZUTO");

/*Formiramo tabelarni izgled*/
   GridLayout gl= new GridLayout(3,1,10,20);

 public void init() {

                //Postavljamo tabelarni izgled
                setLayout(gl);
```

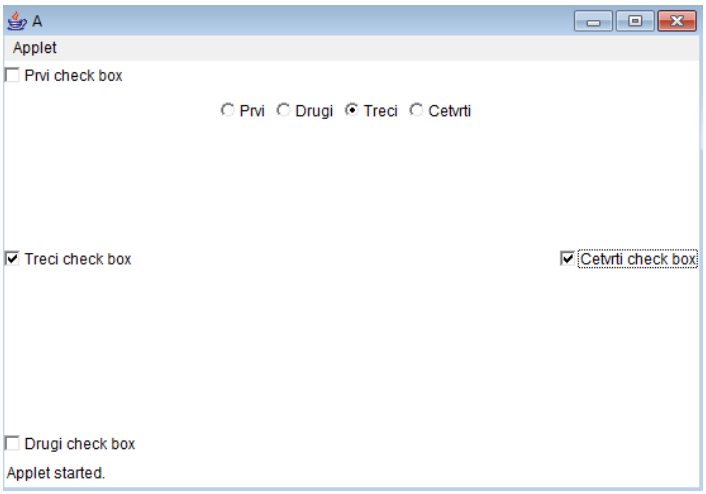
```
//Dodajemo dugmice apletu
                    add(zuto);
                    add(plavo);
                    add(crveno);
}
```

3. Aplet koji realizuje formu na kojoj se nalaze komponente: mreže, radioButton, checkButton itd.

REŠENJE:

```
import java.awt.*;
import java.applet.*;

public class Radio extends Applet {

//Formirajmo Border postavku
        BorderLayout bl=new BorderLayout();

Checkbox c1=new Checkbox("Prvi check box");

//Kreiramo tri check box-a sa odgovarajucim oznakama
   Checkbox c2=new Checkbox("Drugi check box");
   Checkbox c3=new Checkbox("Treci check box");
   Checkbox c4=new Checkbox(); //Kreiramo cetvrti bez oznake.

   //Pravimo grupisana tri checkbox-a
   CheckboxGroup  gr=new CheckboxGroup();
        Checkbox gc1=new Checkbox("Prvi",gr,true);
        Checkbox gc2=new Checkbox("Drugi",gr,false);
        Checkbox gc3=new Checkbox("Treci",gr,false);
        Checkbox gc4=new Checkbox("Cetvrti",gr,false);

//Panel na koji cemo postaviti grupu
        Panel p=new Panel();
        public void init() {

//aktivirajmo Border postavku
        setLayout(bl);
```

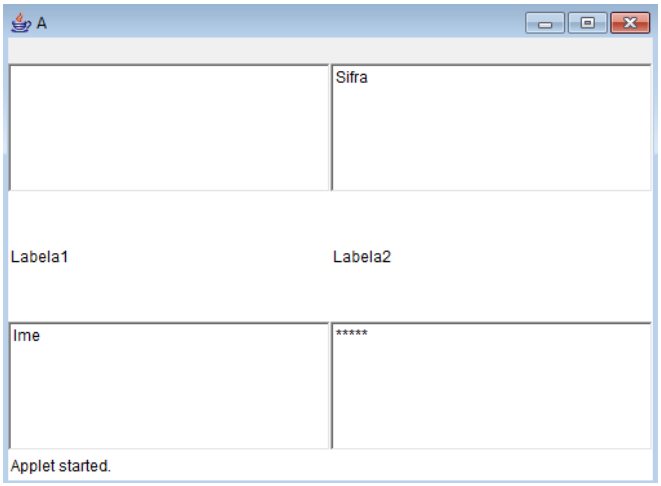
```
//Postavljamo labelu na cetvrti check box.
        c4.setLabel("Cetvrti check box");

//Postavljamo aktivan treci check box;
        c3.setState(true);

//Promeniti aktivni box
        if(!gc3.getState())
        gc3.setState(true);

//Dodajmo komponente
        add(c1,BorderLayout.NORTH);
        add(c2,BorderLayout.SOUTH);
        add(c3,BorderLayout.WEST);
        add(c4,BorderLayout.EAST);
        p.add(gc1);
        p.add(gc2);
        p.add(gc3);
        p.add(gc4);
        add(p,BorderLayout.CENTER);
        }
}
```



4. Aplet koji realizuje formu na kojoj se nalaze komponente: tekstualna polja i labele.

REŠENJE:

```
import java.awt.*;
import java.applet.*;
public class Tekst extends Applet {
//Formiramo mreznu postavku
        GridLayout gl=new GridLayout(3,2);
        TextField t1=new TextField();
//Formiramo prazno tekst polje
        TextField t2=new TextField(5);
 //Formiramo prazno tekst polje sa sirinom za prikaz 5 polja
        TextField t3=new TextField("Ime");
  //Formiramo tekst polje sa upisanim tekstom ime
        TextField t4=new TextField("Sifra",6);
//Formiramo tekst polje sa upisanim tekstom sifra i prostorom za 6 karaktera
        Label  l1=new Label();          //Prazna labela
        Label  l2=new Label("Labela2");  //Labela sa oznakom.
        public void init() {

//Postavimo mreznu postavku
        setLayout(gl);
        //Postavljamo skrivanje karakteka sifre
        t4.setEchoChar('*');
        //Onemogucimo unos karaktera u polje t1.
        t1.setEnabled(false);
        //Uzmimo tekst iz t4 i postavimo ga u t2.
        String tekst=t4.getText();
        t2.setText(tekst);
        //Postavljamo tekst labeli 1.
        l1.setText("Labela1");
        add(t1);add(t2);
        add(l1); add(l2);
        add(t3);add(t4);
        }
}
```



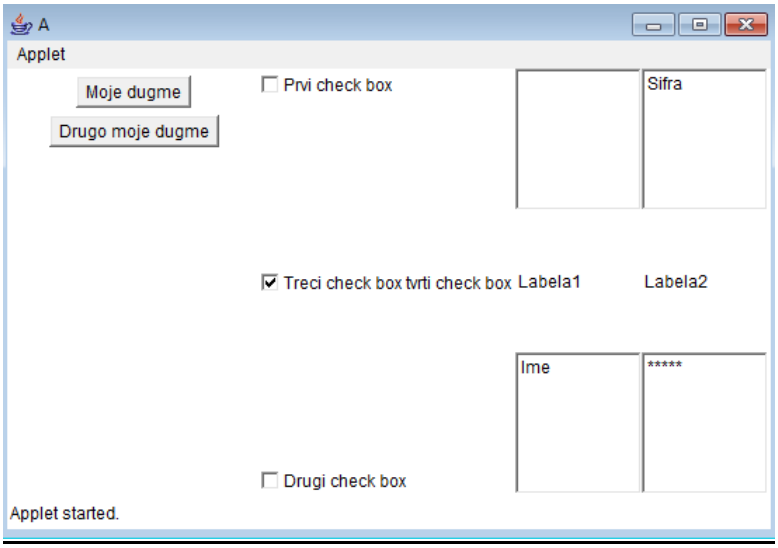
5. Aplet koji realizuje formu na kojoj se nalaze komponente: mreže, checkButton, tekstualna polja, labele,dugmad itd.

REŠENJE:

```
import java.awt.*;
import java.applet.*;

public class Komponente extends Applet {
        //Paneli za odredjene grupe
        Panel dugmici=new Panel();
        Panel radio=new Panel();
   Panel tekst=new Panel();
   Panel p1=new Panel();

//Formiramo postavke
   GridLayout gl=new GridLayout(3,2);
   GridLayout gl1=new GridLayout(4,1);
   GridLayout glavna=new GridLayout(1,3);
   BorderLayout bl=new BorderLayout();
   Button d1= new Button("Moje dugme");

//Kreiramo dugme sa datim tekstom kao oznkom.
   Button d2= new Button();          //Kreiramo dugme bez oznake.
   Checkbox c1=new Checkbox("Prvi check box");

//Kreiramo tri check box-a sa odgovarajucim oznakama
   Checkbox c2=new Checkbox("Drugi check box");
```

```
   Checkbox c3=new Checkbox("Treci check box");
   Checkbox c4=new Checkbox(); //Kreiramo cetvrti bez oznake.

//Pravimo grupisana tri checkbox-a
   CheckboxGroup  gr=new CheckboxGroup();
          Checkbox gc1=new Checkbox("Prvi",gr,true);
          Checkbox gc2=new Checkbox("Drugi",gr,false);
          Checkbox gc3=new Checkbox("Treci",gr,false);
          Checkbox gc4=new Checkbox("Cetvrti",gr,false);
          TextField t1=new TextField();        //Formiramo prazno tekst polje
          TextField t2=new TextField(5);
//Formiramo prazno tekst polje sa sirinom za prikaz 5 polja
          TextField t3=new TextField("Ime");
//Formiramo tekst polje sa upisanim tekstom ime
          TextField t4=new TextField("Sifra",6);

//Formiramo tekst polje sa upisanim tekstom sifra i prostorom za 6 karaktera
          Label  l1=new Label();          //Prazna labela
          Label  l2=new Label("Labela2");  //Labela sa oznakom.

public void init() {
          //Postavimo postavke
          setLayout(glavna);
          radio.setLayout(bl);
          tekst.setLayout(gl);
          p1.setLayout(gl1);

//Postavljamo labelu na drugo dugme.
          d2.setLabel("Drugo moje dugme");

//Postavljamo labelu na cetvrti check box.
          c4.setLabel("Cetvrti check box");

//Postavljamo aktivan treci check box;
          c3.setState(true);

//Promeniti aktivni box
          if(!gc3.getState())
          gc3.setState(true);

//Postavljamo skrivanje karakteka sifre
```

```
        t4.setEchoChar('*');

//Onemogucimo unos karaktera u polje t1.
        t1.setEnabled(false);

//Uzmimo tekst iz t4 i postavimo ga u t2.
        String tekst4=t4.getText();
        t2.setText(tekst4);

//Postavljamo tekst labeli 1.
        l1.setText("Labela1");

//Dodajemo komponente
        dugmici.add(d1);
        dugmici.add(d2);
        radio.add(c1,BorderLayout.NORTH);
        radio.add(c2,BorderLayout.SOUTH);
        radio.add(c3,BorderLayout.WEST);
        radio.add(c4,BorderLayout.EAST);
        p1.add(gc1);
        p1.add(gc2);
        p1.add(gc3);
        p1.add(gc4);
        radio.add(p1,BorderLayout.CENTER);
        tekst.add(t1);
        tekst.add(t2);
        tekst.add(l1);
        tekst.add(l2);
        tekst.add(t3);
        tekst.add(t4);

//Dodajmo jos panele
        add(dugmici);
        add(radio);
        add(tekst);
        }
}
```



6.Napisati applet koji iscrtava string - *Geometrija* od kordinate (175,50) i postavlja oko njega zaobljen pravougaonik.

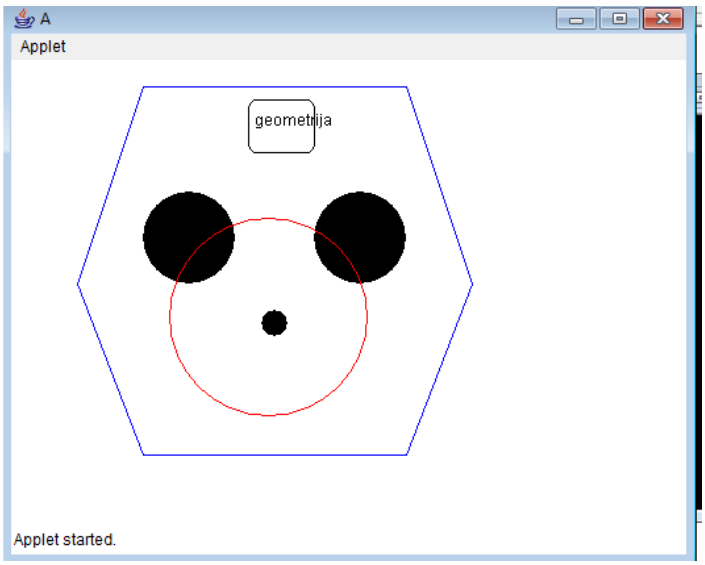
- iscrtava tri ispunjena kruga iz gornjeg levog ugla sa kordinatama i poluprečnikom:
  - (100,100), r=70
  - (230,100), r=70
  - (190,190), r=20
- postavlja boju na crvenu
- iscrtava ne ispunjen krug od koordinate (120,120) poluprečnika r=150
- postavlja boju na plavu
- iscrtava polygon kroz temena:

  A_1 =(100, 20) A_2 =(300, 20) A_3 =(350, 170)
  A_4 =(300, 300)A_5 =(100, 300) A_6 =(50, 170)

REŠENJE:

```
import java.awt.*;
import java.applet.*;
import java.awt.Polygon;
public class Drugi extends Applet {
        public void init() {
        }
        public void paint(Graphics g) {
```

```
            g.drawString("GEOMETRIJA",185, 50 );
            g.drawRoundRect(180,30,50,40,12,12);
            g.fillOval(100,100,70,70);
            g.fillOval(230,100,70,70);
            g.fillOval(190,190,20,20);
            g.setColor(Color.red);
            g.drawOval(120,120,150,150);
        int x[]={100,300,350,300,100,50,100};
        int y[]={20,20,170,300,300,170,20};
        Polygon poly=new Polygon(x,y,7);
        g.setColor(Color.blue);
        g.drawPolygon(poly);
    }
}
```



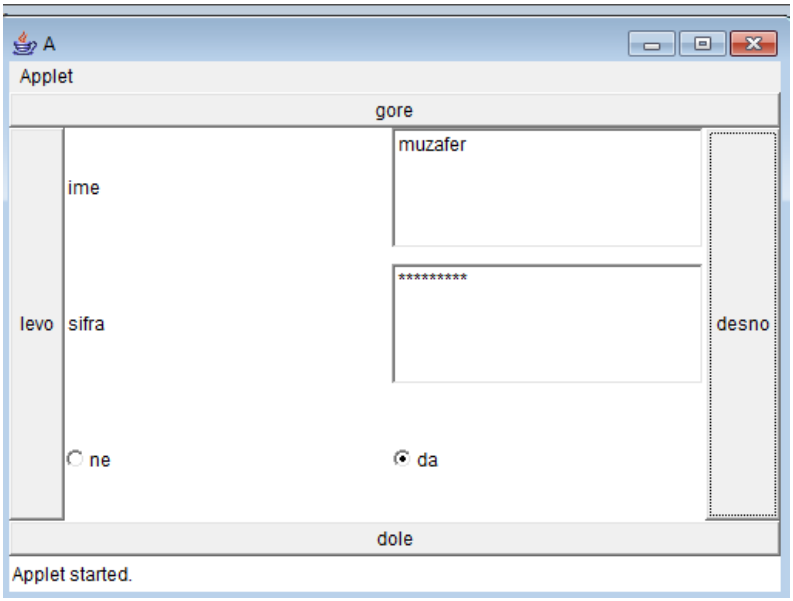
7.Napisati applet koji kreira formu sa border postavkom i postavlja dugmat: gore,dole,levo desno na odgovarajuće strane.

U centralnoj celiji postavlja panel sa mrežnom postavkom sa 3-reda i 2-kolone, i 10 piksela razmakom izmedju komponenata koji sadrži :

- labelu nadimak i tekst polje za njegov unos nadimak
- labelu šifra i tekst polje za unos
- dva checkbox-a DA i NE koja su grupisana

REŠENJE:

```
import java.awt.*;
import java.applet.*;

public class Treci extends Applet {
        BorderLayout b=new BorderLayout();
        Button gore=new Button("gore");
        Button levo=new Button("levo");
        Button desno=new Button("desno");
        Button dole=new Button("dole");
        Panel pane=new Panel();
        CheckboxGroup grupa=new CheckboxGroup();
        Checkbox ne=new Checkbox("ne",grupa,true);
        Checkbox da=new Checkbox("da",grupa,false);
        Label ime_l=new Label("ime");
        TextField ime = new TextField();
        Label sifra_l =new Label("sifra");
        TextField sifra=new TextField();
        GridLayout g=new GridLayout(3,2,10,10);

public void init() {
setLayout(b);
add("North",gore);
add("South",dole);
add("West",levo);
add("East",desno);

pane.setLayout(g);
                pane.add(ime_l);
                pane.add(ime);
                pane.add(sifra_l);
                sifra.setEchoChar('*');
           pane.add(sifra);
           pane.add(ne);
           pane.add(da);
           add("Center",pane);
        }
}
```



8.Napisati applet koji iscrtava string "APRIL" od kordinate (185,75) i postavlja oko njega .

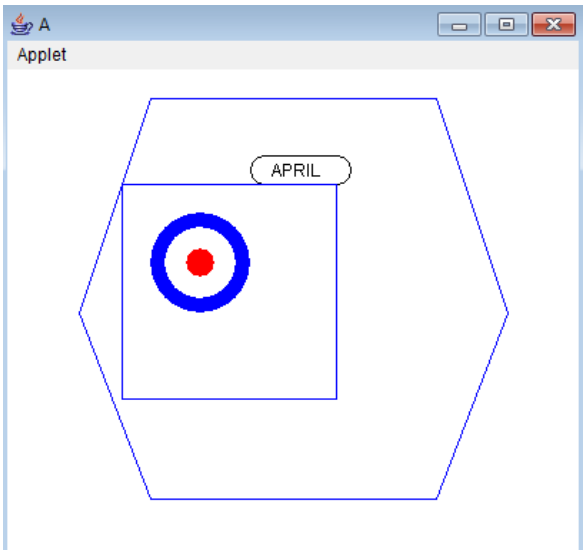
zaobljen pravougaonik
- iscrtava tri ispunjena koncentrična kruga kruga
  - (100,100), r=70,plav
  - r=50,beo
  - r=20,crven
- iscrtava ne ispunjen kvadrat plave boje od koordinate (80,80) stranice a=150
- postavlja boju na crnu
- iscrtava polygon kroz temena:

A_1 =(100, 20) A_2 =(300, 20) A_3 =(350, 170)
A_4 =(300, 300)A_5 =(100, 300) A_6 =(50, 170)

REŠENJE:

```
import java.awt.*;
import java.applet.*;
public class Drugi extends Applet {

public void init() {
                    resize(400,400);
}
```

```
public void paint(Graphics g) {
                    int niz1[] = {100,300,350,300,100,50};
                    int niz2[] = {20,20,170,300,300,170};
                    g.drawString("APRIL", 185,75 );
                    g.drawRoundRect(170,60,70,20,20,20);
                    g.setColor(Color.blue);
                    g.fillOval(100,100,70,70);
                    g.setColor(Color.white);
                    g.fillOval(110,110,50,50);
                    g.setColor(Color.red);
                    g.fillOval(125,125,20,20);
                    g.setColor(Color.blue);
                    g.drawRect(80,80,150,150);
                    g.drawPolygon(niz1,niz2,6);
          }
}
```



2.2 Rad sa događajima

U Javi, događaji se obrađuju pomoću tzv. osluškivača događaja (*event listeners*). Naime, obzirom da se jedan događaj (npr. pritisak na dugme) sastoji iz nekoliko akcija (pozicioniranje kursora miša na dugme, pritisak levog dugmeta miša, otpuštanje levog dugmeta miša), komponenta na osnovu svih tih preduzetih akcija generiše događaj neke od podklasa klase *AWTEvent*("dugme je pritisnuto"), koji se onda prosleđuje odgovarajućem osluškivaču.

Osluškivač tada preduzima određene akcije koje su potrebne da se događaj pritiskanja dugmeta obradi, bez ulaženja u detalje o pojedinačnim akcijama od kojih se taj događaj sastoji.

Osluškivači u Javi su realizovani preko interfejsa. Obzirom da postoji više vrsta događaja (podklase klase *AWTEvent*), svakom od njih odgovara i odgovarajući osluškivač pomoću koga se taj događaj obrađuje. Da biste obradili neki događaj, neophodno je da kreirate klasu koja implementira odgovarajući osluškivač. Na primer,

```
class Reakcija implements ActionListener {
..... }
```

9. Pogledajte drugi zadatak iz apleta, gde smo na panelu postavili 3 dugmeta (nefunkcionalna), a sada ćemo u ovom zadatku dodati funkcionalnost,tako da će klikom na odgovarajući button labela menjati boju.

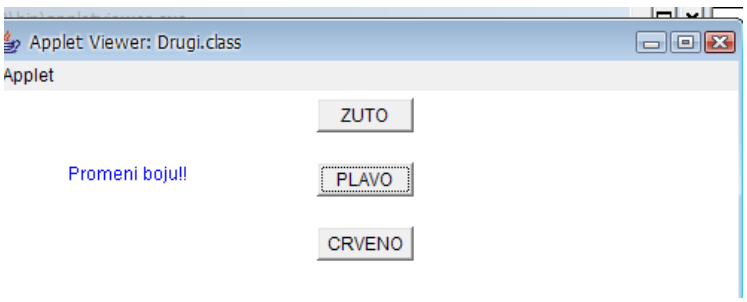
UPUTSTVO:

Kreirati tri buttona(crveno,plavo,zuto),i jednu tekst labelu,
sa sledećim pozicijama ("Promeni boju!!", 50, 60).
-Postaviti mrežu sa sledećim podacima GridLayout(3,1,10,20).
-Postaviti glavni panel na kome se nalazi mreža.
-Koristeći ActionPerformed,dodati funkcionalnost buttonima:
-klikom na dugme "crveno",tekst labela treba da bude crvene boje i tako za naredna dva dugmeta.

REŠENJE:

```
import java.awt.*;
import java.applet.*;
import java.awt.event.*;

public class Dugme3 extends Applet implements ActionListener{

/*Formiramo dugmice*/
        Button crveno= new Button("CRVENO");
   Button plavo= new Button("PLAVO");
   Button zuto= new Button("ZUTO");
   Color  boja=Color.green;
```

```
/*Formiramo planel na kome ce biti dugmici*/
   Panel p=new Panel();
/*Formiramo tabelarni izgled*/
   GridLayout gl= new GridLayout(3,1,10,20);
 public void init() {
                    //Postavljamo tabelarni izgled
                    p.setLayout(gl);
                    //Dodajemo akcije dugmicima
                    zuto.addActionListener(this);
                    plavo.addActionListener(this);
                    crveno.addActionListener(this);
                    //Dodajemo dugmice apletu
                    p.add(zuto);
                    p.add(plavo);
                    p.add(crveno);
                    add(p);
          }
public void paint(Graphics g) {
                    //Postavljamo boju kojom cemo ispisati poruku
                    g.setColor(boja);
                    g.drawString("Promeni boju!!", 50, 60 );}

public void actionPerformed(ActionEvent e){
                    if(e.getSource()==crveno)
                    boja=Color.red;
                    else if(e.getSource()==plavo)
                    boja=Color.blue;
                    else if(e.getSource()==zuto)
                    boja=Color.yellow;
                    repaint();
                    }
}
```



ActionEvent – odgovara semantičkim događajima visokog nivoa generisanim od strane komponente nad kojom je izvršena neka akcija, kao što je, na primer, pritiskanje dugmeta, potvrda unosa teksta pomoću tastera *Enter* ili izbor neke stavke iz menija. Događaji ovog tipa se prosleđuju svim osluškivačima tipa *ActionListener*.

Unutar klase *ActionEvent* postoje sledeći indikatori događaja:

ACTION_PERFORMED – akcija je izvršena,
ALT_MASK – bilo je pritisnuto dugme *Alt* za vreme izvršavanja akcije,
CTRL_MASK - bilo je pritisnuto dugme *Control* za vreme izvršavanja akcije,
SHIFT_MASK – bilo je pritisnuto dugme *Shift* za vreme izvršavanja akcije.

10.Napisati applet koji iscrtava string sa tekstom "GEOMETRIJSKA TELA"-koordinate (325, 50) i postavlja oko njega zaobljen pravougaonik (320,30,140,40,12,12).

B)iscrtava neispunjen krug - (60,60,150,150), i ispunjen krug - (60,60,150,150).

C)iscrtava neispunjen kvadrat sa koordintama - (360,130,140,140)

D)iscrtava polygon kroz temena:

A_1 =(100, 20) A_2 =(300, 20) A_3 =(350, 170)
A_4 =(300, 300)A_5 =(100, 300) A_6 =(50, 170)

E)Boju ispunjenog kruga definisati kao nova_boja*(početna boja je plava)* i dodeliti funkcionalnost na tastere:

B- "plav krug",G-"zeleni krug",Y-"zuti krug".

F)Ako se mišem klikne u domen ispunjenog kruga ,on se transformiše u manji ispunjen krug (100,100,70,70), a zatim ako se ponovo klikne unutar iscrtava se ispunjen krug - (60,60,150,150).

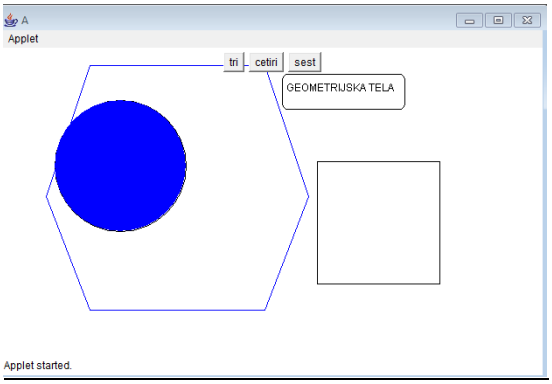
**postaviti uslov za domen* → *if(x>60 && x<150 && y>100 && y<220)*

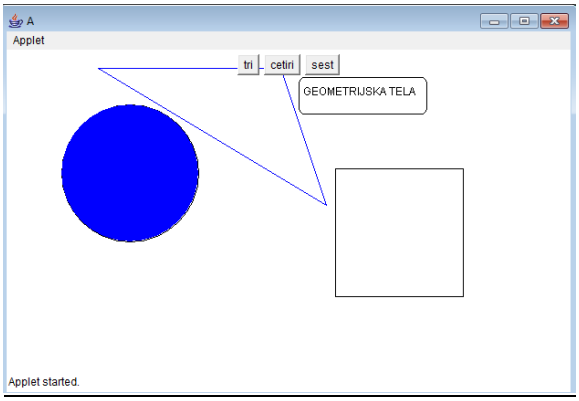
G)Formirati tri butona(tri_temena,cetiri_temena,osam_temena),klikom na jedan od tri butona iscrtava se polygon sa odredjenim brojem temena.

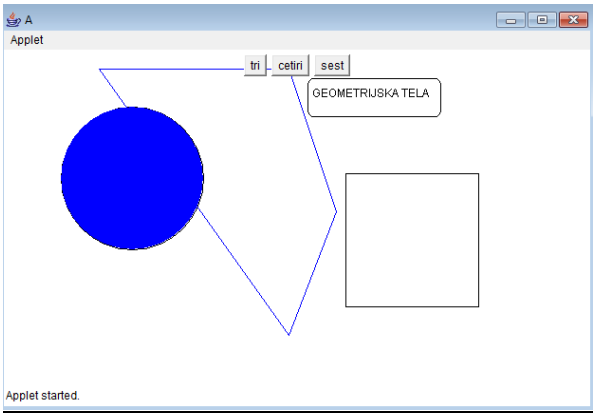
REŠENJE:

```
import java.awt.*;
import java.applet.*;
import java.awt.event.*;

public class Prvi extends Applet implements ActionListener, MouseListener,
KeyListener{

Button tri=new Button("tri");
Button cetiri=new Button("cetiri");
Button sest=new Button("sest");
        Color nova_boja=(Color.blue);
        int aktivno=1;
int tri_aktivno=0;
int cetiri_aktivno=0;
int sest_aktivno=1;

public void mouseExited(MouseEvent e){}
public void mouseEntered(MouseEvent e){}
public void mouseReleased(MouseEvent e){}
public void mousePressed(MouseEvent e){
          int x=e.getX(); int y=e.getY();
          if(x>60 && x<150 && y>100 && y<220)
          if(aktivno==1) aktivno=0;
          else aktivno=1;
          repaint();}
public void mouseClicked(MouseEvent e){}
public void keyReleased(KeyEvent e){}
public void keyTyped(KeyEvent e){}
public void keyPressed(KeyEvent e){
                switch (e.getKeyChar()){
                case 'b':nova_boja=Color.blue;break;
                case 'B':nova_boja=Color.blue;break;
                case 'y':nova_boja=Color.yellow;break;
                case 'Y':nova_boja=Color.yellow;break;
                case 'G':nova_boja=Color.green;break;
                case 'g':nova_boja=Color.green;break;}
                        repaint();
        }
```

```
public void init() {
        tri.addActionListener(this);
        cetiri.addActionListener(this);
        sest.addActionListener(this);
        add(tri);
        add(cetiri);
        add(sest);
                this.addMouseListener(this);
                this.addKeyListener(this);
        }

        public void paint(Graphics g) {
                        g.drawString("GEOMETRIJSKA TELA",325, 50 );
                        g.drawRoundRect(320,30,140,40,12,12);
                        g.drawRect(360,130,140,140);
                        g.drawOval(60,60,150,150);
                   g.setColor(nova_boja);
                   if(aktivno==0)
                        g.fillOval(100,100,70,70);else g.fillOval(60,60,150,150);
                        int niz1[] = {100,300,350,300,100,50};
                        int niz2[] = {20,20,170,300,300,170};
                        if(tri_aktivno==1)g.drawPolygon(niz1,niz2,3);
                        else if(cetiri_aktivno==1)g.drawPolygon(niz1,niz2,4);
                        else if(sest_aktivno==1)g.drawPolygon(niz1,niz2,6);
                }

        public void actionPerformed(ActionEvent e){
        if(e.getSource()==tri)
                {tri_aktivno=1; cetiri_aktivno=0; sest_aktivno=0;}
                else
        if(e.getSource()==cetiri){
                tri_aktivno=0;cetiri_aktivno=1; sest_aktivno=0;
        }
        else if(e.getSource()==sest){
                tri_aktivno=0; cetiri_aktivno=0; sest_aktivno=1;
        }
                repaint();}
        }
```







KeyEvent – generišu se od strane komponente nad kojom je pritisnut, otpušten ili otkucan neki taster sa tastature. Ovi događaji se prosleđuju osluškivaču tipa *KeyListener*. Taster je "otkucan" (*key typed* događaj), ako je pritisnut pa pušten nad nekom kontrolom. Ovo je događaj višeg nivoa, obzirom da obuhvata i slučaj kada je pritisnuta kombinacija tastera (npr. Shift + a). On se generiše kad god je unešena neka kombinacija koja predstavlja *Unicode* karakter, dok ne reaguje na funkcionalne tastere. Događaji

"*key pressed*" i "*key released*" su događaji nižeg nivoa i zavise od platforme. Pomoću njih je moguće obraditi i pritisak na tastere koji ne generišu karakter.

Unutar ove klase postoje dodatni metodi koje možete da koristite da dobijete informaciju o tasteru koji je pritisnut.

getKeyChar() – vraća karakter koji je pritisnut
getKeyCode() – vraća celobrojni kod tastera koji je pritisnut
getKeyText() – vraća string koji opisuje kod tastera (na primer *HOME* ili *F1*)

Indikatori ove vrste događaja su:
KEY_PRESSED – dugme je pritisnuto
KEY_RELEASED – dugme je pušteno
KEY_TYPED – dugme je pritisnuto i pušteno, tj. otkucan je neki karakter

MouseEvent – generiše se od strane komponente nad kojom se desila neka akcija mišem. Ova klasa se koristi za sve događaje miša (pritisak na dugme miša, pokretanje kursora, povlačenje i sl.). Svi ovi događaji se prosleđuju osluškivačima tipa*MouseListener* i *MouseMotionListener*.
Unutar ove klase postoji metod *getButton()* koji daje informaciju o tome koje dugme miša je pritisnuto. Ovaj metod može da vrati jednu od sledećih vrednosti: *BUTTON1*, *BUTTON2*, *BUTTON3* ili *NOBUTTON*.

Indikatori događaja klase *MouseEvent* su:

MOUSE_CLICKED – klik mišem (dugme je pritisnuto i pušteno),
MOUSE_DRAGGED – povlačenje mišem (dugme je pritisnuto, pomeren je kursor miša, dugme je pušteno),
MOUSE_ENTERED – kursor miša je ušao u geometrijski prostor komponente,
MOUSE_EXITED – kursor miša je napustio geometrijski prostor komponente,
MOUSE_MOVED – kursor miša je pomeren,
MOUSE_PRESSED – dugme miša je pritisnuto,
MOUSE_RELEASED – dugme miša je pušteno,
MOUSE_WHEEL – točkić miša je zarotiran.

2.3 Složeni zadaci

11.Napisati applet koji kreira glavni panel sa glavnom mrežom (3 kolone,3 reda) i postavlja :

A) PRVI RED:

- Tekst labela “PRIJAVA”

- Panel_prijava sa mrežnom postavkom Mreza_prijava(2 red, 2 kolone) i u njima :

Labela username,TextField username zatim Labela password,

*TextField password (*zasticeno)*

-postaviti dugme za prijavu

B) DRUGI RED:

-Polje Komentar za unos komentara(10,10)
-Panel_izbor sa mrežnom postavkom Mreza_izbor(3 reda,1 kolona) i u njima:

java,applet,aplikacija (dodeliti funkcionalnost dugmadima,tako da klikom na jedan od njih ispisuje se string JAVA/APPLET/APLIKACIJA, pozicija stringa(185, 50),

-Panel_pitanje sa mrežnom postavkom Mreza_pitanje (1 kolona,3 reda),i u prvom redu tekst , *"za izradu grafičkog interfejsa koristimo:"*,a zatim postaviti dva Checkbox-a (JAVA,PASCAL).

C)TRECI RED:

-Panel_radio sa Border postavkom, i redom dodeliti radio butone (prvi,drugi,treći,četvrti)

-Panel_dugmad sa mrežnom postvkom Mreza_dugmad(1 red,2 kolone) i redom dodati dugmad (Back,Next).

REŠENJE:

```
import java.awt.*;
import java.applet.*;
import java.awt.event.*;
import javax.swing.*;

public class Drugi extends Applet implements ActionListener{
        GridLayout glavna=new GridLayout(3,3);
        Panel p=new Panel();
        Label Naslovna =new Label("PRIJAVA");

        CheckboxGroup grupa=new CheckboxGroup();
        Checkbox prvi=new Checkbox("prvi",grupa,true);
        Checkbox drugi=new Checkbox("drugi",grupa,false);
```

```
Checkbox treci=new Checkbox("prvi",grupa,false);
Checkbox cetvrti=new Checkbox("cetvrti",grupa,false);

Button back=new Button("back");
Button next=new Button("next");
TextArea Komentar = new TextArea(10,10);

Panel panel_prijava=new Panel();
GridLayout mreza_prijava=new GridLayout(2,2);

Panel panel_izbor=new Panel();
GridLayout mreza_izbor=new GridLayout(3,1);
Panel panel_pitanje=new Panel();
GridLayout mreza_pitanje=new GridLayout(3,1);
Panel panel_radio=new Panel();
BorderLayout polarna=new BorderLayout();

Panel panel_dugmad=new Panel();
GridLayout mreza_dugmad=new GridLayout(1,2);

Button prijava=new Button("prijava");
Label username=new Label("username");
TextField username_polje = new TextField();
Label password =new Label("password");
TextField password_polje=new TextField();
Button java=new Button("java");
Button applet=new Button("applet");
Button aplikacija=new Button("aplikacija");
```

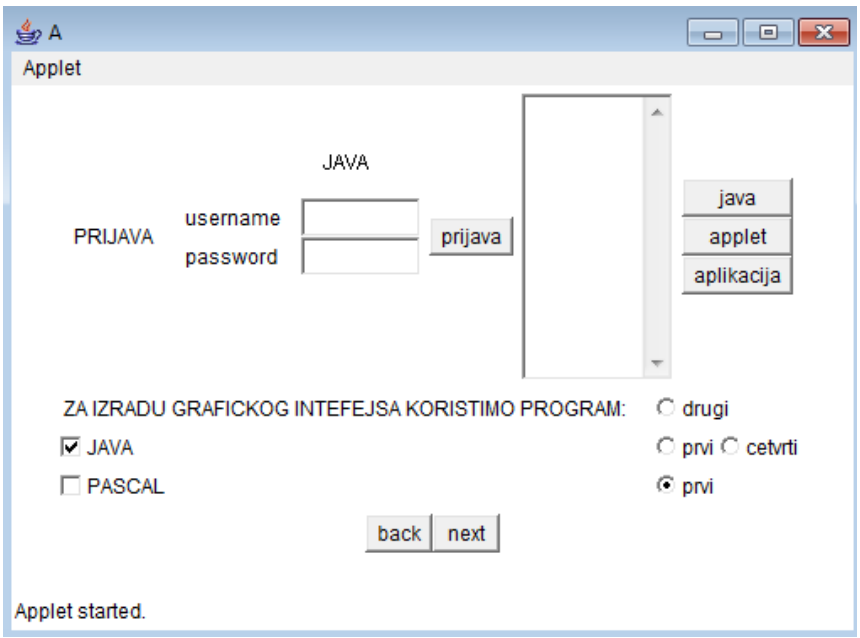
Label Pitanje =new Label(*"ZA IZRADU GRAFICKOG INTEFEJSA KORISTIMO PROGRAM:");*

```
Checkbox odgovor1=new Checkbox("JAVA",true);
Checkbox odgovor2=new Checkbox("PASCAL",false);
int jav=1;int app=0;int apl=0;
```

```
public void init() {
p.setLayout(glavna);
add(Naslovna);
add(panel_prijava);
add(prijava);
```

```
add(Komentar);
add(panel_izbor);
add(panel_pitanje);
add(panel_radio);
add(panel_dugmad);

panel_prijava.setLayout(mreza_prijava);
panel_prijava.add(username);
panel_prijava.add(username_polje);
panel_prijava.add(password);
panel_prijava.add(password_polje);
password_polje.setEchoChar('*');

java.addActionListener(this);
applet.addActionListener(this);
aplikacija.addActionListener(this);

panel_izbor.setLayout(mreza_izbor);

panel_izbor.add(java);
panel_izbor.add(applet);
panel_izbor.add(aplikacija);

panel_pitanje.setLayout(mreza_pitanje);
panel_pitanje.add(Pitanje);
panel_pitanje.add(odgovor1);
panel_pitanje.add(odgovor2);

panel_radio.setLayout(polarna);
panel_radio.add(prvi,BorderLayout.SOUTH);
panel_radio.add(drugi,BorderLayout.NORTH);
panel_radio.add(treci,BorderLayout.WEST);
panel_radio.add(cetvrti,BorderLayout.EAST);

panel_dugmad.setLayout(mreza_dugmad);
panel_dugmad.add(back);
panel_dugmad.add(next);

}
```

```
public void actionPerformed(ActionEvent e){
                    if(e.getSource()==java)
                    {jav=1;app=0;apl=0;}
                    else if(e.getSource()==applet){
                    jav=0;app=1;apl=0;}
                    else if(e.getSource()==aplikacija){
                    jav=0;app=0;apl=1;}
                    repaint();
          }

public void paint(Graphics g) {
          if(jav==1)                    g.drawString("JAVA",185, 50 );
          else if (app==1) g.drawString("APLET",185, 50 );
          else if (apl==1)  g.drawString("APLIKACIJA",185, 50 );
          }
          }
```



12. Napisati applet koji iscrtava neispunjen pravougaonik sa koordintama - (360,130,140,180).

-iscrtava string sa tekstom "GRAFICKI ELEMENTI"-koordinate (325, 50) i postavlja oko njega zaobljen pravougaonik (320,30,140,40,12,12).

-iscrtava neispunjen kvadrat - (60,60,150,150), i ispunjen krug K1 - (60,60,150,150).

-iscrtava polygon kroz temena(*postaviti plavu boju za polygon*):

P_1 =(100, 20) P_2 =(300, 20) P_3 =(350, 170)
P_4 =(300, 300)P_5 =(100, 300) P_6 =(50, 170)

E)Uvesti novu promenljivu "Dimenzija_poligona", i dodeliti funkcionalnost na tastere i klikom na njih iscrtava poligon sa:

4-temena **5-temena** **6-temena**

0-briše krug K1 *("clearRect")*

F)Ako se misem klikne u domen kruga K1 ,on se transformise u String "PLAY"(160,160),crvene boje,

**postaviti uslov za domen* → *if(x>60 && x<150 && y>100 && y<220)*

G)Formirati tri butona I klikom na jedan od tri butona pomera se polygon za odgovarajucu poziciju:

P_190- g.translate(50,190), P_240 – g.translate(50,240), P_290 – g.translate(50,290)

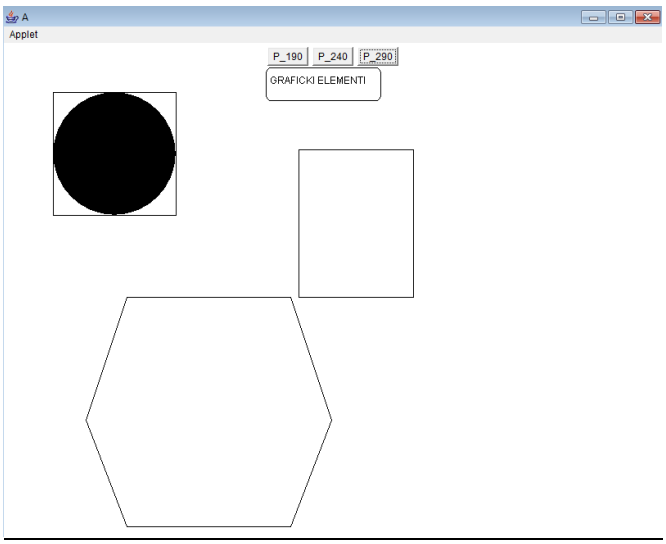
REŠENJE:

```
import java.awt.*;
import java.applet.*;
import java.awt.event.*;

public class Prvi extends Applet implements ActionListener,
MouseListener,KeyListener{
    Button JKT=new Button("P_190");
    Button JST=new Button("P_240");
    Button RST=new Button("P_290");
        int Dimenzija_poligona=6;
        int aktivno=1;

    int j_kt=0;
    int j_st=0;
    int r_st=1;
```

```
        public void mouseExited(MouseEvent e){}
        public void mouseEntered(MouseEvent e){}
        public void mouseReleased(MouseEvent e){}
        public void mousePressed(MouseEvent e){
          int x=e.getX();
          int y=e.getY();
          if(x>60 && x<150 && y>100 && y<220)
          if(aktivno==1) aktivno=0;
          else aktivno=1;
          repaint();}
        public void mouseClicked(MouseEvent e){}
        public void keyReleased(KeyEvent e){}
        public void keyTyped(KeyEvent e){}
        public void keyPressed(KeyEvent e){
switch (e.getKeyChar()){
        case '4':Dimenzija_poligona=4;break;
        case '5':Dimenzija_poligona=5;break;
        case '6':Dimenzija_poligona=6;break;
        case '0':Dimenzija_poligona=0;break;}
                        repaint();
}

public void init() {
                JKT.addActionListener(this);
                JST.addActionListener(this);
                RST.addActionListener(this);
                add(JKT);add(JST);add(RST);
                this.addMouseListener(this);
                this.addKeyListener(this);}

public void paint(Graphics g) {
                g.drawRect(360,130,140,180);
                g.drawString("GRAFICKI ELEMENTI",325, 50 );
                g.drawRoundRect(320,30,140,40,12,12);
                g.drawRect(60,60,150,150);
                g.fillOval(60,60,150,150);
            if(aktivno==0)
                {g.setColor(Color.red);
                g.drawString("PLAY",160,160);}
                else      g.fillOval(60,60,150,150);
                int P1[] = {100,300,350,300,100,50};
```

```
int P2[] = {20,20,170,300,300,170};

if(j_kt==1)              g.translate(50,190);
else if(j_st==1)         g.translate(50,240);
else if(r_st==1)         g.translate(50,290);

if(Dimenzija_poligona==4)
g.drawPolygon(P1,P2,4);
else if(Dimenzija_poligona==5)
g.drawPolygon(P1,P2,5);
else if(Dimenzija_poligona==6)
g.drawPolygon(P1,P2,6);
else if(Dimenzija_poligona==0)
g.clearRect(60,60,150,150);}
```

```
public void actionPerformed(ActionEvent e){
        if(e.getSource()==JKT)
        {j_kt=1;j_st=0;r_st=0;}
        else if(e.getSource()==JST)
        {j_kt=0;j_st=1;r_st=0;}
        else if(e.getSource()==RST){
        j_kt=0;j_st=0;r_st=1;}
        repaint();}
        }
```



13.Napisati applet koji kreira Glavni panel sa glavnom mrežom (4 kolone,2 reda) i postavlja:

A) PRVI RED:

A.1) Tekst labela "PROGRAMSKI JEZICI" i oko njega ispunjen krug crne boje (0,0,150,150).

A.2) Panel_podaci sa mrežnom postavkom Mreza_podaci(3 red, 2 kolone) i u njima :

Labela ime,TextField ime,Labela prezime,TextField prezime, Labela sistem,TextField sistem

**tekst polje sIstem sadrži unet podatak "administrator" i onemogućen je unos*

A.3) Polje Lozinka za unos šifre(10 karaktera)(*zasticeno)
A.4) Postaviti dugme za prijavu.

B) DRUGI RED:

B.1) Panel_pitanje sa mrežnom postavkom Mreza_pitanje(3 reda,1 kolona) i u prvom redu pitanje , *"Apleti su posebna vrsta Java programa koji su namenjeni za ugrađivanje u:",* a zatim postaviti dva dugmeta za odgovor:

-HTML stranice
-PASCAL kodove

Dodeliti funkcionalnost dugmadima,tako da klikom na jedan od njih ispisuje se string TAČAN/NETAČAN odgovor, pozicija stringa(185, 150).

B.2) Panel_check sa Border postavkom, i redom dodeliti check butone (java,c,c++,pascal), a u centralnom delu border postavke postaviti polje Komentar(10,10).

B.3) Panel_radio sa mrežnom postavkom Mreza_radio (1 red,2 kolone) I postaviti grupisane radio buttone redom (web dizajn,e-poslovanje).

B.4) Panel_dugmad sa mrežnom postvkom Mreza_dugmad(1 red,2 kolone) i redom dodati dugmad(Nazad,Napred).

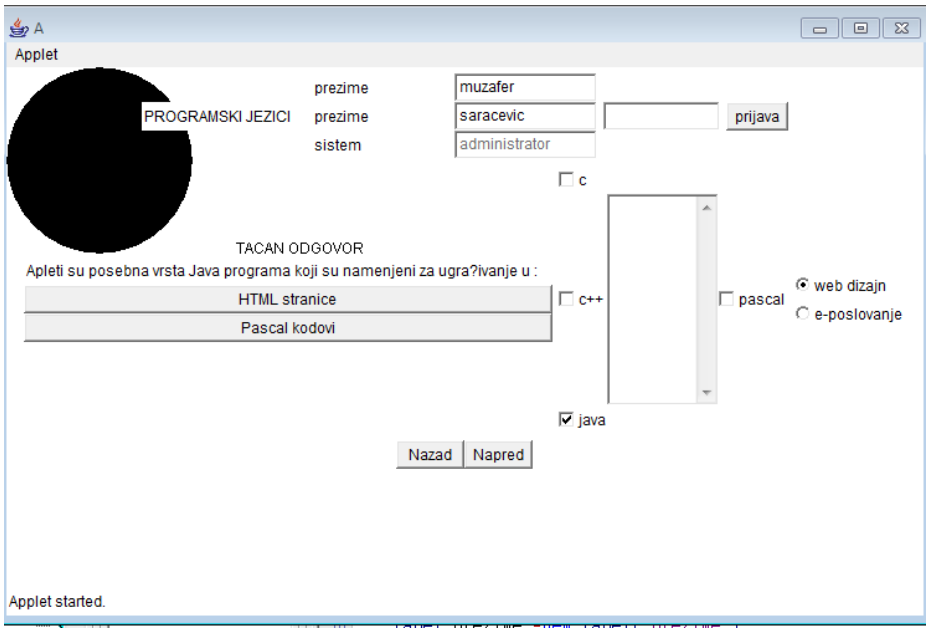
REŠENJE:

```
import java.awt.*;
import java.applet.*;
```

```
import java.awt.event.*;
import javax.swing.*;

public class Prvi extends Applet implements ActionListener{
        GridLayout glavna=new GridLayout(4,2);
        Panel p=new Panel();
        Label Naslovna =new Label("PROGRAMSKI JEZICI")
        Checkbox prvi=new Checkbox("java",true);
        Checkbox drugi=new Checkbox("c", false);
        Checkbox treci=new Checkbox("c++", false);
        Checkbox cetvrti=new Checkbox("pascal", false);

        Button nazad=new Button("Nazad");
        Button napred=new Button("Napred");
        TextArea Komentar = new TextArea(10,10);
        Panel panel_podaci=new Panel();

        GridLayout mreza_podaci=new GridLayout(3,2);
        Panel panel_check=new Panel();
        BorderLayout BL=new BorderLayout();

        Panel panel_pitanje=new Panel();
        GridLayout mreza_pitanje=new GridLayout(3,1);

        Panel panel_radio=new Panel();
        GridLayout mreza_radio=new GridLayout(2,1);

        Panel panel_dugmad=new Panel();
        GridLayout mreza_dugmad=new GridLayout(1,2);
        Button prijava=new Button("prijava");
        Label ime=new Label("prezime");
        TextField ime_polje = new TextField();

        Label prezime =new Label("prezime");
        TextField prezime_polje=new TextField();

        Label Sistem =new Label("sistem");
        TextField sistem_polje=new TextField("administrator");

        TextField password = new TextField(10);
```

```
        Label Pitanje =new Label("Apleti su posebna vrsta Java programa koji
su namenjeni za ugradjivanje u :");

        Button odg1=new Button("HTML stranice");
        Button odg2=new Button("Pascal kodovi");
        CheckboxGroup grupa=new CheckboxGroup();
        Checkbox rb1=new Checkbox("web dizajn",grupa,true);
        Checkbox rb2=new Checkbox("e-poslovanje",grupa,false);
        int r_b1=1;
        int r_b2=0;

public void init() {
                p.setLayout(glavna);
                add(Naslovna);add(panel_podaci);
                add(password);add(prijava);
                add(panel_pitanje);add(panel_check);
                add(panel_radio);add(panel_dugmad);

                panel_podaci.setLayout(mreza_podaci);
                panel_podaci.add(ime);
                panel_podaci.add(ime_polje);
                panel_podaci.add(prezime);
                panel_podaci.add(prezime_polje);
                panel_podaci.add(Sistem);
                panel_podaci.add(sistem_polje);
                password.setEchoChar('*');
                sistem_polje.setEnabled(false);
                odg1.addActionListener(this);
                odg2.addActionListener(this);

                panel_check.setLayout(BL);
                panel_check.add(prvi,BorderLayout.SOUTH);
                panel_check.add(drugi,BorderLayout.NORTH);
                panel_check.add(treci,BorderLayout.WEST);
                panel_check.add(cetvrti,BorderLayout.EAST);
                panel_check.add(Komentar,BorderLayout.CENTER);

                panel_radio.setLayout(mreza_radio);
                panel_radio.add(rb1);panel_radio.add(rb2);
                panel_pitanje.setLayout(mreza_pitanje);
                panel_pitanje.add(Pitanje);
```

```
                panel_pitanje.add(odg1);panel_pitanje.add(odg2);

                panel_dugmad.setLayout(mreza_dugmad);
                panel_dugmad.add(nazad);
                panel_dugmad.add(napred);}

public void actionPerformed(ActionEvent e){
        if(e.getSource()==rb1)
        {r_b1=1;r_b2=0;}
        else
        if(e.getSource()==rb2){
        r_b1=0;r_b2=1;}
        repaint();
}

public void paint(Graphics g) {
if(r_b1==1)
 g.drawString("TACAN ODGOVOR",185, 150 );
else if (r_b2==1)
g.drawString("NETACAN ODGOVOR",185, 150 );
g.fillOval(0,0,150,150);
}
}
```



14. Aplet koji ilustruje kompas sa 4 funkcionalna dugmića sa iscrtavanjem pravougaonika u centralnom delu BorderLayout mreže.

UPUTSTVO:
-Kreirati četiri butona: *sever,jug,istok,zapad.*
-postaviti mrežnu postavku BorderLayout.
-kreirati jedan ispunjen krug sa podacima (200,100,100,100),
-kreirati jedan pravougaonik čija ce pozicija varirati u zavisnosti od klika na jedan od butona *(npr.klik na sever,pravougaonik je okrenut ka severu itd.):*

(istok)- (250,150,50,10)
(zapad)- (200,150,50,10)
(jug)- (250,150,10,50)
(sever)- (250,100,10,50).

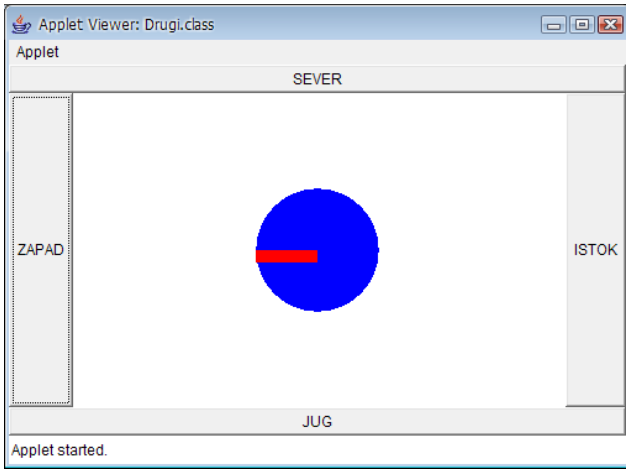
-Pomoću **ActionPerformed (ActionEvent e)**, dodeliti funkcionalnost butonima, tako da klikom na jedan od 4 butona postavlja pravougaonik na zadatu poziciju.

REŠENJE:

```
import java.awt.*;
import java.applet.*;
import java.awt.event.*;

public class Kompas extends Applet implements ActionListener{

//Formiramo dugmice strana
        Button s=new Button("SEVER");
        Button j=new Button("JUG");
        Button i=new Button("ISTOK");
        Button z=new Button("ZAPAD");

        //Formiramo polarni izgled
        BorderLayout bl=new BorderLayout();

        //Indikatori za strane sveta
        int sever=1;
        int jug=0;
        int istok=0;
        int zapad=0;
```

```
public void init() {
        //Postavljamo akcije dugmicima
        s.addActionListener(this);
        j.addActionListener(this);
        i.addActionListener(this);
        z.addActionListener(this);

        //Postavljamo izgled i dugmice
        setLayout(bl);
        add(s,BorderLayout.NORTH);
        add(j,BorderLayout.SOUTH);
        add(i,BorderLayout.EAST);
        add(z,BorderLayout.WEST);
        }

public void paint(Graphics g) {
        //Crtamo kompas
        g.setColor(Color.blue);
        g.fillOval(200,100,100,100);
        g.setColor(Color.red);
        //Strelica za sever
        if(istok==1)
        g.fillRect(250,150,50,10);
        else if(zapad==1)
        g.fillRect(200,150,50,10);
        else if(jug==1)
        g.fillRect(250,150,10,50);
        else if(sever==1)
        g.fillRect(250,100,10,50);
}

public void actionPerformed(ActionEvent e){
if(e.getSource()==s)
{
sever=1;
jug=0;
istok=0;
zapad=0;
else if(e.getSource()==j){
sever=0;
jug=1;
```

```
istok=0;
zapad=0;          }
else if(e.getSource()==i){
sever=0;
jug=0;
istok=1;
zapad=0;          }
else if(e.getSource()==z){
sever=0;
jug=0;
istok=0;
zapad=1;          }
repaint();
} }
```



15. Napisati applet koji kreira formu sa border postavkom: na severu postavlja - panel1 koji ima 3 kolone jedan red i u svakoj po jedno dugme.

Bold,
Italic,
brisi

u centralno polje postavljamo: panel2 sa 2 reda jednom kolonom.

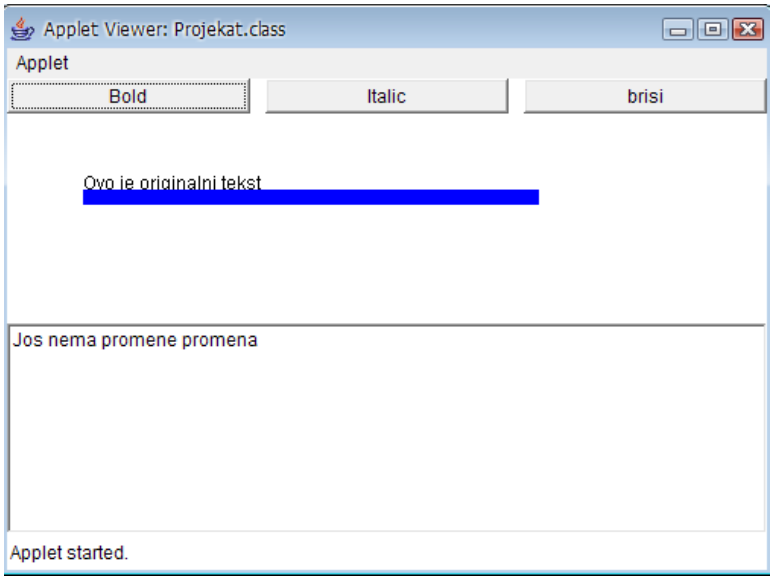
U prvom redu je deo koji ispisuje tekst po vašoj želji i podvlači ga pravougaonikom plave boje.

U drugom redu je tekstualno polje koje ispisuje koje je dugme pritisnuto:

Postaviti akcije za dugmiće:

1) bold i italic menjaju stil ispisivanja teksta
2) brisi , briše tekst

Sva tri dugmeta menjaju tekst u tektualnom polju.
Početni tekst je: "Još nema promene" a pritiskom nekog od ovih dugmića ispisuje se koje je dugme pritisnuto npr."Pritisnuto je dugme bold".



REŠENJE:

```
import java.awt.*;
import java.awt.event.*;
import java.applet.*;
public class Projekat extends Applet implements ActionListener{

        Panel panel1,panel2;
        Button bold,italic, brisi;
        TextPanel t_panel;
        int fBold = 0;
        int fItalic = 0;
        int fSize=12;

        String tekst="Ovo je originalni tekst";
        TextField tf;

public void init()  {
                setFont(new Font("Arial",fBold+fItalic,fSize));
```

```
                    setLayout(new BorderLayout());

panel1 = new Panel();
panel1.setLayout(new GridLayout(1,3,10,5));
bold = new Button("Bold");
bold.addActionListener(this);
panel1.add(bold);

italic = new Button("Italic");
italic.addActionListener(this);
panel1.add(italic);

brisi = new Button("brisi");
brisi.addActionListener(this);
panel1.add(brisi);

add(panel1,BorderLayout.NORTH);

panel2 = new Panel();
                    panel2.setLayout(new GridLayout(2,1));
                    t_panel = new TextPanel();

tf=new TextField("Jos nema promene promena",20);
                    panel2.add(t_panel);
                    panel2.add(tf);
                    add(panel2,BorderLayout.CENTER);
          }
public void actionPerformed(ActionEvent e) {
          String tekst2="Pritisnuto je dugme ";
          String butLabel = e.getActionCommand();
                    if(butLabel.equals("Bold")){
                              if(fBold == 1)
                              fBold = 0;
                              else fBold = 1;
                              tekst2+="bold";}
                    if(butLabel.equals("Italic")){
                              if(fItalic == 2)
                              fItalic = 0;
                              else fItalic = 2;
                              tekst2+="italic";}
                    if(butLabel.equals("brisi")){
```

```
                    tekst=" ";
                    tekst2+="brisi";}
                tf.setText(tekst2);
                t_panel.rewrite();
        }

class TextPanel extends Canvas
  {
  String message;
  public TextPanel()
    {
   rewrite();
        }
        public void rewrite()
    {
        setFont(new Font("Arial",fBold+fItalic,fSize));
        message =tekst;
    repaint();
    }

  public void paint(Graphics g)
    {
    g.drawString(message, 50, 50);
    g.setColor(Color.blue);
    g.fillRect(50,50,300,10);
    }
 }
}
```

Dat je uporedni prikaz Java apleta i Java aplikacija.

JAVA APLET	*JAVA APLIKACIJA*

import java.awt.*;
public class ButtonApplet extends Applet {
 	
Button abort = new Button("Abort");
Button retry = new Button("Retry");
Button fail = new Button("Fail");
 	
public void init() {
add(abort);
add(retry);
add(fail);
 	
}
}
	import javax.swing.*;
	public class Buttons extends JFrame {
	JButton abort = new JButton("Abort");
	JButton retry = new JButton("Retry");
	JButton fail = new JButton("Fail");
	
	public Buttons() {
	super("Buttons");
	setSize(80, 140);
	
	JPanel pane = new JPanel();
	pane.add(abort);
	pane.add(retry);
	pane.add(fail);
	setContentPane(pane);}
	public static void main(String[] arguments) {
	Buttons rb = new Buttons();
	rb.show(); }
	}

KONTROLNA PITANJA

1.Definisati pojam Apleta i objasniti kako se on poziva?
2.Šta radi metoda paint() i kako se definiše?
4.Koja je razlika između apleta i aplikacije?
5.Kako se definiše izvršna metoda kod apleta?
6.Šta radi metoda ActionPerformed?
7.Kako se definiše aplet, koju klasu naseđuje?
8.Šta radi metoda init()?
9.Kako se definišu sledeće komponente na panel (dugme,tekst polje,labela)?
10.Koja je naredba za ispunjen krug i zaobljen provougaonik?
11. Kakva je mreža BorderLayout, a kakva GridLayout?
12. Koji se događaji koriste za dodavanje funkcionalnosti na klik miša?
13.Koji se događaji koriste za dodavanje funkcionalnosti na tastere?
14.Na koji način se može definisati poligon?
15.Šta radi sledeća naredba import java.awt.event.* ?

ZADACI – primer testa br.1

1.Napisati aplet na kojem se iscrtavaju sledeći geometrijski elementi:

- Najpre postaviti boju na plavu ,
- Jedan zaobljen pravougao (180,30,50,40,12,12)
- Jedan ispunjenu elipsu (130,150,70,120)
- Jedan poligon sa 5 temena, pozicije A(100,20) , B(300,20), C(350,170), D(300,300), E(100,300)
- String sa vašim imenom, pozicije 190, 50.

2. Napisati aplet na kojem se postavljaju sledeće komponente: Postaviti mrežu 4 X 3. Redom postavljati

Labela - dobrodošli	Labela - username	Polje-username
Labela - aplet	Labela- password	Polje- password *polje zaštićeno*
Radiobutton-DA	Dugme-NAZAD	Polje za komentar
Radiobutton-NE	Dugme-NAPRED	**Dugme- prijava**

3. Dat je izvorni kod u Javi.Pronadjite nepravilnosti i ispravite.

```
import java.awt.*;
import java.applet.*;
public class Drugi {
        public void int(Graphics g) {
                public static void main(){
                g.fillOval(100,100,70,70);
                g.setColor();
                g.drawString(geometrija);
                g.drawOval(120,120,150,150);
          int x={100,300,350,300,100,50};
        y[]={20,20,170,300,300,170};
          Polygon poly=new (x,y,6);
          g.drawPolygon(poly);
        }
}
```

4. Napisati aplet na kojem će se nalaziti jedan funkcionalni **Button i jedan pravougaonik.**
   Dodeliti mu funkcionalnost tako da klikom na njega na panelu se ispisuje *„pritisnuli ste dugme*“.
   Takodje dodeliti funkcionalnost na tastere tako da ako se pritisne na

odgovarajuci taster pravougaonik će menjati boju (Z-zeleno, C-crveno, S-sivo, R- crno).

5. Konverotvati sledeći aplet u aplikaciju.

```
import java.awt.*;
import java.applet.*;

public class MojAplet extends Applet {

    Button dugme1=new Button("Dugme");
   Label ime_l=new Label("ime");
    TextField ime = new TextField();
    Label sifra_l =new Label("sifra");
    TextField sifra=new TextField();

    public void init() {
    add(sifra);
    add(ime_l);
    add(ime);
    add(sifra_l);
     add(dugme1);
    }
}
```

ZADACI – primer testa br.2

1.Napisati aplet na kojem se iscrtavaju sledeći geometrijski elementi:

- Jedan pravougaonik (180,30,50,40)
- Postaviti boju na zelenu ,
- String sa vašim imenom, pozicije 210, 70.
- Jedan ispunjen krug (150,180,120,120)
- Jedan trougao, A(100,20) , B(300,20), C(350,170)

2. Napisati aplet na kojem se postavljaju sledeće komponente: Postaviti mrežu 6 X 2. Redom postavljati

Labela - **dobrodošli**	Labela - username
Polje za komentar	Polje-username
CheckButton-DA	Labela- password
CheckButton-NE	Polje- password *polje zaštićeno*
Dugme-BACK	Labela - prijava
Dugme-NEXT	**Dugme- OK**

5. Dat je izvorni kod u Javi.Pronadjite nepravilnosti i ispravite.

```
import java.awt.*;

public class Dugme1  {
Button crveno= new Button("CRVENO");
Button plavo= new Button("PLAVO");
Label L=new Label (“string”);
Button zuto= new ("ZUTO");
 gl= new GridLayout(3,1,10,20);
 public void paint() {
                    crveno.setl
                    setLayout();
                    add(zuto.setColor);
                    add(plavo);
                    add(crveno);
public ActionPerformed(){}

}
```

6. Napisati aplet na kojem će se nalaziti jedan funkcionalni **Button i jedan ispunjen krug.**
Dodeliti mu funkcionalnost tako da klikom na njega na panelu se ispisuje „*pritisnuli ste dugme*“.
Takodje dodeliti funkcionalnost na tastere tako da ako se pritisne na odgovarajuci taster krug će menjati boju (Z-zeleno, C-crveno, S-sivo, R-crno).

5. Konverotvati sledeći aplet u aplikaciju.

```
import java.awt.*;
import java.applet.*;

public class MojAplet extends Applet {

   Label username=new Label("ime");
    TextField ime = new TextField();
    Label opis =new Label("opis");
    TextArea komentar=new TextArea ("opisi postupak");
   Button dugme1=new Button("Dugme");

    public void init() {

            add(komentar);
            add(username);
            add(ime);
            add(opis);
      add(dugme1);

    }
}
```

ZADACI – primer testa br.3

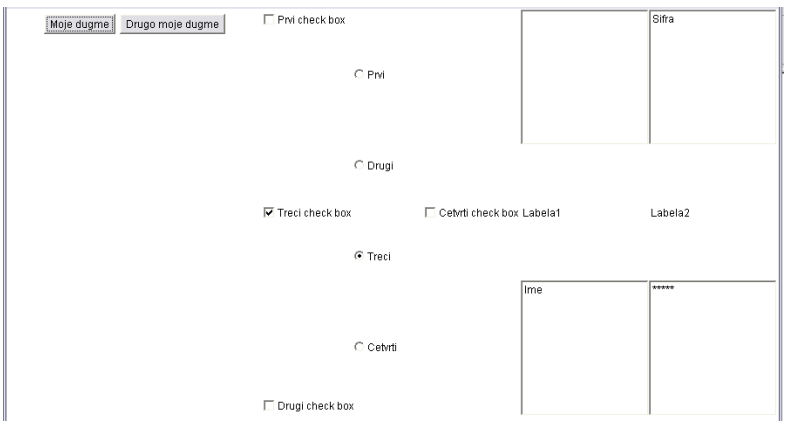
1.Napisati aplet za rad sa grafickim komponentama koji:
Kreira 4 panela:
Panel dugmici,Panel radio,Panel tekst,Panel p1

Zatim kreira 4 mreze:
GridLayout gl=new GridLayout(3,2);
GridLayout gl1=new GridLayout(4,1);
GridLayout glavna=new GridLayout(1,3);

BorderLayout bl=new BorderLayout();
Kreira sledece butone:
Buton d1,d2.
Zatim 4 radio butona I 4 checkboxa.

Kreira 4 tekst polja (t1,t2,t3,t4) **I dve tekst labele**(l1,l2).

**za polje t1 onemoguciti unos,a za polje t4 postaviti skrivene karaktere.*
**zatim prebaciti tekst iz polja t4 u polje t2*
**postaviti tekst labeli l1 "labela 1"*
**aktivirati treci check box*



Napisati applet koji kreira form sa mreznom sa 4 redova i 3 kolone postavkom i postavlja :
labele: ime,prezime, sifra u prvom redu, zatim;

polja za unos imena ,prezimena i sifre(zasticeno) u drugom redu;
panel1,panel2,panel3 u trecem redu
(postaviti border postavku za panel2 I u njemu cetiri checkbox-a
S,N,W.E koja su grupisana) dumice levo ,centar,desno u cetvrtom redu.

Napisati applet koji crta medu od sledecih kruzica:

1) usi: neispunjeni krugovi poluprecnika 10 sa centrom O_1(20,20) i O_2(100,20),Boja:Siva

2)glava: ispunjen krug koji je upisan u kvadrat sa gornjim levim uglom (20,20) i stranicom 80,Boja:Siva

3)telo: a) mrsavo elipsa upisana u pravougaonik (20,100) stranica 80,120
b)debelo (0,100) 120,120,Boja:Siva

4)Oci :Krugovi sa centrima(50,50) i (70,50) poluprecnika 5

Boju ociju definisati kao novu boju c i menjati tipki B- “plave oci ”,G-”zelene oci”,Y-“zute oci”.Ako se misem kliskne u pravougaonik u koji je upisano debelo telo telo se menja(ako je bilo debelo postaje mrsavo i obrnuto)
(podrazumevane vrednosti su plavoki buca)

Napisati applet koji:

- postavlja boju na crnu
- iscrtava string “java” od kordinate (185,75) i
- postavlja oko njega zaobljen pravougaonik
- iscrtava ne ispunjen kvadrat plave boje od koordinate (80,80) stranice a=150
- iscrtava dva ispunjena kruga
  - (100,100), r=70,plav
  - (50,50) , r=50,beo
- iscrtava polygon kroz temena:
- A_1=(100, 20) A_2=(300, 20) A_3=(350, 170)
- A_4=(300, 300) A_5 =(100, 300)

3 DEO

GRAFIČKI

KORISNIČKI INTERFEJS

3 GRAFIČKI KORISNIČKI INTERFEJS

Swing, koji je deo Java Foundation Class bibliteke, jeste ekstenzija Abstract Windowing Toolkit koji je integrisan u Java 1.2.

Swing pruža veću funkcionalnost nove komponente, proširene mogućnosti starih komponenti, bolje upravljanje događajima. Svi elementi Swing-a su deo paketa java.awt.swing. Da bi se koristile Swing klase, mora se koristiti iskaz import sa pozivom željenih klasa ili uključiti sve klase u iskaz kao u primeru:

import java.awt.swing.*;

Swing komponente su podklase klase **JComponent.**

Pri kreiranju jednostavne Swing aplikacije, prvi koraci su kreiranje podklase klase JFrame. Klasa JFrame je proširenje klase Frame.

3.1 Rad sa grafčkim elementima i događajima

Java može da se koristi za razvoj raznovrsnih aplikacija. Postoje jednostavni tekstualno-zasnovani programi koji se nazivaju konzolnim aplikacijama. Takvi programi podržavaju samo tekstualni unos i ispis na monitoru vašeg računara. Takođe, možete da pravite aplikacije sa grafičkim korisničkim interfejsom (engl. Graphical User Interface – GUI). Ove aplikacije raspolažu sa menijima, paletama alatki, dugmadima, trakama za pomeranje sadržaja drugim kontrolama koje reaguju na miša.

1. Napisati program koji prikazuje panel i izborom prve dve tačke crta pravu i omogućiti da klikom na dugme 'o' se iscrtava krug od prve dve tačke.

- zatim svakim sledećim izborom tačke crta trougao sacinjen od prve dve I zadnje tacke

- na kraju izborom cetvrte tacke crta pravougaonik.

Dati tipkama sledeće uloge:

P-popunjava figuru ako nije popunja i iscrtava samo okvir ako je popunjena

U-omogucava povratak prethodnog dogadjaja

K-pretvara pravouganik u krivu III-reda

ispisuje da li se nalazimo u touglu ili ne (tekstualni ispis)

prilikom prelaska preko figure, figura postaje popunjena

postaviti gradijent -boja1 na ‚R'(red) -boja2 na ‚B'(blue)

Dugme '+' *povećava veličinu kojom se icrtava linija za 5

Dugme '-' *smanjuje veličinu kojom se icrtava linija na 5

2-pomera kordinate prve tačke pravougaonika za -5 po y-osi
8-pomera kordinate prve tačke pravougaonika za +5 po y-osi
4-pomera kordinate prve tačke pravougaonika za -5 po x-osi
6-pomera kordinate prve tačke pravougaonika za +5 po x-osi

REŠENJE:

```
import java.awt.*;
import java.awt.event.*;
import javax.swing.*;
import java.awt.geom.*;

public class Simple extends JPanel implements MouseListener, KeyListener {

    static JFrame frame;
    int korak=1;
    int x1, y1; //kooordinate pocetne tacke
    int x2, y2; // koordinate krajne tacke
    int xp, yp;// koordonate tacke p-gornjef levog temena
    Boolean popuni=false;
    Boolean krug=false;
    Boolean okvir=false;
    Boolean kriva=false;
    int debljina=5;

        Color boja2= Color.black;
        Color boja1= Color.black;
```

```
        public void  mouseEntered (MouseEvent e){}
        public void  mouseExited (MouseEvent e){}
        public void  mouseReleased (MouseEvent e){}
        public void  mousePressed (MouseEvent e){}
        public void  mouseClicked (MouseEvent e){
        if(korak==1){
                x1=e.getX();
                y1=e.getY();

                korak=2;
                repaint();
                  }
      else
      if(korak>=2){
        x2=e.getX();
        y2=e.getY();
        korak=3;
        repaint();
        }
                }

  public Simple () {
}
        public void keyPressed (KeyEvent e) {}
        public void keyReleased (KeyEvent e) {}
        public void keyTyped (KeyEvent e) {
                switch (e.getKeyChar()){
                case('p'):
                case ('P'): popuni=!popuni;
                break;
                case('r'):
                case ('R'): korak=1;//DA BRISE SVE SA EKRANA
                break;
                case('k'):
                case('K'): krug=!krug;
                break;
                case('o'):
                case('O'): okvir=!okvir;
                break;
                case('a'):
                case('A'): boja1=Color.white;
```

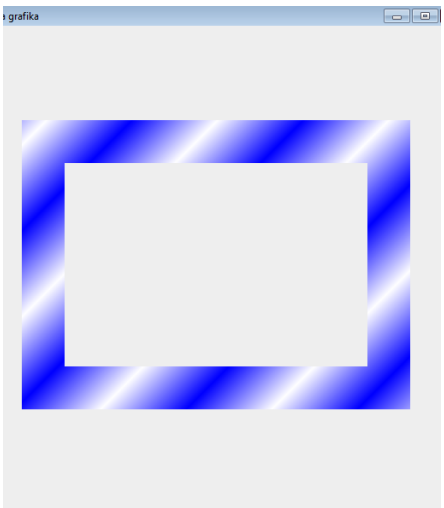
```
                    break;
                    case('b'):
                    case('B'): boja1=Color.gray;
                    break;
                    case('c'):
                    case('C'): boja1=Color.black;
                    break;
                    case('x'):
                    case('X'): boja2=Color.yellow;
                    break;
                    case('y'):
                    case('Y'): boja2=Color.green;
                    break;
                    case('Z'):
                    case('z'): boja2=Color.blue;
                    break;
                    case('1'): debljina+=5;
                    break;
                    case('3'):  if(debljina>5)
                                debljina-=5;
                    break;
                    case('2'): y1-=5;
                               y2-=5;
                    break;
                    case('8'): y1+=5;
                               y2+=5;
                    break;
                    case('4'): x1-=5;
                               x2-=5;
                    break;
                    case('6'): x1+=5;
                    x2+=5;
                    break;

                    case('T'):
                    case('t'): kriva=!kriva;
                    break;
                    }
                    repaint();
}
```

```
public void paint (Graphics g) {
                    super.paint(g);
Graphics2D g2 = (Graphics2D) g;
GradientPaint gr=new GradientPaint(0,0,boja1,50, 50,boja2,true);
                    g2.setPaint(gr);
BasicStroke bs=new BasicStroke(debljina,
BasicStroke.CAP_ROUND,BasicStroke.JOIN_MITER);
          g2.setStroke(bs);
                    if(korak==3){
                    xp=(x1<x2)?x1:x2;
                    yp=(y1<y2)?y1:y2;
           if(!krug){
Rectangle2D rect=new Rectangle2D.Float(xp,yp,Math.abs(x2-x1), Math.abs(y2-y1));
          g2.draw(rect);
          if(popuni)
           g2.fill(rect);
            if(okvir){
            g2.setColor(Color.black);
           g2.draw(rect);}
          }
          else
          if(kriva){
           Point2D p1=new Point2D.Float(x1,y1);
           Point2D p2=new Point2D.Float(x2,y2);
           Point2D c1=new Point2D.Float(x2,y1);
           Point2D c2=new Point2D.Float(x1,y2);
           CubicCurve2D K3=new CubicCurve2D.Float();
           K3.setCurve(p1,c1,c2,p2);
           g2.draw(K3);
           }
          else {
Ellipse2D l=new Ellipse2D.Float(xp,yp,Math.abs(x2-x1), Math.abs(y2-y1));
          g2.draw(l);
          if(popuni)
           g2.fill(l);
            if(okvir){
            g2.setColor(Color.black);
           g2.draw(l);}
}
           }
 }
```

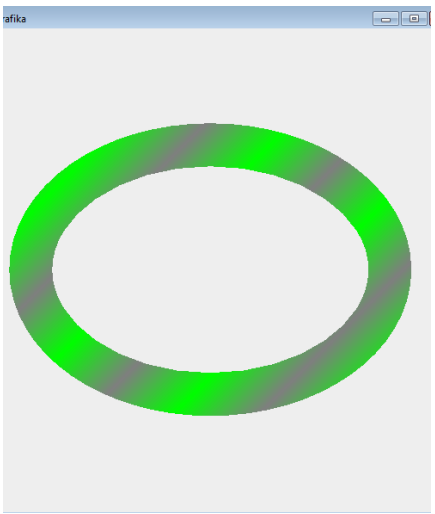
```
public static void main(String s[]) {
Simple panel = new Simple();
    // da je panel osetljiv na dogadjaj misa
    panel.addMouseListener(panel);
    frame = new JFrame("Ova se kafa ohladi");
    frame.addWindowListener(new WindowAdapter() {
public void windowClosing(WindowEvent e) {System.exit(0);}
    });
    frame.getContentPane().add("Center", panel);
     frame.addKeyListener(panel);
   frame.setSize(new Dimension(600,600));
    frame.setVisible(true);
    }
}
```

Pri radu sa JFrame objektima postoke razlike u odnosu na na rad sa njegovom kopijom AWT. Umesto da dodajete kontejnere i komponente direktno u okvir, one se moraju dodati na prelazni kontejner koji se naziva pano sadržaja-content pane.
Objekat JFrame je podeljen na nekoliko različitih panoa. Glavni pano sa kojim se radi je content pane, koji predstavlja kompletnu oblast okvira u koji će se komponente dodavati.

Rezultat br.1



Rezultat br.2



Postupak dodavanja komponente u *content pane*:

- kreirati JPanel objekat (Swing verzija panela).
- dodati sve komponente (kontejnere) u JPanel upotrebom metoda add(Component).
- napraviti od panoa JPanel content pane upotrebom metoda setContentPane(Container).
- JPanel objekat je jedini argument.

2. Napisati program koji prikazuje panel i izborom dve tačke crta krug ciji je prečnik data duž (prvo teme je fiksno) .

Dati tipkama sledeće uloge:
P-popunjava krug kada u se kursor nalazi u njemu,
U-omogućava da se vratimo jedan korak u nazad,
K-crta pravougaonik kome je data duz dijagonala,
w,W-omugućava clipping area mehanizam za oblik zone uzima zadnju nacrtanu figuru.

A-postavlja belu boju za prvu boju GradientPaint metode
B- postavlja sivu za prvu boju GradientPaint metode
C- postavlja crnu za prvu boju GradientPaint metode

X- postavlja žutu za drugu boju GradientPaint metode
Y- postavlja zelena za drugu boju GradientPaint metode
Z- postavlja plava za drugu boju GradientPaint metode

1-povećava veličinu kojom se icrtava linija za 5
3-smanjuje veličinu kojom se icrtava linija na 5

2-pomera kordinate prve tačke pravougaonika za -5 po y-osi
8-pomera kordinate prve tačke pravougaonika za +5 po y-osi
4-pomera kordinate prve tačke pravougaonika za -5 po x-osi
6-pomera kordinate prve tačke pravougaonika za +5 po x-osi

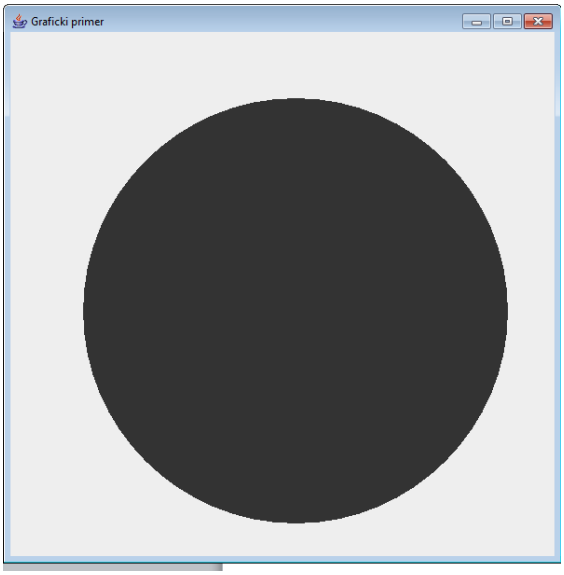
REŠENJE:

```
import java.awt.*;
import java.awt.event.*;
import javax.swing.*;
import java.awt.geom.*;
```

```
public class Simple extends JPanel implements
MouseListener,MouseMotionListener, KeyListener {

    static JFrame frame;
    int korak=1;
    int x1,y1;
    int x2,y2;
    int xp,yp;
    double d,r;
    double dp, rp;

    Boolean popunjena=false;

 public Simple () {
    }

public void mouseMoved(MouseEvent e){
    Ellipse2D el=new Ellipse2D.Double((x1+x2)/2-r, (y1+y2)/2-r,d,d);
//crtanje elipse
    Point2D point=new Point2D.Float(e.getX(),e.getY());
//za crtanje tacke

if (el.contains(point)){
            popunjena=true;
            }
            else
            popunjena=false;
    repaint();
    }

public void mouseDragged(MouseEvent e){}
 public void mouseEntered(MouseEvent e){}
 public void  mouseExited (MouseEvent e){}
 public void  mouseReleased (MouseEvent e){}
  public void  mousePressed (MouseEvent e){}

public void  mouseClicked (MouseEvent e){
            if(korak==1) {
                        x1=e.getX();
                        y1=e.getY();
```

```
                korak=2;
                }
                else
                if(korak>=2){
                        if(korak==3){
                                xp=x2;
                                yp=y2;
                                rp=r;
                                dp=d;}
                                else
                                {xp=e.getX();
                                yp=e.getY();}

                        x2=e.getX();
                        y2=e.getY();
                        d=Math.sqrt((x1-x2)*(x1-x2)+(y1-y2)*(y1-y2));
                        r=d/2;
                        korak=3;
                        }
                        repaint();
        }

public void keyPressed (KeyEvent e) {}
        public void keyReleased (KeyEvent e) {}

public void keyTyped (KeyEvent e) {
                switch (e.getKeyChar()){
                        case ('U'):
                        case ('u'): x2=xp; y2=yp; r=rp; d=dp;
                        break;
                        }
                }

public void paint (Graphics g) {
                super.paint(g);
                Graphics2D g2 = (Graphics2D) g;
    if(korak==3)
    {
        Ellipse2D e=new Ellipse2D.Double((x1+x2)/2-r, (y1+y2)/2-r,d,d);
        g2.draw(e);
        if(popunjena)
```

```
            g2.fill(e);
            }
    }

public static void main(String s[]) {
    Simple panel = new Simple();
    panel.addMouseListener(panel);
    panel.addMouseMotionListener(panel);
    frame = new JFrame("Graficki primer");
    frame.addKeyListener(panel);
    frame.addWindowListener(new WindowAdapter() {

public void windowClosing(WindowEvent e) {System.exit(0);}
    });
    frame.getContentPane().add("Center", panel);
    frame.setSize(new Dimension(600,600));
    frame.setVisible(true);
    }
}
```



Iako se Swing biblioteka može koristiti i sa Java verzijom 1.1 (uz dodavanje biblioteke u CLASSPATH), sve mogućnosti biblioteke su dostupne tek od verzije 1.2.

Od verzije 1.2 ova biblioteka je proglašena za standard za razvoj korisničkog interfejsa u Java aplikacijama, dok je AWT zadržan zbog kompatibilnosti sa starijim programima. Swing je postao sastavni deo veće biblioteke nazvane JFC.

3. Napisati program koji prikazuje panel i izborom prve dve tačke crta pravu a zatim svakim sledećim izborom tačke crta trougao sačinjen od prve dve izabrane tačke i zadnje izabrane tačke .

Dati tipkama sledeće uloge:
P-popunjava figuru ako nije popunja i iscrtava samo okvir ako je popunjena,
R-briše sve sa ekrana i omugućava nam da napravimo novu sliku nezavisnu od prethodne,
K-pretvara trougao u krivu II-reda,
O-ispisuje da li se nalazimo u touglu ili ne.

A-postavlja belu boju za prvu boju GradientPaint metode,
B- postavlja sivu za prvu boju GradientPaint metode,
C- postavlja crnu za prvu boju GradientPaint metode,
X- postavlja žutu za drugu boju GradientPaint metode,
Y- postavlja zelena za drugu boju GradientPaint metode,
Z- postavlja plava za drugu boju GradientPaint metode.

1-povećava veličinu kojom se icrtava linija za 5,
3-smanjuje veličinu kojom se icrtava linija na 5,
2-pomera kordinate prve tačke pravougaonika za -5 po y-osi ,
8-pomera kordinate prve tačke pravougaonika za +5 po y-osi,
4-pomera kordinate prve tačke pravougaonika za -5 po x-osi,
6-pomera kordinate prve tačke pravougaonika za +5 po x-osi.

REŠENJE:

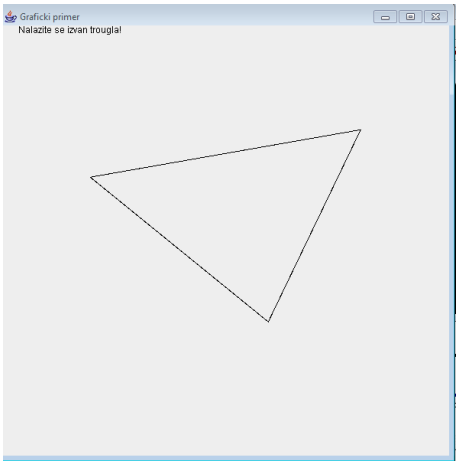
```
import java.awt.*;
import java.awt.event.*;
import javax.swing.*;
import java.awt.geom.*;
public class Simple extends JPanel
        implements MouseListener, KeyListener , MouseMotionListener
        {
  static JFrame frame;
```

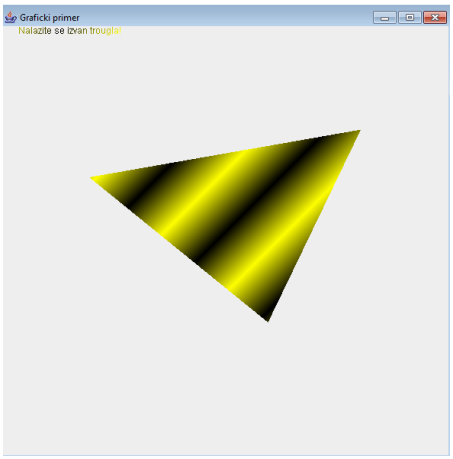
```
   int x1,y1;//Kordinate prvog temena
   int x2,y2;//Kordinate drugog temena
   int x3,y3;//Kordinate zadnjeg temena
   int korak=1;
   int oblik = 1;

   Color boja1 = new Color(0, 0, 0);
   Color boja2 = new Color(0, 0, 0);

   int debljina = 5;
   Boolean popuni;
   Boolean okvir;
  String ss="";

public Simple () {
          popuni = false;
          okvir = true;
   }

public void mouseMoved(MouseEvent e){
Point2D point = new Point2D.Float(e.getPoint().x, e.getPoint().y);

      if(oblik==1 &&  korak>3){
      GeneralPath path1 = new GeneralPath(GeneralPath.WIND_EVEN_ODD, 3);
                        path1.moveTo(x1, y1);
                  path1.lineTo(x2, y2);
                  path1.lineTo(x3, y3);
                  path1.closePath();
      if (path1.contains(point))
        ss = "Nalazite se  unutar trougla!";
      else
        ss = "Nalazite se izvan trougla!";
      }
     repaint();
      }

 public void mouseDragged(MouseEvent e){}
          public void mouseEntered(MouseEvent e) {}
          public void mouseExited(MouseEvent e) {}
          public void mouseReleased(MouseEvent e) {}
          public void mousePressed(MouseEvent e) {}
```

```
public void mouseClicked(MouseEvent e) {
                    if(korak==1)
                    { x1=e.getX();
                      y1=e.getY();
                      korak=2;
                      System.out.println("prva"+korak);
                    }
                    else if(korak==2)
                    { x2=e.getX();
                      y2=e.getY();
                      korak=3;
                      System.out.println("druga"+korak);
                    }
                    else{
                      x3=e.getX();
                      y3=e.getY();
                      korak=4;
                       System.out.println("druga"+korak);
                    }
                    repaint();
          }
          public void keyPressed(KeyEvent e) {}
          public void keyReleased(KeyEvent e) {}

public void keyTyped(KeyEvent e) {
                    switch (e.getKeyChar()) {
                              case 'p' : popuni = !popuni;
                              break;
                              case 'r' :
                                        korak = 1;
                                        okvir = true;
                                        popuni = false;
                              break;
                              case 'k' : oblik = 2;
                                       ss="Ovo je kriva drugog reda";
                              break;
                              case 'o' : okvir = !okvir;
                              break;
                              case 'a' : boja1 = Color.white;
                              break;
                              case 'b' : boja1 = Color.gray;
```

```
                        break;
                        case 'c' : boja1 = Color.black;
                        break;
                        case 'x' : boja2 = Color.red;
                        break;
                        case 'y' : boja2 = Color.blue;
                        break;
                        case 'z' : boja2 = Color.yellow;
                        break;
                        case '1' : debljina = 5;
                        break;
                        case '2' : debljina = 10;
                        break;
                        case '3' : debljina = 15;
                        break;
                }
                repaint();
        }

public void paint (Graphics g) {
        super.paint(g);
        Graphics2D g2 = (Graphics2D) g;
        //g2.setPaintMode();
GradientPaint grad = new GradientPaint(360, 200, boja1, 400, 240, boja2, true);
        g2.setPaint(grad);
        BasicStroke thick = new BasicStroke(debljina, BasicStroke.CAP_ROUND,
BasicStroke.JOIN_MITER);
        g2.setStroke(thick);
        if(korak==3) {
                g2.draw(new Line2D.Float(x1, y1, x2, y2));
                System.out.println("Linija");
                                }
        else if(korak>3){
                g2.drawString(ss, 20, 10);
                switch (oblik) {
                        case 1 :
GeneralPath path1 = new GeneralPath(GeneralPath.WIND_EVEN_ODD);
                path1.moveTo(x1, y1);
                path1.lineTo(x2, y2);
                path1.lineTo(x3, y3);
                path1.closePath();
```

```
                if (okvir == true) {
                 g2.draw(path1);
                                }
                                if (popuni == true)
                                 g2.fill(path1);
                        break;
                        case 2 :
                  QuadCurve2D quad = new QuadCurve2D.Float();
                Point2D pt1 = new Point2D.Float(x1, y1);
                                Point2D pt2 = new Point2D.Float(x2, y2);
                                Point2D ctl1 = new Point2D.Float(x3, y3);

                quad.setCurve(pt1, ctl1, pt2);
                 if (okvir == true) {
                 g2.draw(quad);
                                }
                                if (popuni == true)
                                 g2.fill(quad);

                        break;
                 }
        }
   }

public static void main(String s[]) {

   Simple panel = new Simple();
   panel.addMouseListener(panel);
   panel.addMouseMotionListener(panel);
   frame = new JFrame("Graficki primer");
   frame.addKeyListener(panel);
   frame.addWindowListener(new WindowAdapter() {
      public void windowClosing(WindowEvent e) {System.exit(0);}
   });
   frame.getContentPane().add("Center", panel);
   frame.setSize(new Dimension(600,600));
   frame.setVisible(true);
   }
}
```





3.2 Dodatni zadaci – AWT I SWING

4.Forma sa Toolbar-om

REŠENJE:

```
import java.awt.*;
import java.awt.event.*;
import javax.swing.*;

public class ToolBar extends JFrame {

    public ToolBar() {
        super("ToolBar");
        ImageIcon image1 = new ImageIcon("button1.gif");
        JButton button1 = new JButton(image1);
```

```
        ImageIcon image2 = new ImageIcon("button2.gif");
        JButton button2 = new JButton(image2);
        ImageIcon image3 = new ImageIcon("button3.gif");
        JButton button3 = new JButton(image3);

        JToolBar bar = new JToolBar();
        bar.add(button1);
        bar.add(button2);
        bar.add(button3);

        JTextArea edit = new JTextArea(8,40);
        JScrollPane scroll = new JScrollPane(edit);
        JPanel pane = new JPanel();
        BorderLayout bord = new BorderLayout();
        pane.setLayout(bord);
        pane.add("North", bar);
        pane.add("Center", scroll);
        setContentPane(pane); }

public static void main(String[] arguments) {
        ToolBar frame = new ToolBar();
        frame.pack();
        frame.setVisible(true);
    }
}
```

5.Forma za sabiranje 2 broja.

REŠENJE:

```
import java.awt.*;
import javax.swing.*;
import java.awt.event.*;
public class Zbir extends JFrame implements ActionListener {
    JTextField one = new JTextField("0", 5);
    JLabel plus = new JLabel("+");
```

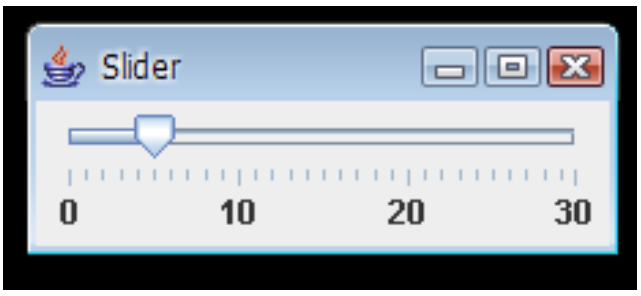
```
   JTextField two = new JTextField("0", 5);
   JButton equals = new JButton("=");
   JTextField result = new JTextField("0", 5);

 public Zbir () {
       super("Add Two Numbers");
       setSize(400, 60);
       Container pane = getContentPane();

       FlowLayout flow = new FlowLayout();
       pane.setLayout(flow);
      equals.addActionListener(this);

       pane.add(one);
       pane.add(plus);
       pane.add(two);
       pane.add(equals);
       pane.add(result);
       setContentPane(pane);
       setVisible(true);}

public static void main(String[] arguments)
{
     Zbir frame = new Zbir ();
   }

public void actionPerformed(ActionEvent evt)
{
       try {
          int value1 = Integer.parseInt(one.getText());
          int value2 = Integer.parseInt(two.getText());
          result.setText("" + (value1 + value2));
       } catch (NumberFormatException exc) {
          }
}
}
```

6.Forma za unos podatka

REŠENJE:

```
import java.awt.event.*;
import javax.swing.*;

public class TitleFrame extends JFrame {
   public TitleFrame() {
      super("Title Frame");
      setSize(300, 200);
      setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
      setVisible(true);
      String response = JOptionPane.showInputDialog(null,
         "Enter a Title for the Frame:");
      setTitle(response);
   }

 public static void main(String[] arguments) {
      JFrame frame = new TitleFrame();
   }
}
```

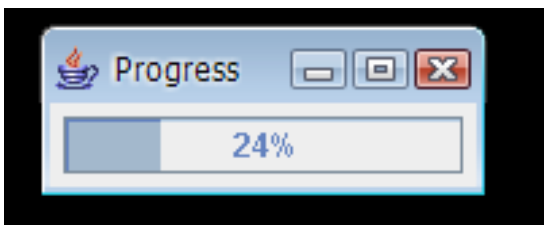
7.Slajder



REŠENJE:

```
import java.awt.event.*;
import javax.swing.*;

public class Slider extends JFrame {
   public Slider() {
      super("Slider");

setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
JSlider pickNum = new JSlider(JSlider.HORIZONTAL, 0, 30, 5);

pickNum.setMajorTickSpacing(10);
pickNum.setMinorTickSpacing(1);
      pickNum.setPaintTicks(true);
      pickNum.setPaintLabels(true);
      JPanel pane = new JPanel();
      pane.add(pickNum);
      setContentPane(pane);
   }

public static void main(String[] args) {
      Slider frame = new Slider();
      frame.pack();
      frame.setVisible(true);
   }
}
```

8.Progress bar



REŠENJE:

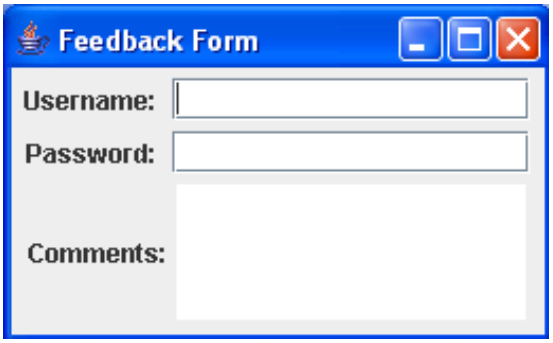
```
import java.awt.*;
import java.awt.event.*;
import javax.swing.*;
```

```
public class Progress extends JFrame {
   JProgressBar current;
   JTextArea out;
   JButton find;
   Thread runner;
   int num = 0;
   public Progress() {
      setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
      JPanel pane = new JPanel();
      pane.setLayout(new FlowLayout());
      current = new JProgressBar(0, 2000);
      current.setValue(0);
      current.setStringPainted(true);
      pane.add(current);
      setContentPane(pane);
   }
   public void iterate() {
      while (num < 2000) {
         current.setValue(num);
         try {
            Thread.sleep(1000);
         } catch (InterruptedException e) { }
         num += 95;
      }
   }
   public static void main(String[] arguments) {
      Progress frame = new Progress();
      frame.pack();
      frame.setVisible(true);
      frame.iterate();
   }
}
```

9. Napisati program "Prijava":

-importovati paket Swing
-klasa nasledjuje Jframe

Program prikazuje Prozor Prijava koja sadrzi polja za unos:
-korisnickog imena
-lozinke
-komentara



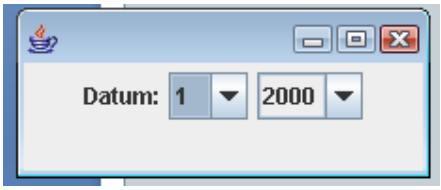
REŠENJE:

```
import javax.swing.*;

public class Prijava extends JFrame {
   JTextField username = new JTextField(15);
   JPasswordField password = new JPasswordField(15);
   JTextArea comments = new JTextArea(4, 15);
   public Prijava() {

    setSize(260, 160);
      JPanel pane = new JPanel();
      JLabel usernameLabel = new JLabel("Username: ");
      JLabel passwordLabel = new JLabel("Password: ");
      JLabel commentsLabel = new JLabel("Comments:");
      pane.add(usernameLabel);
      pane.add(username);
      pane.add(passwordLabel);
      pane.add(password);
      pane.add(commentsLabel);
      pane.add(comments);

      setContentPane(pane);
      show();
   }
   public static void main(String[] arguments) {
      Prijava input = new Prijava();
   }
}
```

10.Napisati program "Datum":

-importovati paket Swing
-klasa nasledjuje Jframe

Program prikazuje Datum sa dva padajuca menija:
-za mesec
-za godinu



REŠENJE:

```
import javax.swing.*;

public class Datum extends JFrame {
   JComboBox mesec = new JComboBox();
   JComboBox godina= new JComboBox();

   public Datum() {

      setSize(220, 90);
      JPanel pane = new JPanel();
      JLabel exp = new JLabel("Datum:");
      pane.add(exp);
      for (int i = 1; i < 13; i++)
         mesec.addItem("" + i);
      for (int i = 2000; i < 2010; i++)
         godina.addItem("" + i);
      pane.add(mesec);
      pane.add(godina);
            setContentPane(pane);
      show();    }
   public static void main(String[] arguments) {
      Datum ct = new Datum();
   }
}
```

KONTROLNA PITANJA

1.Šta je AWT, a šta Swing?

2.Objasniti sledeći red koda:
public class Simple extends JPanel implements MouseListener, KeyListener

3. Kako se definišu krive?

4.Šta radi JComboBox, a šta JTextArea

5.Kako se difiniše gradijent?

6. Objasniti sledeći red koda:
public class Datum extends JFrame

7.Šta rade sledeće naredbe:
BasicStroke, GeneralPath

8.Napravi razliku između sledećih naredbi:
Point2D, Graphics2D, Ellipse2D

9. Šta radi sledeći kod:
setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);

10.Šta radi sledeći kod:
JOptionPane.showInputDialog

11. Kako odraditi sledeću metodu koja povećava veličinu kojom se icrtava linija za 5?

12. Kako odraditi sledeću metodu koja pomera kordinate prve tačke pravougaonika za +5 po y-osi?

13. Kako odraditi sledeću metodu koja omogućava tako da prilikom prelaska preko figure, figura postaje popunjena ?

14. Objasniti sledeću naredbu e.getKeyChar())

15. Objasniti sledeću naredbu panel.addMouseMotionListener(panel);

DODATAK A

PRIMENA PROGRAMSKOG JEZIKA JAVA U IZRADI INFORMACIONOG SISTEMA

IZRADA INFORMACIONOG SISTEMA NA TEMU "PRIVATNA SKOLA ENGLESKOG JEZIKA"

Zadatak ovog projekta jeste:

- *unos, brisanje i ažuriranje kandidata i profesora kao i vrste kurseva koje pohadjaju*
- *pregled kandidata i profesora kao i vrste kurseva koje pohadjaju*
- *izdavanje izvještaja o položenim kursevima i podaci o kandidatima i profesora*

Objekti koji su bitni u sistemu jesu :

- ***kandidati(učenici),kursevi i predavači.***

Svi kandidati čine klasu *Ucenici*. Učenici predstavljaju one koji koriste usluge i pohadjaju kurseve. Ova klasa ima atribute redni broj, ime, prezime, datum rodjenja, i adresu.

Sve vrste kurseva čine klasu *Kursevi*. Objekti ove klase povezani su sa profesorima preko ključa id_prof. Atributi klase su broj kursa,naziv kursa, duzina pohadjanja,ime profesora koji je preuzelo kurs.

Svi profesori čine klasu *Predavaci*. Predavači predstavljaju one koji daju usluge i vezani su za kurseve. Ova klasa ima atribute redni broj, ime, prezime, zvanje, adresu.

A.1 Konekcija na bazu podataka

KOD ZA USPOSTAVLJANJE VEZE SA BAZOM PODATAKA (J-ODBC)

```
import java.sql.*;
class Baza {
        java.sql.Connection dbConn=null;
        Statement naredba1=null;
        public void poveziSaBazom(){
                try {
                        //uspostavljanje veze
                        Class.forName ("sun.jdbc.odbc.JdbcOdbcDriver");
```

```
                    //navodjenje ODBC izvora
                    String sourceURL= "jdbc:odbc:PFEJ";

                    //uspostavljanje veze
                    dbConn= DriverManager.getConnection(sourceURL);

                    // ako smo stigli ovde, veza je ok
                    System.out.println ("Veza sa bazom je: "+dbConn);
                    System.out.println ("Veza uspostavljena\\n");


                    naredba1=dbConn.createStatement();
                    }

                    catch (Exception e){
          System.out.println ("Izuzetak izbacen"+ e.getMessage());
                              }
    }


public  void zatvoriBazu(){
                              try
                    {
                              if (naredba1 != null)
                                        naredba1.close();
                              if (dbConn != null)
                                        dbConn.close();
                    }
                    catch (SQLException sqle)
                    {
               System.out.println("SQLIzuzetak za vreme close(): " +
                                        sqle.getMessage());
                    }
               }
               }
```

A.2 Glavna forma

Na glavnoj formi nalazi se meni sa stavkama:

1.PODACI
2.PREGLED
3.ZAPISNIK
4.AUTOR

REŠENJE:

```
import java.awt.*;
 import java.awt.event.*;
 import javax.swing.*;
 import javax.swing.event.*;
 import java.awt.Toolkit;
 import javax.swing.JToolBar;
 import java.io.*;
```

```
//izvrsna klasa koja sadrzi glavni meni
public class PFEJ implements ActionListener {
static JFrame  Prozor=new JFrame ("PRIVATNA SKOLA ENGLESKOG
JEZIKA");//glavni prozor caption

private JMenuItem unosUcenika;     //stavke podmenia za unos,
private JMenuItem unosKurseva;  //stavke podmenija za unos premeta
 private JMenuItem unos_Predavaca;  //stavke podmenija za unos profesora

private JMenuItem pregledUcenika;  //pregled,
private JMenuItem pregled_Predavaca;
private JMenuItem pregledKurseva;

private JMenuItem unosKursaPolozenog;
private JMenuItem pregledKursaPolozenog;
private JMenuItem about;
private JMenuItem home;

private static JMenu un=new JMenu ("PODACI     ");// meni za unos
private static JMenu pr=new JMenu ("PREGLED    ");//meni za pregled
private static JMenu k=new JMenu ("ZAPISNIK     ");
private static JMenu ab=new JMenu ("AUTOR   ");
private static JMenu ho=new JMenu ("HOME     ");

JButton Bucenik, Bpredavac, Bkurs, Bp_kurs, Bp_ucenik, Bp_predavac, Bautor, Bizlaz;
        JToolBar tlbDugmad;
        JScrollPane sp2;
        JPanel p0;
GridLayout gl = new GridLayout (2,1);

 public PFEJ(){
                Bucenik = new JButton();
                Bpredavac = new JButton();
                Bkurs = new JButton();
                Bp_kurs = new JButton();
                Bp_ucenik = new JButton();
                Bp_predavac = new JButton();
                Bautor = new JButton();
                Bizlaz = new JButton();
```

```
            tlbDugmad = new JToolBar(JToolBar.HORIZONTAL);
            tlbDugmad.add(Bucenik);
            tlbDugmad.add(Bpredavac);
            tlbDugmad.add(Bkurs);
            tlbDugmad.add(Bp_kurs);
            tlbDugmad.add(Bp_ucenik);
            tlbDugmad.add(Bp_predavac);
            tlbDugmad.add(Bautor);
            tlbDugmad.add(Bizlaz);

            Bucenik.setIcon(new ImageIcon("wi0062-48.gif"));
            Bpredavac.setIcon(new ImageIcon("wi0054-48.gif"));
            Bkurs.setIcon(new ImageIcon("wi0122-48.gif"));
            Bp_kurs.setIcon(new ImageIcon("wi0126-48.gif"));
            Bp_ucenik.setIcon(new ImageIcon("literat.gif"));
            Bp_predavac.setIcon(new ImageIcon("front1.gif"));
            Bautor.setIcon(new ImageIcon("wi0063-48.gif"));
            Bizlaz.setIcon(new ImageIcon("g0.gif"));

            sp2 = new JScrollPane(tlbDugmad);
            sp2.setEnabled(true);

            Bucenik.addActionListener(this);
            Bpredavac.addActionListener(this);
            Bkurs.addActionListener(this);
            Bp_kurs.addActionListener(this);
            Bp_ucenik.addActionListener(this);
            Bp_predavac.addActionListener(this);
            Bautor.addActionListener(this);
            Bizlaz.addActionListener(this);

            BorderLayout bl = new BorderLayout();
            Prozor.setLayout(bl);

    slika1.setIcon(new ImageIcon("POZ.jpg"));
    slika2.setIcon(new ImageIcon("BANER.jpg"));

    Prozor.add(slika1, BorderLayout.CENTER);
    Prozor.add(sp2, BorderLayout.NORTH);
Prozor.add(slika2, BorderLayout.SOUTH);
    }
```

```
 JLabel slika1=new JLabel();
 JLabel slika2=new JLabel();

public static void main (String [] args){
        //formiramo istancu klase
PFEJ obj=new PFEJ();

        //postavljamo dimenzije prozora
        Toolkit alat= Prozor.getToolkit();
        Dimension prozorVel= alat.getScreenSize();
        Prozor.setBounds(prozorVel.width/8,prozorVel.height/8,535,638);

        //postavlja se glavni meni
        JMenuBar meniLinija=new JMenuBar ();
        Prozor.setJMenuBar(meniLinija);

//dodajemo podmenie i precice za njih
meniLinija.add(un);
        obj.dodajUn();
        un.setMnemonic('P');
        meniLinija.add(pr);

obj.dodajPr();
        pr.setMnemonic('R');
        k.setMnemonic('Z');
        meniLinija.add(k);

obj.dodajK();
        meniLinija.add(ab);

obj.dodajAb();
        ab.setMnemonic('A');
        meniLinija.add(ho);
obj.dodajHo();
        ho.setMnemonic('H');

        Prozor.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
        Prozor.setVisible(true);
}
```

```
void dodajUn(){

        //stavka za unos ucenika
unosUcenika=new JMenuItem (" Ucenici");
 un.add(unosUcenika);
unosUcenika.setAccelerator(KeyStroke.getKeyStroke('S', Event.CTRL_MASK));
        unosUcenika.addActionListener(this);
        //separator
        un.addSeparator();

        //stavka za unos predmeta
unosKurseva=new JMenuItem(" Kursevi");
un.add(unosKurseva);
unosKurseva.setAccelerator(KeyStroke.getKeyStroke('D', Event.CTRL_MASK));
        unosKurseva.addActionListener(this);

        //separator
        un.addSeparator();
        //stavka za unos profesora

unos_Predavaca=new JMenuItem(" Predavaci");
un.add(unos_Predavaca);
unos_Predavaca.setAccelerator(KeyStroke.getKeyStroke('N', Event.CTRL_MASK));

unos_Predavaca.addActionListener(this);
 un.addSeparator();un.addSeparator();
        // stavka za izlaz

JMenuItem izlaz=new JMenuItem("Exit");
        izlaz.addActionListener(new ActionListener (){
                public void actionPerformed (ActionEvent dogadjaj )
                  {
                        System.exit(0);
                  }
                  });

un.add(izlaz);

izlaz.setAccelerator(KeyStroke.getKeyStroke('X', Event.CTRL_MASK));
        }
```

```
void dodajPr(){
        //stavka za unos
pregledUcenika=new JMenuItem (" Ucenika");
pr.add(pregledUcenika);
 pregledUcenika.setAccelerator(KeyStroke.getKeyStroke('L', Event.CTRL_MASK));
        pregledUcenika.addActionListener(this);
        //separator
        pr.addSeparator();
        //stavka za unos predmeta
        pregledKurseva=new JMenuItem (" Kurseva");
        pr.add(pregledKurseva);
pregledKurseva.setAccelerator(KeyStroke.getKeyStroke('P', Event.CTRL_MASK));
        pregledKurseva.addActionListener(this);
        //separator
        pr.addSeparator();
        //stavka za unos
pregled_Predavaca=new JMenuItem (" Predavaca");
pr.add(pregled_Predavaca);
pregled_Predavaca.setAccelerator(KeyStroke.getKeyStroke('G', Event.CTRL_MASK));

pregled_Predavaca.addActionListener(this);
        }

void dodajK(){

//stavka za unoS
        unosKursaPolozenog=new JMenuItem (" Unos");
        k.add(unosKursaPolozenog);
unosKursaPolozenog.setAccelerator(KeyStroke.getKeyStroke('I',
Event.CTRL_MASK));
        unosKursaPolozenog.addActionListener(this);
        pr.addSeparator();
        pregledKursaPolozenog=new JMenuItem (" Pregled");
        k.add(pregledKursaPolozenog);

pregledKursaPolozenog.setAccelerator(KeyStroke.getKeyStroke('I',
Event.CTRL_MASK));
pregledKursaPolozenog.addActionListener(this);
        }
```

```
void dodajAb(){

about=new JMenuItem (" O Autoru");
 ab.add(about);
about.setAccelerator(KeyStroke.getKeyStroke('A', Event.CTRL_MASK));
            about.addActionListener(this);
            }

void dodajHo(){

home=new JMenuItem (" Pocetna");
ho.add(home);
home.setAccelerator(KeyStroke.getKeyStroke('A', Event.CTRL_MASK));
            home.addActionListener(this);
            }

 //metoda za realizaciju akcija na osnovu stavke koja je izabrana u meniu

public void actionPerformed (ActionEvent e){
if(e.getSource()==unosUcenika || e.getSource()==Bucenik){
 System.out.println(Prozor.getContentPane());
unosUcenika pan=new unosUcenika();
 pan.show();
}

if(e.getSource()==unos_Predavaca || e.getSource()==Bpredavac){
System.out.println(Prozor.getContentPane());
unos_Predavaca pan=new unos_Predavaca();
 pan.show();
}

if(e.getSource()==unosKurseva || e.getSource()==Bkurs){
System.out.println(Prozor.getContentPane());
unosKurseva pan=new unosKurseva();
 pan.show();
}

if(e.getSource()==pregledKurseva){
System.out.println(Prozor.getContentPane());
PregledKurs pan=new PregledKurs();
 pan.show();
```

```
}

if(e.getSource()==pregledUcenika || e.getSource()==Bp_ucenik){
System.out.println(Prozor.getContentPane());
PregledUcenika pan=new PregledUcenika();
pan.show();
}

if(e.getSource()==pregled_Predavaca || e.getSource()==Bp_predavac){
 System.out.println(Prozor.getContentPane());
PregledPredavaci pan=new PregledPredavaci();
pan.show();
}

if(e.getSource()==unosKursaPolozenog || e.getSource()==Bp_kurs){
System.out.println(Prozor.getContentPane());
UnosKursaPolozenog pan=new UnosKursaPolozenog();
pan.show();
}

if(e.getSource()==pregledKursaPolozenog){
System.out.println(Prozor.getContentPane());
PregledKursPolozeni pan=new PregledKursPolozeni();
pan.show();
}

if(e.getSource()==about || e.getSource()==Bautor){
System.out.println(Prozor.getContentPane());
about pan=new about();
pan.show();}

if(e.getSource()==home){
System.out.println(Prozor.getContentPane());
pocetna pan=new pocetna();
Prozor.add(pan);
Prozor.setContentPane(pan);
Prozor.show();
                    }
          }
}
```

OBJAŠNJENJE POJEDINIH SEGMENATA GLAVNE FORME

Ako posmatramo navedenu formu ovog informacionog sistema možemo uočiti da za svaku komponentu koja je obrađena u Abstract Windowing Toolkit-u, postoji odgovarajuća Swing komponenta.

Tako na primer labele su implementirane u Swing upotrebom klase JLabel. Konstruktor metodi:

- JLabel(String, int) natpis sa tekstom i poravnanjem koje je definisano jednom od tri specifikacije iz klase SwingConstants: LEFT, CENTER, RIGHT
- JLabel(String, Icon, int) natpis sa tekstom, ikonom i poravnanjem

U ovom projektu se takođe koriste JButton komponente. Za Swing dugme koristi se klasa JButton. Konstruktor metodi

- JButton(String)
- JButton(Icon)
- JButton(String, Icon)

Tekst polja su implementirana u Swing klasom JTextField. Razlika između ovih tekst polja i njihovih AWT parnjaka je što metod za ispis skrivenog unosa teksta, setEchoChar(char), nije podržan u klasi JTextField. Konstruktor metodi:

- JTextField(int) tekst polje specificirane širine.
- JTextField(String, int) tekst polje koji sadrži dati tekst i ima specificiranu širinu

Klasa JPasswordField se koristi za kreiranje tekst polja koja koriste karaktere za skrivanje unosa. Ova klasa poseduje konstruktor metode kao i JTextField: JPasswordField(int) i JPasswordField(String, int).

Tekst oblasti su implementirane u Swing klasom JTextArea. Konstruktor metodi:

- JTextArea(int, int) tekst oblast sa navedenim brojem redova i kolona,
- JTextArea(String, int, int) tekst oblast sa navedenim tekstom, brojem redova i kolona

Klasa JCheckBox je implementacija za check box-ove u Swing-u. Funkcionalnost je ista kao kod AWT sem čto sadrže i ikone kao obeležje.

Konstruktor metodi :

- JCheckBox(String) check box sa navedenim tekst natpisom
- JCheckBox(String, boolean) check box sa navedenim tekst natpisom koji je aktiviran ako je drugi argument true
- JCheckBox(Icon) check box koji sadrži navedenu ikonu
- JCheckBox(Icon, boolean) check box koji sadrži navedenu ikonu i koji je aktivirana ako je drugi argument true
- JCheckBox(String, Icon) check box koji sadrži navedeni tekst i ikonu
- JCheckBox(String, Icon, boolean) check box koji sadrži navedeni tekst i ikonu koji su aktivirani ako treći argument je true
- Check box grupe su implementirane u Swingupotrebom klase ButtonGroup.

Radio dugmad su implementirana u Swing posredstvom klase JRadioButton. Konstruktor metode su iste kaoi u slučaju klase JCheckBox.

Liste izbora koje su bile kreirane u AWT-u upotrebom klase Choice, su prisutne posredstvom klase JComboBox.

Koraci pri kreiranju su:
1. Konstruktor JComboBox() se koristi bez argumenata.
2. Metod polja za potvrdu addItem(Object) se koristi da bi se dodale stavke na listi.
3. Metod polja za potvrdu setEditable(boolean) se koristi sa argumentom false.

Scrollbarsi su implementirane u Swing putem klase JScrollBar. Konstruktor metodi:

- JScrollBar(int) scrollbar sa navedenom orientacijom.
- JScrollBar(int, int, int, int, int) scrollbar sa navedenom orijentacijom, početnom vrednosti, scroll box veličinom, minimalnom vrednošću, maximalnom vrednošću.

- Orijentacija je indicirana promenljivima klase SwingConstant HORIZONTAL ili VERTICAL.

Meni PODACI sluzi za unos, brisanje, pretraživanje i modifikaciju sledećih podataka:

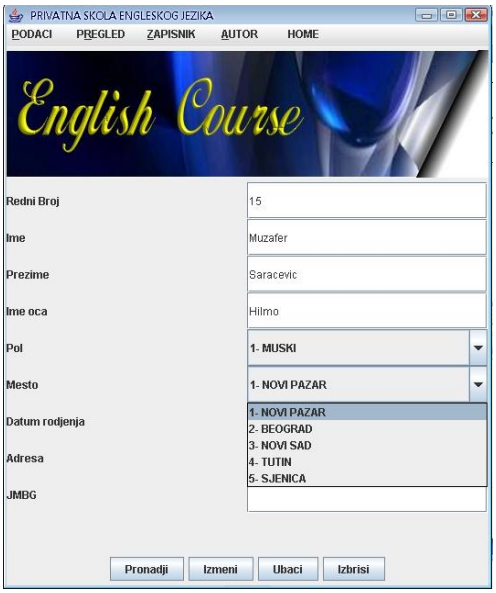
Učenika,Kurseva,Predavača

Meni PREGLED služi za tabelarni prikaz već unetih podataka.

Učenika,Kurseva,Predavača sa mogučnošću pretrage po kriterijumu.

Meni ZAPISNIK služi za pregled, unos, brisanje, pretraživanje i modifikaciju: ***Položenih kurseva, koji se navode u zapisniku***

A.3 Forma za unos,brisanje,promenu i pretragu



REŠENJE:

```
import javax.swing.*;
import java.awt.*;
import java.awt.event.*;
import java.sql.*;
import java.util.*;

public class unosUcenika extends JFrame implements ActionListener {

   Baza baza=new Baza();          //Pravimo instancu za bazu.
   private int r_br;
   private String ime;
         private String prezime;
         private String ime_oca;
         private int pol;
         private int mesto;
         private String datum_rodj;
         private String adresa;
         private String JMBG;
         private String p;
         private String m;

         JTextField tfr_br;
         JTextField tfime;
         JTextField tfprezime;
         JTextField tfime_oca;
         JComboBox cbpol;
         JComboBox cbmesto;
         JTextField tfdatum_rodj;
         JTextField tfadresa;
         JTextField tfJMBG;

         JLabel lr_br= new JLabel("Redni Broj");
         JLabel lime = new JLabel("Ime");
         JLabel lprezime = new JLabel("Prezime");
         JLabel lime_oca = new JLabel("Ime oca");
         JLabel lpol = new JLabel("Pol");
         JLabel lmesto = new JLabel("Mesto");
         JLabel ldatum_rodjl = new JLabel("Datum rodjenja");
         JLabel ladresa = new JLabel("Adresa");
```

```
        JLabel lJMBG = new JLabel("JMBG");
        GridLayout gl = new GridLayout (10,2);

        JButton dPronadji;
        JButton dSmesti;
        JButton dUbaci;
        JButton dIzbrisi;
        JPanel p1;
        JPanel p2;
        JPanel p0;

public unosUcenika() {
        listanje lis = new listanje(); // pravimo instancu liste izvestaja
        String[] Pol=lis.izlistajPol();
        String[] Mesto=lis.izlistajMesto();
        tfr_br = new JTextField(20);
        tfime = new JTextField(20);
        tfprezime = new JTextField(20);
        tfime_oca = new JTextField(20);
        cbpol = new JComboBox(Pol);
        cbmesto = new JComboBox(Mesto);
        tfdatum_rodj = new JTextField(20);
        tfadresa = new JTextField(20);
        tfJMBG = new JTextField(20);
        this.setSize(535, 638);
        this.setTitle("UNOS UCENIKA");
        this.setLocation(700, 100);

        p1 = new JPanel();
        p2 = new JPanel();
        p0 = new JPanel();

        dPronadji = new JButton("Pronadji");
                dSmesti = new JButton("Izmeni");
                dUbaci = new JButton ("Ubaci");
                dIzbrisi = new JButton ("Izbrisi");
        slika1.setIcon(new ImageIcon("picture1.jpg"));
                p2.add(slika1);

        p0.setLayout(gl);
                p0.add(lr_br);
```

```
                p0.add(tfr_br);
                p0.add(lime);
                p0.add(tfime);
                p0.add(lprezime);
                p0.add(tfprezime);
                p0.add(lime_oca);
                p0.add(tfime_oca);
                p0.add(lpol);
                p0.add(cbpol);
                p0.add(lmesto);
                p0.add(cbmesto);
                p0.add(ldatum_rodjl);
                p0.add(tfdatum_rodj);
                p0.add(ladresa);
                p0.add(tfadresa);
                p0.add(lJMBG);
                p0.add(tfJMBG);

                p1.add(dPronadji);
                p1.add(dSmesti);
                p1.add(dUbaci);
                p1.add(dIzbrisi);

                BorderLayout bl = new BorderLayout();
                this.setLayout(bl);

                this.add(p0, BorderLayout.CENTER);
                 this.add(p2, BorderLayout.NORTH);
                this.add(p1, BorderLayout.SOUTH);

                dPronadji.addActionListener(this);
                dSmesti.addActionListener(this);
                dUbaci.addActionListener(this);
                dIzbrisi.addActionListener(this);

}
JLabel slika1=new JLabel();

public void actionPerformed (ActionEvent e){
                if (e.getActionCommand().equals("Pronadji"))
                {
```

```
            System.out.println("dPronadji pritisnuto");
            if (tfr_br.getText().equals(""))
                  System.out.println("Molim Vas unesite ");
            else
            {
                  System.out.println("r_br = " + tfr_br.getText());
                  this.r_br = Integer.parseInt(tfr_br.getText());
                  pronadjiUBazi();
                  tfime.setText(ime);
                  tfprezime.setText(prezime);
                  tfime_oca.setText(ime_oca);
                  cbpol.setSelectedItem(p);
                  cbmesto.setSelectedItem(m);
                  tfdatum_rodj.setText(datum_rodj);
                  tfadresa.setText(adresa);
                  tfJMBG.setText(JMBG);

            }
      }
      if (e.getActionCommand().equals("Izmeni"))
      {
            System.out.println("dSmesti pritisnuto");
            if (tfr_br.getText().equals(""))
                  System.out.println("Molim Vas, unesite ");
            else
            {
                  azuriratiBazu();
                        }
      }
      if (e.getActionCommand().equals("Ubaci"))
      {
            System.out.println("dSmesti pritisnuto");
            if (tfr_br.getText().equals(""))
                  System.out.println("Molim Vas, unesite ");
            else
            {
                  ubaci_u_Bazu();
                        }
      }
      if (e.getActionCommand().equals("Izbrisi"))
      {
```

```
                        System.out.println("dIzbrisi pritisnuto");
                        if (tfr_br.getText().equals(""))
                                System.out.println("Molim Vas, unesite ");
                        else
                        {
                         izbrisiBazu();
                        }
                }
}

   public String ubaci_u_Bazu() {
                try
                {
                        // ====== Uspostavljanje veze sa Bazom ======

                        baza.poveziSaBazom(); //Pravimo vezu sa bazom.


                        r_br= Integer.parseInt(tfr_br.getText());
                        ime = tfime.getText();
                        prezime = tfprezime.getText();
                        ime_oca = tfime_oca.getText();
pol =
Integer.parseInt(cbpol.getSelectedItem().toString().substring(0,cbpol.getSelectedItem().
toString().indexOf('-')));
mesto =
Integer.parseInt(cbmesto.getSelectedItem().toString().substring(0,cbmesto.getSelectedIt
em().toString().indexOf('-')));
                        datum_rodj = tfdatum_rodj.getText();
                        adresa = tfadresa.getText();
                        JMBG = tfJMBG.getText();
                String proveri="Select * from ucenici where r_br=" + r_br;
                        ResultSet r=baza.naredba1.executeQuery(proveri);
                        if (!r.next())
                        {

                        //Ubaci
                String   azurirajString =
                "INSERT INTO  ucenici VALUES('" +r_br+"','"+
                ime+"','"+prezime+"','"+ime_oca+"','"+pol+"','"+
                mesto+"','"+datum_rodj+"','"+adresa+"','"+JMBG+"')";
```

```
                System.out.println(azurirajString);
                        baza.naredba1.executeUpdate(azurirajString);
                        return "Uspesno azuriranje";
                }
                else
                System.out.println("Postoji index sa tim broj");
                return "greska";
        }
                catch (Exception e)
                {
        System.out.println("Izbacen izuzetak: " + e.getMessage());
                        return "Neuspesno pretrazivanje";
                }
                finally
                {
                        try
                        {
                                if (baza.naredba1 != null)
                                        baza.naredba1.close();
                                if (baza.dbConn != null)
                                        baza.dbConn.close();
                        }
                        catch (SQLException sqle)
                        {
System.out.println("SQLIzuzetak za vreme close(): " + sqle.getMessage());
                        }
                }

        }

public String izbrisiBazu() {
                try
                {
                        baza.poveziSaBazom();

                        r_br = Integer.parseInt(tfr_br.getText());

                        String izbrisiString =
                                "Delete from ucenici" +
                                " WHERE r_br = " + r_br;
                        System.out.println(izbrisiString);
```

```
                        baza.naredba1.executeUpdate(izbrisiString);

            return "Uspesno brisanje";
                }
                catch (Exception e)
                {
System.out.println("Izbacen izuzetak: " + e.getMessage());
                        return "Neuspesno pretrazivanje";
                }
                finally
                {
                        try
                        {
                                if (baza.naredba1 != null)
                                        baza.naredba1.close();
                                if (baza.dbConn != null)
                                        baza.dbConn.close();
                        }
                        catch (SQLException sqle)
                        {
        System.out.println("SQLIzuzetak za vreme close(): " +
                                        sqle.getMessage());
                        }
                }
        }

public String azuriratiBazu()
        {
                try
                {
                        // ==== Uspostavljanje veze sa Bazom ===

                        baza.poveziSaBazom();
                        r_br= Integer.parseInt(tfr_br.getText());
                        ime = tfime.getText();
                        prezime = tfprezime.getText();
                        ime_oca = tfime_oca.getText();
pol =
Integer.parseInt(cbpol.getSelectedItem().toString().substring(0,cbpol.getSelectedItem().
toString().indexOf('-')));
```

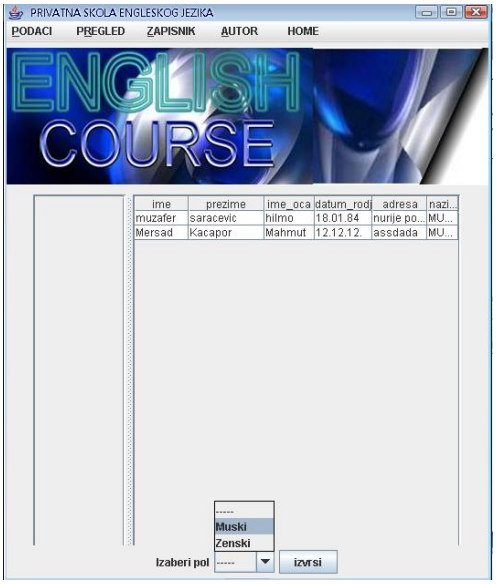
```
mesto =
Integer.parseInt(cbmesto.getSelectedItem().toString().substring(0,cbmesto.getSelectedIt
em().toString().indexOf('-')));
                        datum_rodj = tfdatum_rodj.getText();
                        adresa = tfadresa.getText();
                        JMBG = tfJMBG.getText();

//Azuriranje imena UCENIKA
                        String azurirajString =
                                "Update ucenici " +
                                "SET ime = '" + ime + "' " +
                                "WHERE r_br = " + r_br;
                        System.out.println(azurirajString);
                        baza.naredba1.executeUpdate(azurirajString);

//Azuriranje prezimena UCENIKA
                        azurirajString =
                                "Update ucenici " +
                                "SET prezime = '" + prezime + "' " +
                                "WHERE r_br = " + r_br;
                        System.out.println(azurirajString);
                        baza.naredba1.executeUpdate(azurirajString);

//Azuriranje - ime oca
                        azurirajString =
                                "Update ucenici " +
                                "SET ime_oca = '" + ime_oca + "' " +
                                "WHERE r_br = " + r_br;
                        System.out.println(azurirajString);
                        baza.naredba1.executeUpdate(azurirajString);

//Azuriranje pola
                        azurirajString =
                                "Update ucenici " +
                                "SET pol = " + pol + " " +
                                "WHERE r_br = " + r_br;
                        System.out.println(azurirajString);
                        baza.naredba1.executeUpdate(azurirajString);

//Azuriranje mesta
                        azurirajString =
```

```
                    "Update ucenici " +
                    "SET mesto = " + mesto + " " +
                    "WHERE r_br = " + r_br;
                System.out.println(azurirajString);
                baza.naredba1.executeUpdate(azurirajString);

//Azuriranje datuma rodjenja
                azurirajString =
                    "Update ucenici " +
                    "SET datum_rodj = " + datum_rodj + " " +
                " WHERE r_br = " + r_br;
                System.out.println(azurirajString);
                baza.naredba1.executeUpdate(azurirajString);

//Azuriranje adrese stanovanja
                azurirajString =
                    "Update ucenici " +
                    "SET adresa = '" + adresa + "' " +
                    "WHERE r_br = " + r_br;
                System.out.println(azurirajString);
                baza.naredba1.executeUpdate(azurirajString);

//Azuriranje JMBG-a
                azurirajString =
                    "Update ucenici " +
                    "SET JMBG = '" + JMBG + "' " +
                    "WHERE r_br = " + r_br;
                System.out.println(azurirajString);
                baza.naredba1.executeUpdate(azurirajString);

                    return "Uspesno azuriranje";
            }
            catch (Exception e)
            {
System.out.println("Izbacen izuzetak: " + e.getMessage());
                    return "Neuspesno pretrazivanje";
            }
            finally
            {
                    try
                    {
```

```
                                        if (baza.naredba1 != null)
                                                baza.naredba1.close();
                                        if (baza.dbConn != null)
                                                baza.dbConn.close();
                                }
                                catch (SQLException sqle)
                                {
System.out.println("SQLIzuzetak za vreme close(): " +
                                                sqle.getMessage());
                                }
                        }

        }

public String pronadjiUBazi()
        {
                try
                {
                // ======== Uspostavljanje veze sa bazom ===
                        baza.poveziSaBazom();
                        r_br=Integer.parseInt(tfr_br.getText());
                        //Pretraga
String naredbaSQL =
        "SELECT * FROM ucenici s where s.r_br = " + r_br;

ResultSet rezultat = baza.naredba1.executeQuery(naredbaSQL);
                        while (rezultat.next())
                        {
                                ime = rezultat.getString("ime");
                                prezime= rezultat.getString("prezime");
                                ime_oca= rezultat.getString("ime_oca");
                                datum_rodj= rezultat.getString("datum_rodj");
                                adresa=rezultat.getString("adresa");
                                JMBG= rezultat.getString("JMBG");
                                System.out.println(ime);
                        }

String naredbaSQL3= "select p.id_pola, p.naziv_pola from pol p, ucenici s"+
" where s.pol = p.id_pola and s.r_br = " +r_br;
ResultSet r2 = baza.naredba1.executeQuery(naredbaSQL3);
                while (r2.next())
```

```
                    {
pol=r2.getInt("id_pola");
p=Integer.toString(pol)+"- "+r2.getString("naziv_pola");
System.out.println(p);
                                        }
String naredbaSQL2= "select m.id_mesta, m.naziv_mesta from mesto m, ucenici s"+
" where s.mesto = m.id_mesta and s.r_br = " +r_br;
ResultSet r = baza.naredba1.executeQuery(naredbaSQL2);

while (r.next()) {
mesto=r.getInt("id_mesta");
m=Integer.toString(mesto)+"- "+r.getString("naziv_mesta");
System.out.println(m);
}
                    return "Uspesno pretrazivanje";
                }
                catch (Exception e)
                {
        System.out.println("Izuzeztak izbacen: " + e.getMessage());
                    return "Neuspesno pretrazivanje";
                }
                finally
                {
                    try
                    {
                        if (baza.naredba1 != null)
                            baza.naredba1.close();
                        if (baza.dbConn != null)
                            baza.dbConn.close();
                    }
                    catch (SQLException sqle)
System.out.println("SQLIzuzetak za vreme close(): " +
                                        sqle.getMessage()); }
                    }
                }
        }
```

A.4 Forma za listanje



REŠENJE:

```
import javax.swing.*;
import java.awt.*;
import java.awt.event.*;
import java.util.EventObject;
import java.sql.*;
import java.util.Vector;

public class PregledUcenika extends JFrame
{
   JPanel p1 ;           // Panel na kom se nalaze uslovi
   JComboBox cb1;        // Combo box
   JLabel lbl1;          // Labela
         JTable table;         // Tabela rezultata
         JButton izvrsi;       // Dugme za izvrsenje naredbe
         JScrollPane tableScroller;  //Skrolovani panel za tebelu
   JScrollPane panel;                //Skrolovani panel za tebelu
     String naredba = "SELECT
ucenici.ime,ucenici.prezime,ucenici.ime_oca,ucenici.datum_rodj,ucenici.adresa,pol.nazi
v_pola from ucenici,pol " ; //Prvi deo upita
     String uslov=" WHERE ucenici.pol=pol.id_pola";       //String sa uslovima
```

```
    String uslov2="";

   JPanel p3;
        JPanel p2;
        JPanel p0;

      /*KONSTRUKTOR*/
        public PregledUcenika()
        {
                String s=pregledBaze();
                this.setSize(535, 638);
                this.setTitle("PREGLED UCENIKA");
                this.setLocation(700, 100);
        }
        // Funkcija koja kreira tabelu sa odgovarajucim upitom
public JTable createTable()
        {
                String colNames[] = null;  //String-niz imena kolona
           Baza baza=new Baza();      //Istanca baze
                Vector dataSet = new Vector(); //Vektor rezultata.
                try{

// ===== Uspostavljanje veze sa bazom ===========
                baza.poveziSaBazom();
                /*Konstruisemo upit*/

String naredbaSQL = naredba+uslov+uslov2;
System.out.println(naredbaSQL);

/*Izvrsavamo upit*/
        ResultSet rezultat1 = baza.naredba1.executeQuery(naredbaSQL);
                        /*Uzimamo podatke o kolonama*/
                        ResultSetMetaData md = rezultat1.getMetaData();
                        colNames = new String[md.getColumnCount()];
                        for(int i=0;i<colNames.length;i++){
                        colNames[i] = md.getColumnLabel(i+1);
                        }
        int nColumns = md.getColumnCount();

/*Pravimo vektor rezultata upita*/
                        while(rezultat1.next()){
```

```
                    Vector rowData = new Vector();
                        for(int i=1;i<=nColumns;i++){
                        rowData.addElement(rezultat1.getObject(i));

                    }
                    dataSet.addElement(rowData);
                    }

/*Pravimo string-matricu rezultata*/
                    int nRows = dataSet.size();
                    String[][] rowDataS = new String[nRows][colNames.length];
                    for(int i=0;i<nRows;i++){
                    Vector row = (Vector)dataSet.elementAt(i);
                    for(int j=0;j<row.size();j++)
                    rowDataS[i][j]=((Object)row.elementAt(j)).toString();
                    }

/*Formiramo tabelu*/
                    JTable table1 = new JTable(rowDataS,colNames);
            //Vracamo vrednost tabele
            return(table1);

        }
        catch(Exception e)
            {
                System.out.println("dzemo");
            return null;
            }
            finally
            {
                try
                {
                    if (baza.naredba1 != null)
                        baza.naredba1.close();
                    if (baza.dbConn != null)
                        baza.dbConn.close();
                }
                catch (SQLException sqle)
                {
        System.out.println("SQLIzuzetak za vreme close(): " +
                        sqle.getMessage());
```

```
                    }
                }

}
```

Metoda za listanje

```
public String pregledBaze()
        {
        // Formiramo tabelu
        table=createTable();
        // Formiramo elemente panela
        p1 = new JPanel();
        p2 = new JPanel();
        p0 = new JPanel();

        slika1.setIcon(new ImageIcon("picture2.jpg"));

        p2.add(slika1);
        p1 = new JPanel();
        lbl1 = new JLabel("Izaberi pol");
        p1.add(lbl1);
        String[] Pol = {"-----","Muski","Zenski"};
        cb1 = new JComboBox(Pol);
        p1.add(cb1);
        panel = new JScrollPane(p1);
        tableScroller = new JScrollPane(table);
JSplitPane splitter = new
JSplitPane(JSplitPane.HORIZONTAL_SPLIT,panel,tableScroller);

splitter.setDividerLocation(100);
   p0.add(splitter);
   izvrsi=new JButton("izvrsi");
   izvrsi.addActionListener(new ButtonListener());
   p1.add(izvrsi);

BorderLayout bl = new BorderLayout();
                this.setLayout(bl);

this.add(p1, BorderLayout.SOUTH);
this.add(p2, BorderLayout.NORTH);
this.add(p0, BorderLayout.CENTER);
```

```
return "Uspeno izvrsen upit";
            }
JLabel slika1=new JLabel();

 class ButtonListener implements ActionListener{
            public void actionPerformed(ActionEvent event){
                        if(event.getActionCommand().equals("izvrsi") )
{
    if (cb1.getSelectedIndex() != 0)
            uslov2 =" AND pol.naziv_pola=" + """+ cb1.getSelectedItem().toString()
            + """;
       else
     uslov2="";
          // Brisemo staru tabelu i postavljamo novu
     JViewport viewport = tableScroller.getViewport();
                        viewport.remove(table);
             JTable table2=createTable();
            viewport.add(table2);}
            }

   }
}
```

4 DEO

OBJEKTNO ORIJENTISANA ANALIZA I DIZAJN - UML MODELOVANJE

4 UML

Objektno orijentisana analiza i dizajn omogućava svim učesnicima u razvoju aplikacije da na jednostavan i sveobuhvatan način steknu uvid u analizu i implementaciju konkretnog problema. Shodno tome, javila se potreba za univerzalnim jezikom namenjenim za objektno-orjentisano modelovanje. Rezultat je UML (Unified Modeling Language) univerzalni jezik za modelovanje koji služi za specifikaciju, vizuelizaciju, konstrukciju i dokumentaciju razvoja sistema. Koristi se u različitim fazama razvoja, od specifikacije zahteva do testiranja završenih, gotovih sistema.

UML je sredstvo modeliranja koje je upotrebljivo i za čoveka i za mašinu i da se uspostavi eksplicitna veza između koncepata i izvršnog koda.Takodje cilj je da se predstavi kompletan sistem, ne samo softverski deo, korišćenjem objektno-orjentisanih koncepata. Analiza i rešavanje problema na ovakav način ima mnoge prednosti.

UML standard koji se primenjuje kod objektno orijentisanog pristupa predviđa odgovarajuće poglede na sistem, s tim što se u svakom pogledu sistem može opisati sa statičkog (strukturnog) i dinamičkog aspekta. Koristi se za konstrukcija software-a kod koga treba odraditi plan ,nudi mogućnost vizualizacije u više dimenzija i nivoa detalja i prikladan je za nadogradnju nasljeđenih, starih sistema. Generalna klasifikacija standardnih UML dijagrama:

***Statički** :*

- Dijagram slučajeva korišćenja (*Use-Case Diagrams*)
- Dijagrami klasa i objekata (Class & Object Diagrams)

***Dinamički** :*

- Dijagrami ponašanja (*Behavior*):
    - dijagrami stanja (*Statechart Diagrams)*
    - dijagrami aktivnosti (*Activity Diagrams*)
- Dijagrami interakcije (*Interaction*):
    - dijagram sekvenci (*Sequence Diagram*)
    - dijagram saradnje (*Collaboration Diagram*)

Fizički :

- Dijagrami implementacije (*Implementation*):
    - dijagram komponenti (*Component Diagram*)
    - dijagram realizacije (*Deployment Diagram*)

Za izradu UML dijagrama korišćeni su:

- **STAR UML i**
- **Visual Paradigm for UML**.

STAR UML je alat za kreiranje klasnih dijagrama i ostalih vrsti dijagrama u UML-u. STAR UML je alat za kreiranje raznih vrsti dijagrama u UML-U. On omogućava automatsko generisanje JAVA koda na osnovu dijagrama i obrnuto , generisanje dijagrama u skladu sa određenim JAVA kodom. Dobijeni kod će samo trebati da se učini malo funkcionalnijim u tom smistu što će morati da se dopiše" kod" koji predstavlja akcije između klasa.

Visual Paradigm for UML- je alat koji podržava UML-a 2.1 .Pored modeliranja i UML-podrške, obezbeđuje modeliranje poslovnih procesa (BPMN), objektno-relacionog mapiranja generatora za Javu, NET i PHP-a.

Prednosti su: mogućnost generisanja koda iz klase modela, a takođe i relacione strukture baze podataka (tabele) . Generisani kod se sastoji od klase zasnovane na dijagramima klasa i kontrolera klase koji pruža čitanje, pisanje i brisanje funkcije za baze podataka.
Još neki od primera UML razvojnih alata su: Rational Rose, Microsoft Visio, Borland Together, Appolo for Eclipse, Enterprise Arhitect, Eclipse Uml2Tools, ArgoUML, MonoUML, Rational Software Architect, MagicDraw UML, PowerDesigner, Visible Analyst for UML, MetaUML, UmlDesigner itd.

ZADACI:

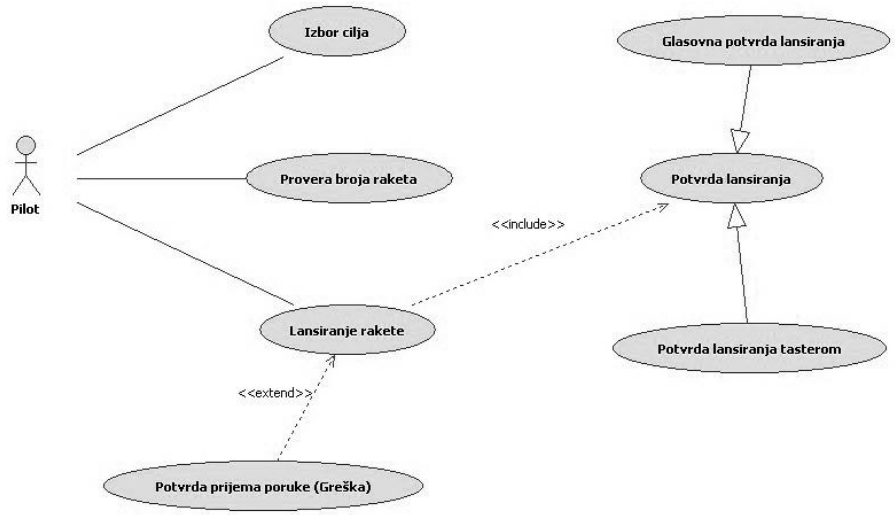
4.1 Statički dijagrami

U objektno- orijentisanom pristupu razvoja sistema osnovni elementi pomoću kojih se opisuje sistem su klase i veze između njih, te objekti posmatranih klasa, njihove veze i poruke koje objekti međusobno razmenjuju u cilju izvršavanja određenih aktivnosti. Dijagram klasa i dijagram objekata su dijagrami pomoću kojih se predstavlja statička struktura. Dijagram klasa prikazuje skup klasa, interfejsa i saradnji, kao i njihovih veza. Dijagrami klasa su najvažniji i najviše korišćeni UML dijagrami i predstavljaju statičku strukturu sistema.

1. Prikazati dijagram slučaja korišćenja koji realizuje se softverski sistem za kontrolu lansiranja raketa. Pilot može izvršiti izbor cilja, lansiranje rakete, proveru broja raketa. Svako lansiranje rakete potrebno je da potvrdi, pri čemu

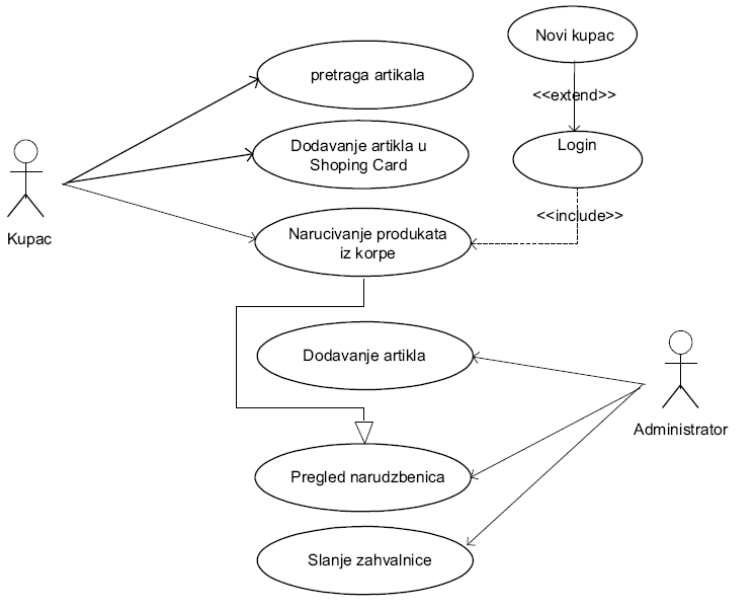
se potvrda može obaviti odgovarajućom glasnom komandom ili pritiskanjem odgovarajućeg tastera na upravljačkoj konzoli. Ukoliko se tokom lansiranja desi kvar na sistemu za lansiranje pilot će biti obavešten o ovome događaju i potrebno je da potvrdi prijem poruke.

REŠENJE:



2. Slučajevi korišćenja za web aplikaciju online prodavnice:

REŠENJE:



3. Nacrtati **dijagram klasa** koji opisuje deo poslovnog sistema jedne kompanije. Svi radnici kompanije se mogu podeliti u dve grupe:

- prvu, koju čine zaposleni u administraciji (A) i
- drugu, koju čine članovi projektnih timova tj. zaposleni koji rade na reklamnim kampanjama (C).

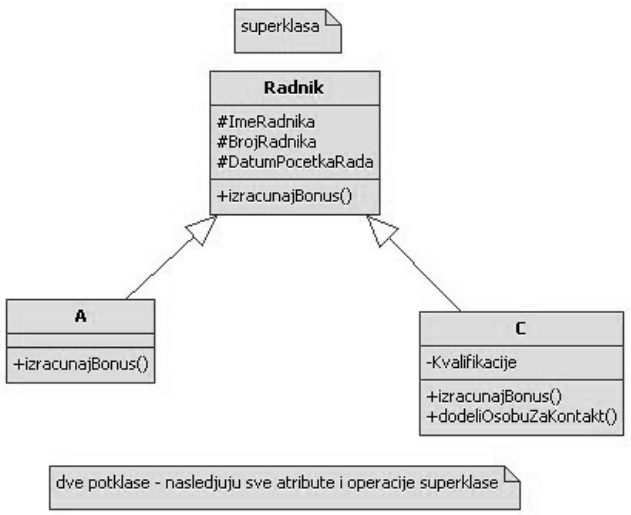
Za sve zaposlene neophodno je zapamtiti osnovne podatke tj. ***ime i prezime, jedinstveni broj radnika kao i datum početka rada***. Jedna od razlika zaposlenih je računanje godišnjih bonusa:

- bonus kod C se računa na osnovu profita kampanja na kojima su radili
- bonus kod A se računa na osnovu prosečnog profita svih kampanja u prethodnoj godini.

Razlike među zaposlenima su još i da:

- je kod C neophodno zapamtiti kvalifikacije
- C može biti osoba za kontakt sa klijentom
- A se ne može dodeliti ni jednoj kampanji

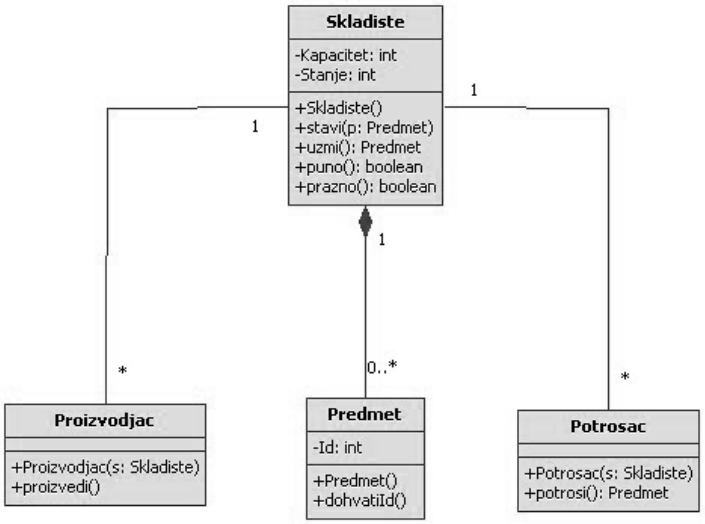
REŠENJE:



4. Nacrtati **dijagram klasa** na jeziku UML sledećeg sistema klasa:

Predmet *- ima jedinstven, automatski generisan ID,*
Skladište *- može da sadrži zadati broj predmeta,predmeti mogu da se stavljaju i uzimaju jedan po jedan.*
Proizvođač *- može da napravi jedan predmet i da ga stavi u skladište, koje se zadaje prilikom stvaranja proizvođača,*
Potrošač *- može da uzme jedan predmet iz skladišta, koje se zadaje prilikom stvaranja potrošača.*

REŠENJE:



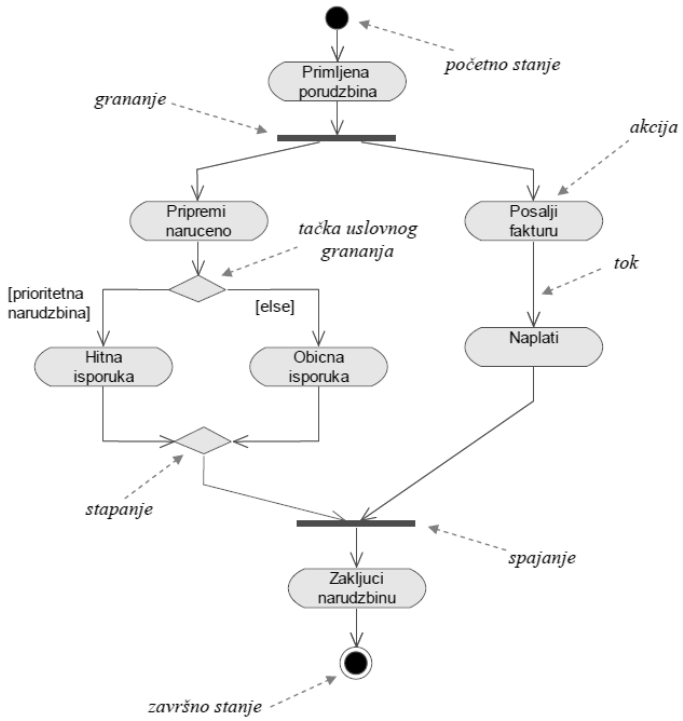
4.2 Dinamički dijagrami

Dinamičkim modelom opisuju se dinamičke karakteristike sistema, odnosno pojedinačno ponašanje objekata u sistemu, kao i ponašanje kompletnog sistema koji se modelira. Tokovi kontrole i podataka ne mogu biti odgovarajuće shvaćeni razmatrajući samo pojedinačne klase sistema izolovano. Mora se opisati na koji način objekti u sistemu međusobno komuniciraju u cilju ostvarivanja očekivanog posmatranja.

Za razmatranje složenog ponašanja sistema pojedine pojave se moraju posmatrati u sklopu celine u cilju razmatranja funkcionisanja sistema. Ovde su predstavljeni dijagrami za prikaz dinamičkih karakteristika sistema: dijagrami sekvenci, saradnje, stanja, aktivnosti.

5. Koristeći **dijagram aktivnosti** modelovati proces koji opisuje tok posla od trenutka primanja do trenutka zaključivanja narudžbine u jednoj trgovinskoj kompaniji. Dijagram treba da obuhvati akcije kao što su: primi porudžbinu, pripremi naručeno, pošalji fakturu,zaključi narudžbinu...

REŠENJE:



6. Realizovati dva **dijagrama stanja** :

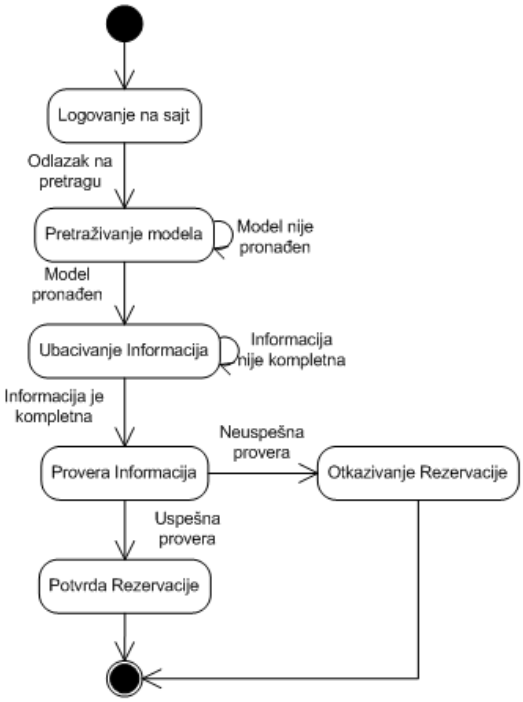
Prvi dijagram treba da pokazuje deo sistema vezan za rezervaciju računara.

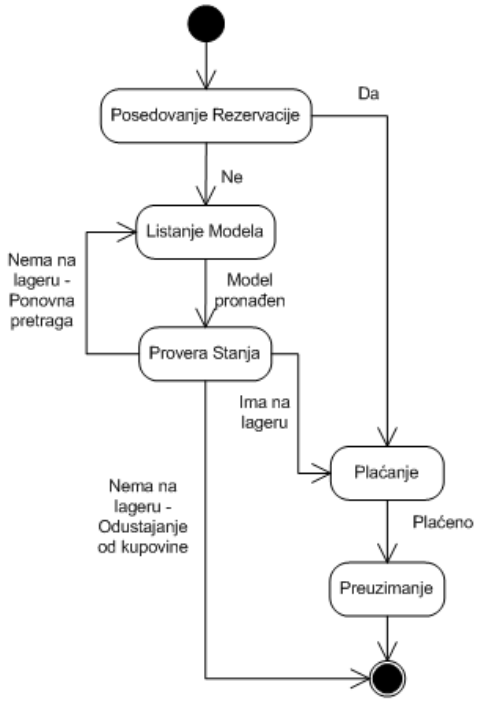
- Ubačena je i mogućnost otkazivanja rezervacije zbog neuspešno unete informacije o kupcu.

Deo sistema vezan za kupovinu prikazan je na drugom dijagramu. Ovde je ubačena i mogućnost posedovanje rezervacije, prilikom koje se prelazi direktno na plaćanje.

- Ukoliko se ne poseduju rezervacije, vrši se normalni tok kupovine.

REŠENJE:

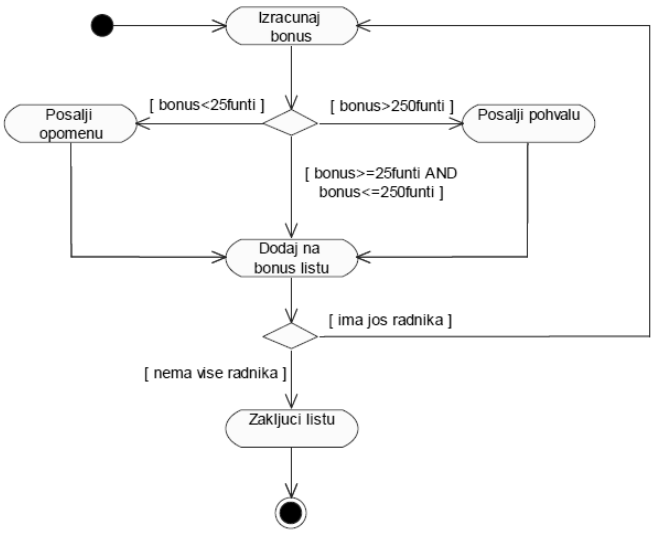




7. Koristeći **dijagram aktivnosti** opisati operaciju pripreme bonus liste u kompaniji. Da bi se bonus lista formirala neophodno je prvo izračunati bonus svakog radnika a zatim:

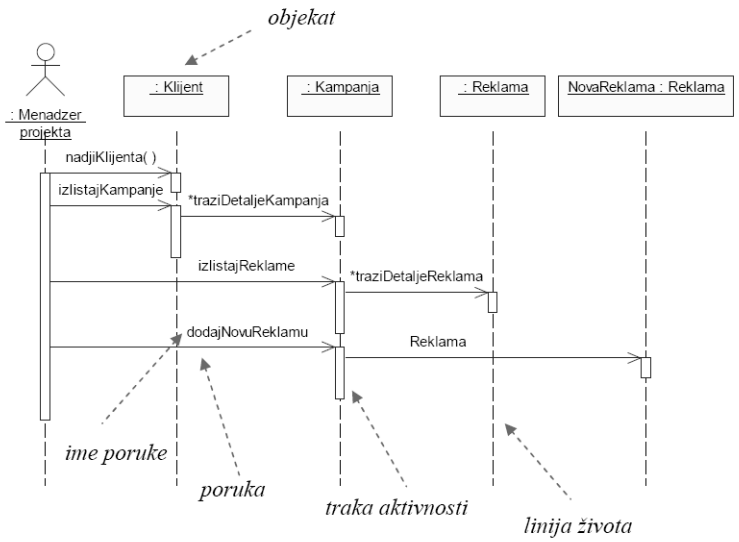
- ako je izračunati bonus manji od 25 funti napisati opomenu radniku I dodati ga na bonus listu;
- ako je izračunati bonus veći od 250 funti napisati pohvalu radniku i dodati ga na bonus listu;
- ako je izračunati bonus između 25 i 250 funti samo dodati radnika na bonus listu.

REŠENJE:



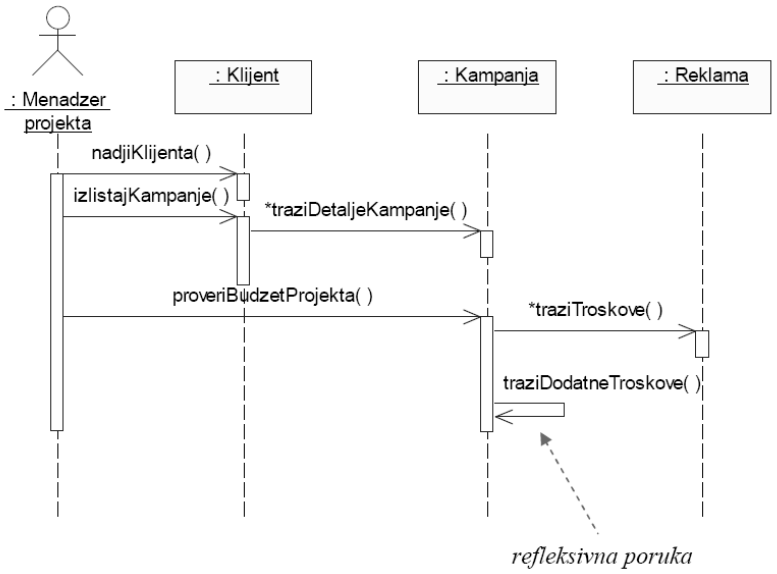
8. Nacrtati **dijagram sekvence** za slučaj korišćenja - *Dodaj nov način reklame*. Ovaj slučaj korišćenja obuhvata podatke o svim načinima relamiranja koji se koriste u okviru datog projekta. Menadžer projekta treba najpre da nađe odgovarajućeg klijenta, zatim traženu kampanju tog klijenta, a zatim na osnovu detaljnih informacija o svim reklamama korišćenim u okviru tražene kampanje, menadžer projekta dodaje novu reklamu.

REŠENJE:

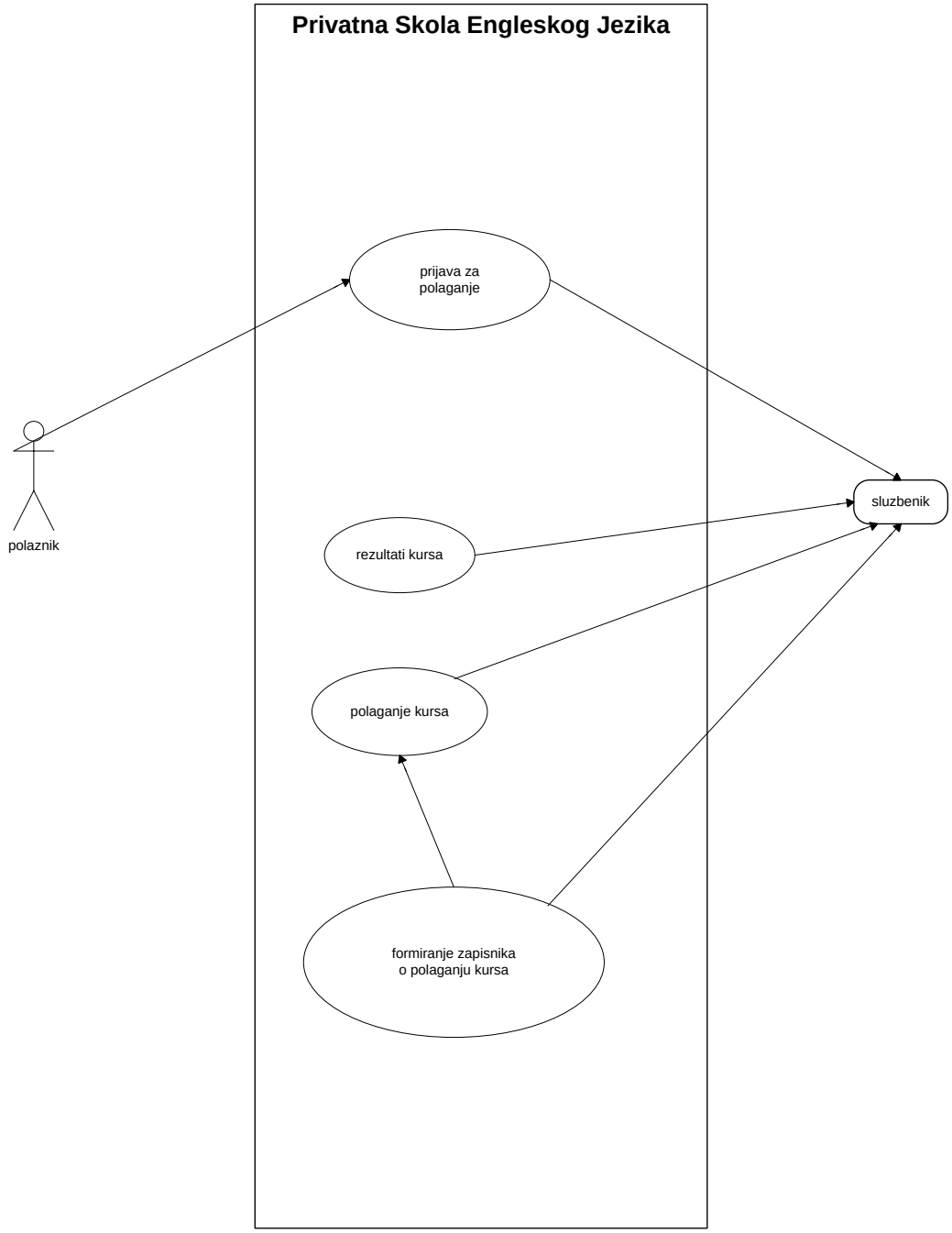


9. Nacrtati **dijagram sekvence** za slučaj korišćenja - *Proveri budžet projekta*. U okviru ovog slučaja korišćenja vrši se provera da li je došlo do prekoračenja budžeta ili ne. Menadžer projekta treba najpre da nađe odgovarajućeg klijenta, zatim traženu kampanju tog klijenta, a zatim da proveri da li je došlo do prekoračenja budžeta ili ne. Ukupni troškovi projekta se računaju kao suma troškova svake reklame plus svi ostali, dodatni troškovi projekta.

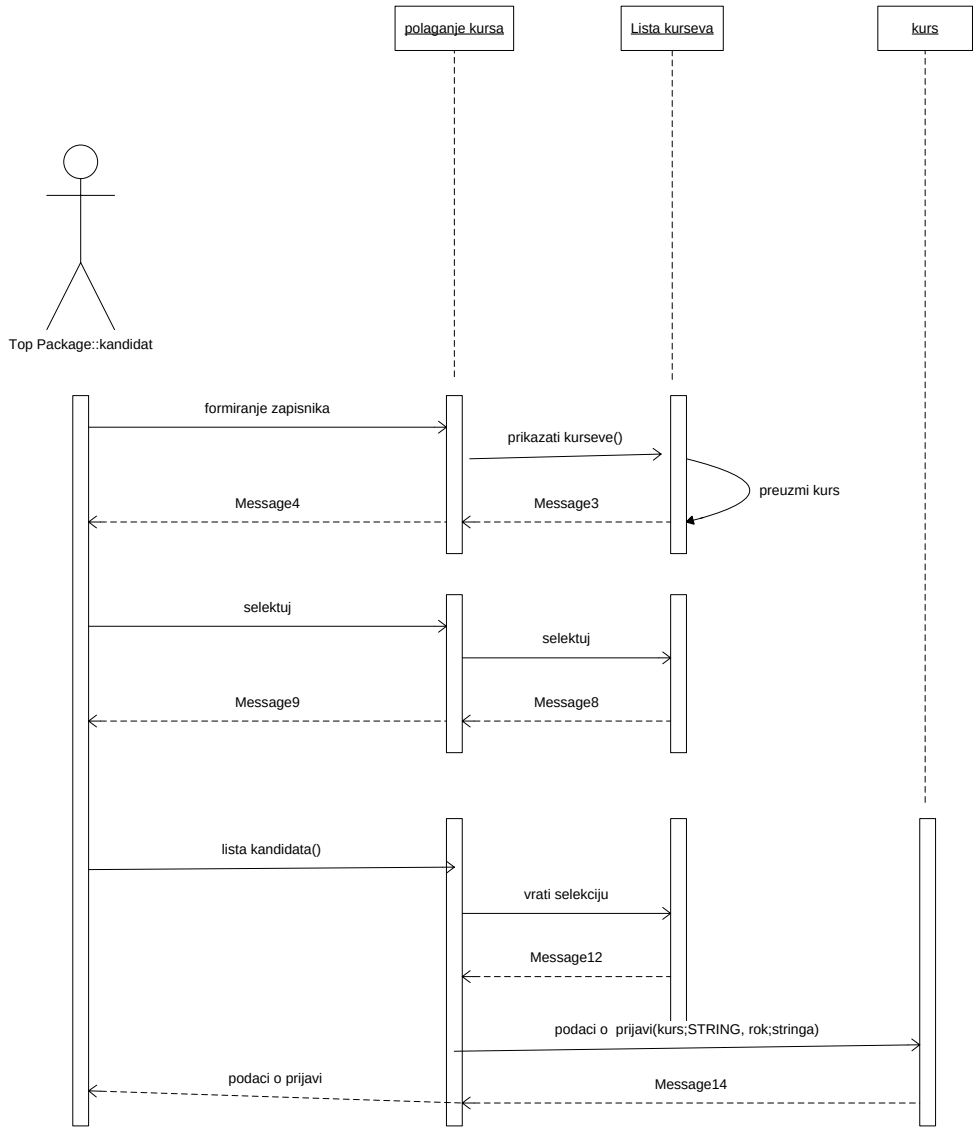
REŠENJE:



Slede dijagrami slučajeva korišćenja i dijagram sekvenci koji su rađeni za informacioni sistem koji je dat u prilogu (Dodatak A).

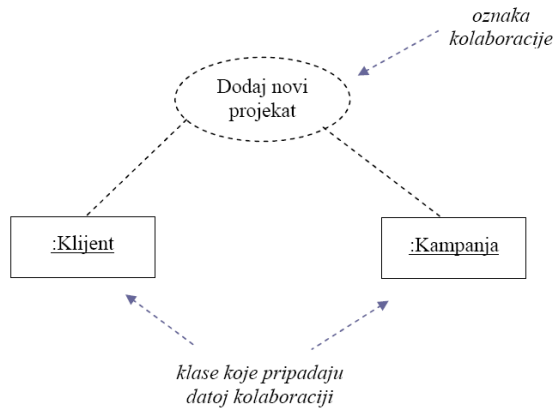


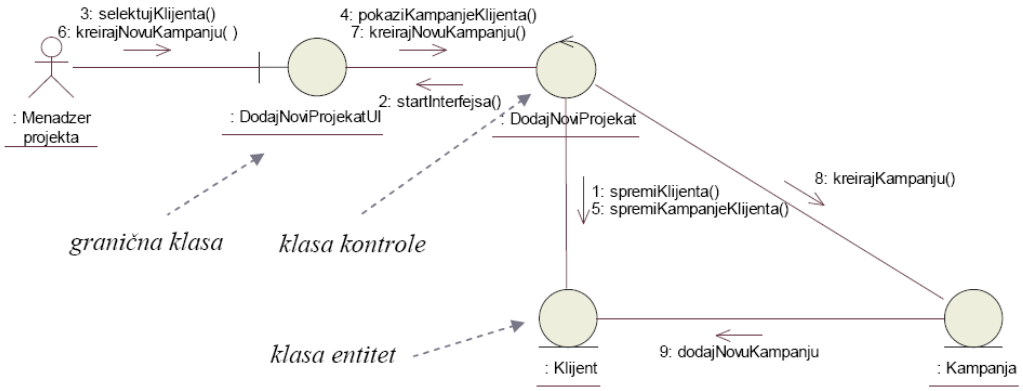
Dijagram za prijavu kursa

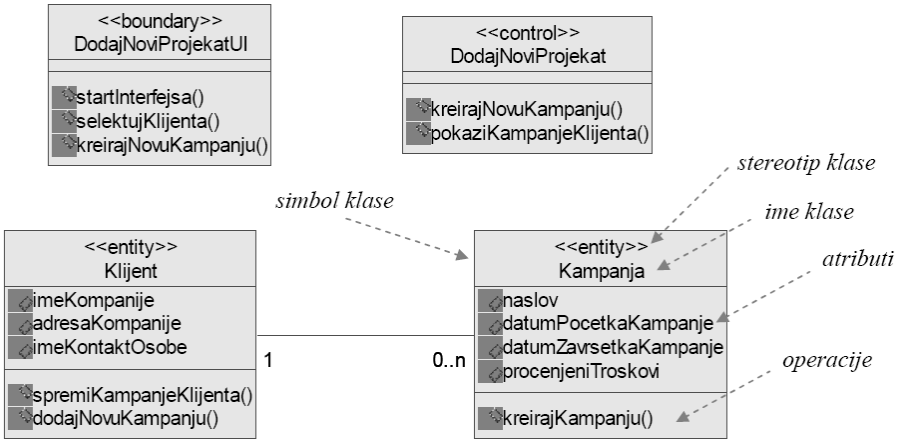


Dijagram za prijavu kursa

10. Nacrtati **dijagram kolaboracije** i dijagrama klase za slučaj korišćenja - *Dodaj novi projekat*. Slučaj korišćenja Dodaj novi projekat opisuje unos podataka o novom projektu (kampanji), uključujući procenjene troškove kao i pretpostavljeni datum završetka projekta. Menadžer projekta vrši unos ovih podataka.

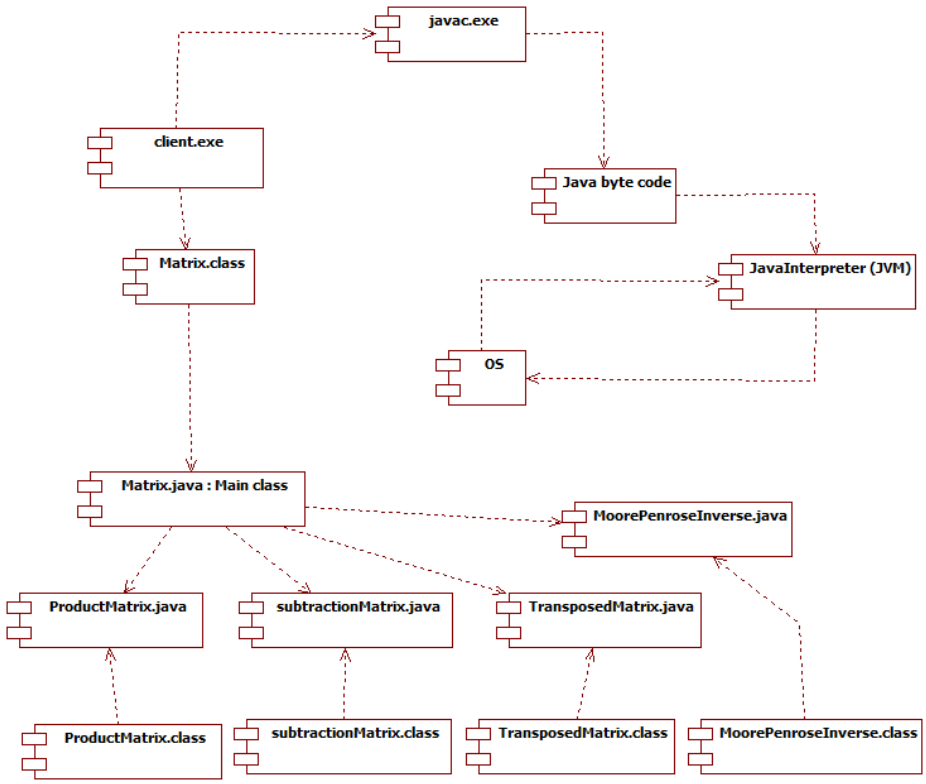

oznaka
kolaboracije
Dodaj novi
projekat
:Klijent
:Kampanja
klase koje pripadaju
datoj kolaboraciji


3: selektujKlijenta()
6: kreirajNovuKampanju( )
4: pokaziKampanjeKlijenta()
7: kreirajNovuKampanju()
2: startInterfejsa()
: Menadzer
projekta
: DodajNoviProjekatUI
: DodajNoviProjekat
1: spremiKlijenta()
5: spremiKampanjeKlijenta()
8: kreirajKampanju()
granična klasa
klasa kontrole
klasa entitet
9: dodajNovuKampanju
: Klijent
: Kampanja


<<boundary>>
DodajNoviProjekatUI
startInterfejsa()
selektujKlijenta()
kreirajNovuKampanju()
<<control>>
DodajNoviProjekat
kreirajNovuKampanju()
pokaziKampanjeKlijenta()
stereotip klase
simbol klase
ime klase
atributi
operacije
<<entity>>
Klijent
imeKompanije
adresaKompanije
imeKontaktOsobe
spremiKampanjeKlijenta()
dodajNovuKampanju()
1
0..n
<<entity>>
Kampanja
naslov
datumPocetkaKampanje
datumZavrsetkaKampanje
procenjeniTroskovi
kreirajKampanju()

4.3 Fizicki model sistema

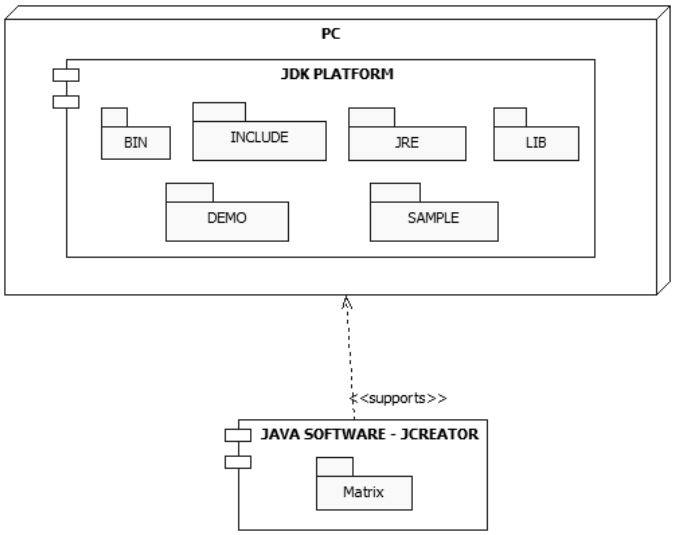
Dijagrami komponenti *(component diagram)* prikazuju strukturu komponenti sistema, opisuju zavisnost komponenti ili interfejsa komponenti sistema. Komponente dijagrama su izvorni kodovi, biblioteke, dinamičke komponente (.dll), izvršni programi (.exe).



4.1: *Dijagram komponenti*

Dat je postupak kompajliranja i izvršavanja Java programa. Java kompajler konvertuje Java izvorni kod u bajt programski kod, bajt programski kod se interpretira pomoću JVM, jer on sadrži naredbe koje prepoznaje JVM i koje preslikava u naredbe konkrektnog OS.

Dijagram realizacije ili razvoja *(deployment diagram)* prikazuje hardversku strukturu sistema, i takodje daje akcenat na vezu izmedju hardverskih i softverskih komponenata.



4.2: *Dijagram realizacije*

Program napisan u Javi se zapravo ne izvršava direktno na računaru, već u okviru standardizovanog okruženja po imenu Java 2 Platforma koje se u vidu softvera implementira na čitavom nizu računara i operativnih sistema.
Platforma Java ima dve komponente – softversku implementaciju hipotetičkog računara po imenu Java virtuelna mašina (*Java Virtual Machine*, JVM) i Javin programski interfejs aplikacije (*Java Application Programming Interface*, Java API) koji predstavlja skup softverskih komponenti koje obezbeđuju sve ono što je neophodno za pisanje zaokruženih interaktivnih aplikacija u Javi.

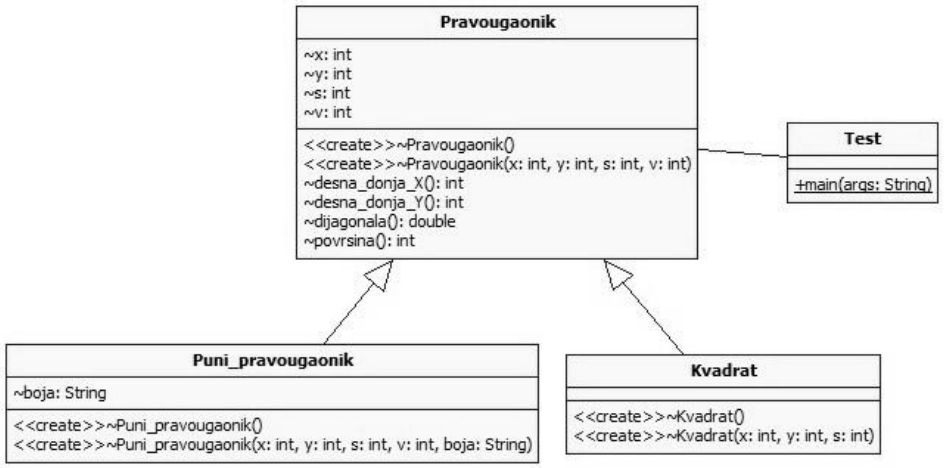
4.4 Generisanje Java izvornog koda iz UML dijagrama i obrnuto

Postoji mogućnost generisanja koda iz UML dijagrama klasa u java izvorni kod. To se može odraditi u STAR UML-u ili u Visual Paradigm-u. STAR UML je alat za kreiranje raznih vrsti dijagrama u UML-U. On omogućava automatsko generisanje JAVA koda na osnovu dijagrama I obrnuto, generisanje dijagrama u skladu sa određenim JAVA kodom.Postupak kod STAR UML-a je sledeći:

Da biste generisali JAVA kod , idite na tools na meniju pri vrhu, izaberite JAVA , pa Generate Code. Iz dialog box-a, izaberite model, verovatno nazvan model1 i idite na next.

• Izaberite opciju select all i idite na next
• Izaberite zeljeni output direktorijum i idite na next
• U Option setup obavezno čekirajte Generate the Documentation by JavaDoc i Generate empty JavaDoc. Ostale opcije ne treba čekirati
• STAR UML će nakon ovog generisati kod na osnovu naših dijagrama. Za izlazak idite na FINISH
• Možete u skladu sa gore opisanim da dopunite kod da biste dodati željene i potrebne funkcionalnosti

Pogledajte klasu Pravougaonik sa klasama naslednicama (kvadrat i puni_Pravougaonik). Navedene klase smo radili u odeljcima – *klase i objekti i nasleđivanje.* Sledi dijagram klasa iz kog se može izgenerisati struktura:



ZADACI – test br.1

1. Prikazati **<u>dijagram slučaja korišćenja</u>** koji realizuje softverski sistem za **simulaciju razvoja softvera**. Korisnik može izvršiti sledeci izbor: **dizajn sistema, projektovanje i implementacija**. Za izbor faze **dizajn sistema** potrebno je potvrditi izbor metode za razvoj softvera pri čemu se potvrda može obaviti sledecim metoda:

- *Model vodopada*
- *V model*
- *Fazni razvoj (inkrementalni i iterativni)*
- *Prototipski model*

Ukoliko se tokom procesa desi greska korisnik će biti obavešten o ovom dogadjaju i potrebno je da potvrdi prijem poruke.

2.Realizovati **<u>dijagram klasa</u>** *za sledeci kod:*

```
class Racunari{
String Artikal;
int Cena;
int Nabavna_cena;
double Porez;
int Kategorija;
//metode
        void Provera_Kategorija(){...}
        void Ispis(){...}
        void Snizenje(){...}      }
class Hardware extends Racunari{
String Brend;
int Sifra;
//metode
        int Provera_Sifre(){...}
        void Promena_PDV(){...} }
class Veleprodaja extends Racunari{
int Kolicina;
//metode
        int Popust(){...}
        void Ispis_Veleprodaja(){...} }
```

Nivo pristupa:

Atributi protected

Nivo pristupa:

Atributi private

Nivo pristupa:

Atributi protected

Dodati klasu magacin koja je u vezi kompozicije (jaka agregacija) sa klasom Racunari.

3. Koristeći **dijagram aktivnosti** modelovati proces kreiranja tekstualnog dokumenta.

Dijagram treba da obuhvata akcije kao što su: **otvaranje i zatvaranje** programa obradu teksta**,**

unos teksta, **snimanje** dokumenta... Ako je potrebno,prilikom unosa teksta moguće je u dokument
ubaciti tabelu koristeći poseban program za **kreiranje tabela**. Na kraju je neophodno **odštampati** kreirani dokument.

*4.*Dat je kod u Javi. Realizovati dijagram stanja (*postoji 6 stanja).

```
class state1 {
void e1( ) {
if (broj < 100 ) {
        if (a < 0 ) {
        ac.setState(ac.s2State);}
        else if (a < 5) {
        ac.setState(ac.s3State); }
else
        ac.setState(ac.s4State);}
}
else {
      if (a>105 && a<125){
      ac.setState(ac.s5State);}
      else if (a>126 && a<150){
      ac.setState(ac.s6State);}
} }
```

*5.*Nacrtati **dijagram sekvence** za slučaj korišćenja - provera kredita klijenta**.**

U okviru ovog slučaja korišćenja vrši se provera da li je došlo do prekoračenja odgovarajuce sume ili ne. Operater u korisnickom servisu treba najpre da uspostavi vezu sa odgovarajucem klijentom, zatim traženu tarifu tog klijenta, a zatim da proveri da li je došlo do prekoračenja ili ne. Ako je doslo onda se racunanje sume vrsi *standardno* a ako ne onda se koristi *popust metoda*.

Ukupna suma se računa kao suma fiksne pretplate svakog meseca plus svi ostali, dodatni troškovi klijenta. (klase koje treba navesti su : klijent, tarifa, rac_metoda, popust_metoda)

ZADACI – test br.2

1.Realizovati **use case diagram**. Menadzer projekta ima sledece nadleznosti:
a) da izabere ucesnike projekta
b) da dodaje nov nacin reklame
c) da proveri budzet projekta.
Prilikom provere budzeta projekta se moze javiti potreba za stampanjem izvestaja i stampanje zavrsnog racuna. Slucaj Nadji kampanju ukljucuje sve navedene aktivnosti koje obavlja menadzer projekta.

2.Realizovati **dijagram klasa** za sledeci kod:

```
class Student{
//atributi
        String ime;
        String prezime;
        String JMBG;
        String broj_indexa;
        int godina_studija;
        String smer;
        String odsek;
        String fakultet;
//operacije
        void Fakultet(){}
        void Brucos(){}
        void veca_godina(){}
        void ispisi_Pedagogija(){
        void ispis(){}
}
class Student_Informatika extends Student{
        double prosek;
        void ispis_iznad9(){ }
        }
class Test{
        public static void main(String [] Args){}
}
```

Dodati klasu student_Matematicar koja je u vezi kompozicije (jaka agregacija) sa klasom Student.Svi atributi su protected,a metode private.

3. **Koristeći dijagram aktivnosti** opisati operaciju pripreme bonus liste u kompaniji.

Da bi se bonus lista formirala neophodno je prvo izračunati bonus svakog radnika a zatim:

- ako je izračunati bonus manji od 25 000 din. napisati opomenu radniku i dodati ga na bonus listu;
- ako je izračunati bonus veći od 30 000 din. dodati stimulaciju radniku i dodati ga na bonus listu;
- ako je izračunati bonus između 25 000 i 30 000 din. samo dodati radnika na bonus listu.

4. Nacrtati dijagram stanja za klasu **Radnik** kojom se opsiju životni ciklus radnika od trenutka zaposlenja do napuštanja posla.

Prilikom zaposlenja svakom zaposlenom se dodeljuje koeficijent rada na osnovu koga se, kasnije, računa plata. Godinu dana posle prestanka radnog odnosa podaci o radniku se arhiviraju.

Dok je radnik u radnom odnosu on može ili ne mora biti član projektnog tima i istovrmeno može ili ne mora biti osoba za kontakt sa klijentom.

5. Nacrtati dijagram sekvence za slučaj korišćenja **Dodaj nov način polaganja testa**.

Ovaj slučaj korišćenja obuhvata podatke o svim načinima polaganja koji se koriste u okviru projekta polaganja nekog testa za neki kurs.

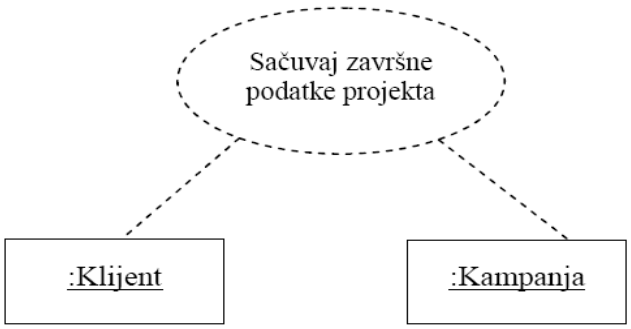
Rukovodilac projekta treba najpre da nađe odgovarajućeg kandidata, zatim traženi nacin polaganja za tog kandidata , a zatim na osnovu detaljnih informacija o svim nacinima polaganjima korišćenim u okviru projekta, rukovodilac dodaje novi nacin polaganja.

ZADACI – test br.3

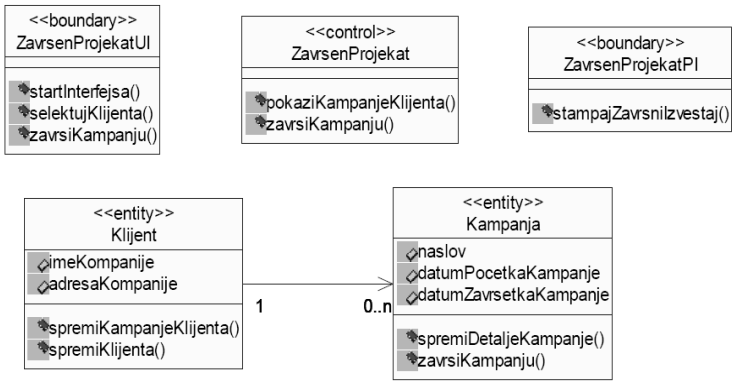
1. Odraditi dijagram saradnje za slučaj korišćenja ***Sačuvaj završne podatke projekta***. Slučaj korišćenja Sačuvaj završne podatke projekta treba da, kada se projekat završi, izvrši unos stvarnih troškova projekta i zabeleži datum završetka projekta.

Na osnovu ovoga štampa se izveštaj koji se koristi za formiranje završnih računa koje treba naplatiti od klijenata. Menadžer projekta je odgovoran za ovaj slučaj korišćenja.

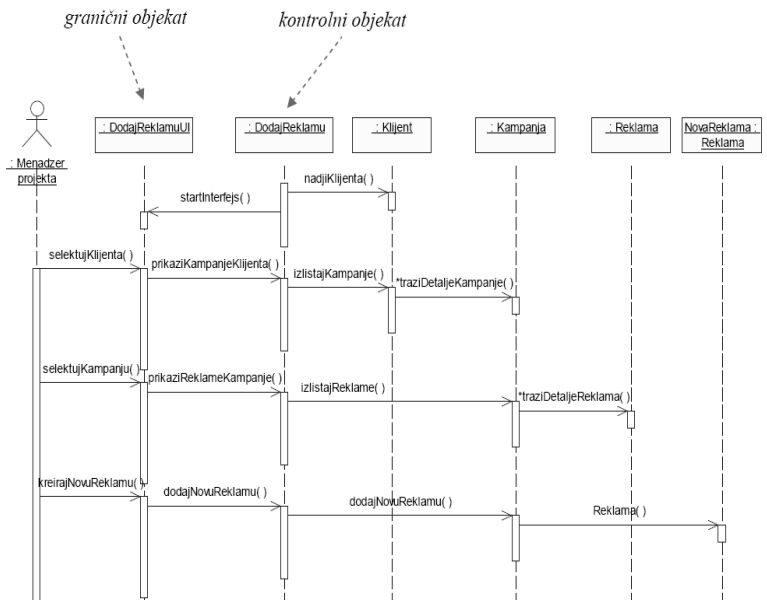
Dati su pomoćni dijagrami:



Dijagram klasa za slučaj korišćenja Proveri budžet projekta.



2.Dat je dijagram sekvenci. Konvertovati navedeni dijagram u dijagram saradnje:



3. Za slučaj upotrebe „Upis godine“ u IS studentske službe.

Učesnik u ovom slučaju upotrebe je radnik studentske službe koji pokreće formu IS (klasa Forma) kako bi izvršio upis određenog studenta u narednu godinu.

Od radnika se traži da unese broj indeksa za studenta koga želi da upiše u narednu godinu. Nakon unosa broja indeksa, Sistem vrši proveru da li postoji student u bazi podataka.

Ukoliko student ne postoji u bazi podataka, radniku se prosleđuje poruka o grešci. Ukoliko student postoji u bazi, Sistem traži u bazi podataka podatke o položenim ispitima za datog studenta, nakon čega vrši proveru broja nepoloženih ispita.

Ukoliko je broj nepoloženih ispita veći od 2, sistem vraća poruku korisniku o nepostojanju uslova za upis naredne godine.

Ukoliko broj nepoloženih ispita nije veći od 2, Sistem upisuje studenta u narednu godinu u bazi podataka, a korisniku se vraća poruka o tome.

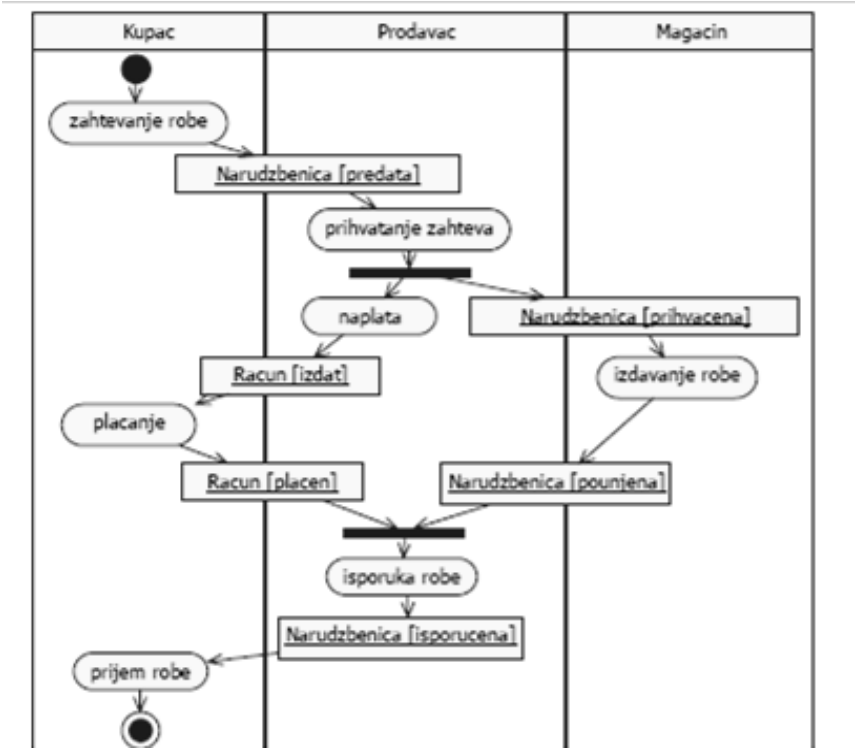
Osim toga, ukoliko je student položio sve ispite pre 01.09. u bazu podataka student se automatski upisuje na budžet, a radniku se šalje potvrda o tome.

Odraditi dijagram sekvence za navedeni primer.

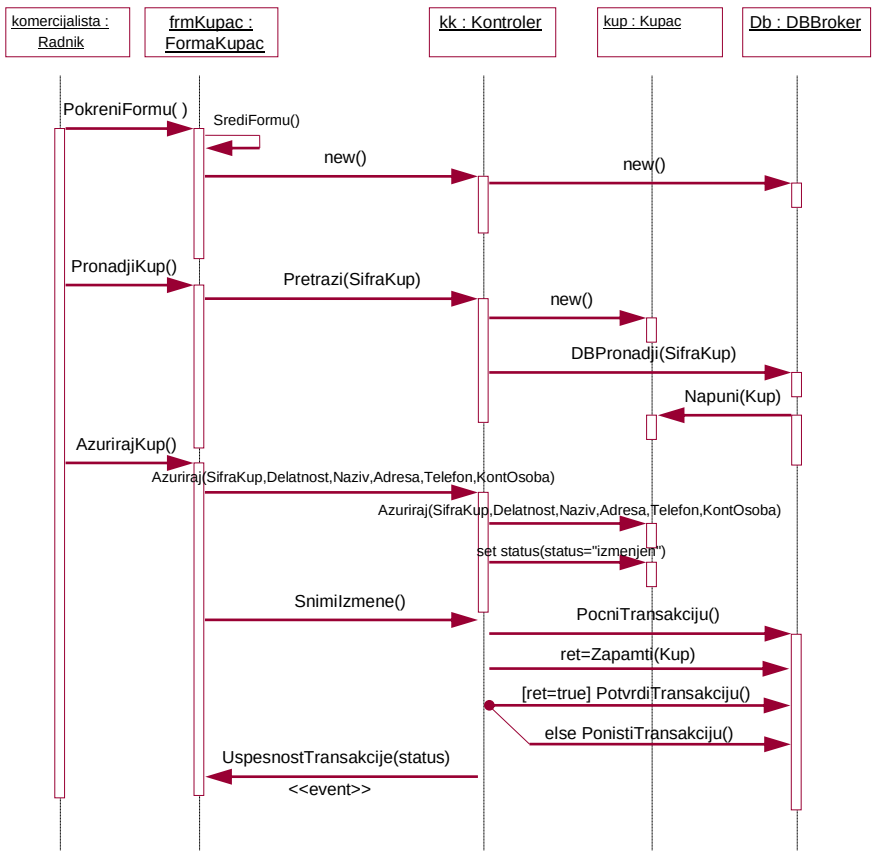
4.Za primer iz zadatka br.3 odraditi dijagram kolaboracije (dijagram komunikacije) .

ZADACI – test br.4

1.Naveden je dijagram aktivnosti.Na osnovu njega odraditi dijagram saradnje. Klase su Kupac, Prodavac i Magacin. Metode koje se razmenjuju izmedju njih su akcije koje su date na dole prikazanom dijagramu.



2.Dat je dijagram sekvenci. Konvertovati navedeni dijagram u dijagram saradnje:



3.Odraditi dijagram sekvenci na osnovu opisa za kupovinu proizvoda iz aparata:

- Kupac ubaci novčić u aparat.
- Kupac napravi odabir.
- Novac putuje do blagajne.
- Blagajna proverava da li ima proizvoda u aparatu.
- Ako nema proizvoda, aparat vraća novac ili nudi kupcu izbor nekog drugog proizvoda i ponovi postupak provere proizvoda.
- Inače, blagajna proverava da vidi da li je kupac ubacio iznos koji odgovara iznosu proizvoda.
- Ako je ubačeni iznos veći nego što je cena proizvoda, blagajna izračunava i proverava rezerve.
- Ako ima dovoljno novaca u rezervi, onda se korisniku vraća novac.
- Ako nema dovoljno novaca u rezervi, onda se korisniku vraća celokupni iznos.
- Ako se ubaci nedovoljni iznos, aparat ne radi ništa već čeka da se ubaci još novaca.
- Inače, vraća proizvod.

4.Napraviti dijagram saradnje za simulaciju „upravljanja liftom“ po sledećem opisu:

- Putnik pritisnuo dugme za aktivaciju lifta
- Sistem detektuje da li je pritisnuto dugme za pozivanje
- Lift ide na sprat na kojem je putnik
- Otvaraju se vrata
- Putnik ulazi i izabere dugme za željeni sprat
- Zatvaraju se vrata
- Odlazi na željeni sprat
- Otvaraju se vrata
- Putnik izlazi
- Zatvaraju se vrata lifta

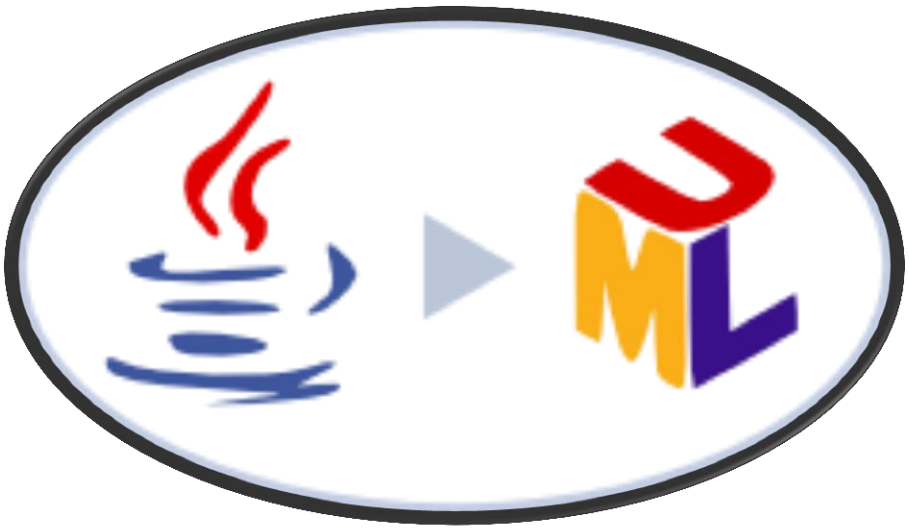
Pretpostavljene klase su : Putnik,Lift,Sistem

KONTROLNA PITANJA

1.Kakav je jezik UML ,navedi njegovu funkciju i koje su prednosti?
2.Šta prikazuje statički model sistema a šta dinamički?
3.Šta predstavlja fizički model sistema i koji su to dijagrami?
4.Navedi sličnosti između dijagrama aktivnosti i dijagrama stanja?
5.Navedi sličnosti između dijagrama sekvenci i dijagrama saradnje?
6.Da li postoji sličnost između dijagrama aktivnosti i dijagrama saradnje?
7.Šta predstavljaju dijagrami klasa?
8.Šta prikazuje tzv.Use case dijagram?
9.Koje su osnovne komponente dijagrama aktivnosti a koje dijagrama stanja?
10.Da li postoji veza između programskog jezika JAVA i UML-a?
11.Kakvi su to dijagrami ponašanja i da li postoji sličnost sa dijagramima interakcije?
12.Navedi redosled izrade dijagrama kod dizajna i analize nekog IS?
13.Koje su karakteristike dijagrama realizacije?
14.Koji su to dijagrami koji su najvažniji i najviše korišćeni i predstavljaju statičku strukturu sistema?
15.Navedi bar 5 UML razvojnih alata?

DODATAK B

PRIMENA UML ALATA U DIZAJNU I ANALIZI INFORMACIONOG SISTEMA



PRIMENA UML ALATA U DIZAJNU I ANALIZI INFORMACIONOG SISTEMA

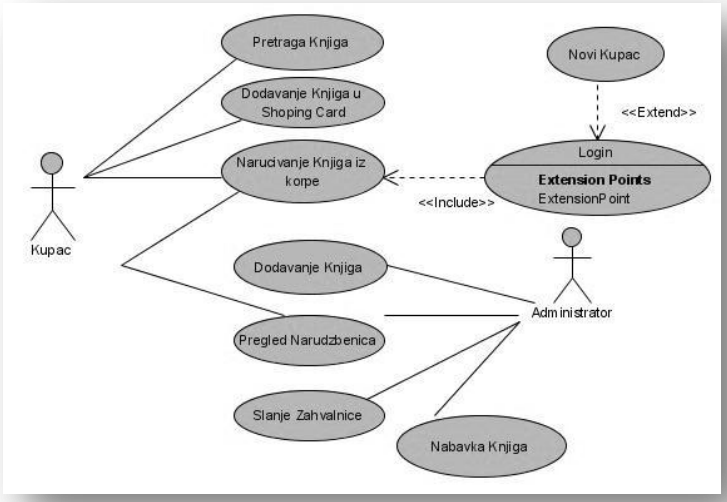
U nastavku predstavljen je modul Web aplikacije baziran na poslovanju On-line Knjižare, sa svim objašnjenim procesima počevši od pretrage, nabavke, brisanja, izmena kao i mogućnost On-line kupovine.

B.1 Postupak izrade

Prilikom modelovanja za ovakve tipove problema, treba najpre odraditi slučajeve korišćenja, od koga dalje poticu svi sledeći dijagrami. Zatim sledi projektni pogled , kod koga se statički aspekt sistema prikazuje preko dijagrami klasa i dijagrami objekata, a dinamički aspekt preko dijagrama interakcije, dijagrami promene stanja i dijagrami aktivnosti. Nakon toga je potrebno odraditi procesni pogled koji prikazuje "niti upravljanja" i procese preko kojih se ostvaruje konkurentnost i sinhronizacija procesa u sistemu. Preko ovoga pogleda prvenstveno se analiziraju performanse, skalabilnost i propusnost sistema. Predstavljaju se sa istom vrstom dijagrama kao i projektni pogled, samo se akcenat stavlja na "aktivne klase" klase koje reprezentuju procese i niti. Sledi implementacioni pogled prikazuje komponente i fajlove sa kojima se sistem realizuje. Statički aspekt se prikazuje dijagramima komponenti, a za dinamički se koriste dijagrami interakcije, dijagrami promene stanja i dijagrami aktivnosti. Konacno, pogled razmeštaja (Deployment view) koji prikazuje sistemsko- hardversku topologiju. Prikazuje se distribucija hardverskih komponenti i instalacija softverskih komponenti na njima. Statički aspekt se opisuje dijagramima razmeštaja, a za dinamički se ponovo koriste dijagrami interakcije, dijagrami promene stanja i dijagrami aktivnosti.

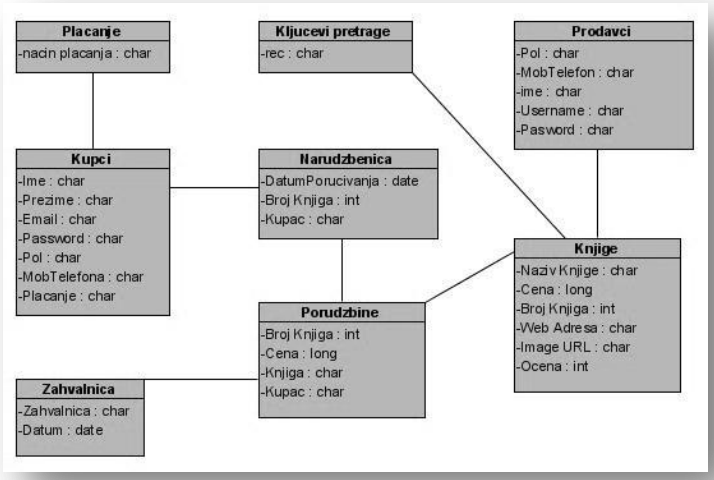
B.2 Statički model sistema

Statičkim modelom opisuju se entiteti sistema. Dijagram klasa i dijagram objekata su dijagrami pomoću kojih se prikazuje statička struktura sistema. Entiteti i veze između njih predstavljeni su klasama i vezama između klasa prikazanim na dijagramu klasa.



Dijagram 1 – Statički dijagram: generalni pogled na sistem

Klasa predstavlja skup atributa i operacija kojima se opisuje struktura i ponašanje objekata posmatrane klase.



Dijagram 2 – dijagram klasa

B.3 Dinamički model sistema

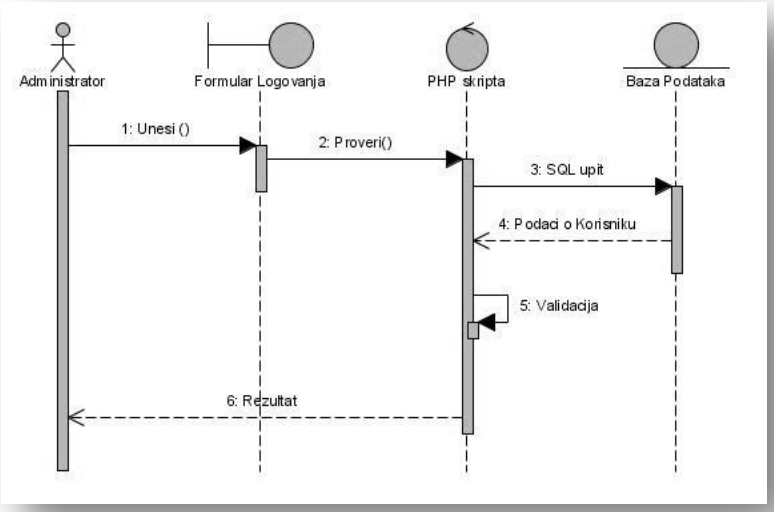
Dinamičkim modelom se opisuju karakteristike sistema, odnosno pojedinačno ponašanje objekata u sistemu, kao i ponašanje kompletnog sistema koji se modelira. Dinamičke karakteristike sistema opisuju se:

- objektima,ulogama koje objekti mogu imati u sistemu,
- vezama koje važe između objekata,
- porukama koje objekti međusobno prosljeđuju i operacijama koje se izvršavaju kao odgovor na primljene poruke,
- stanjima u kojima se objekti mogu naći i promjenama stanja kao odgovor na primljene poruke.

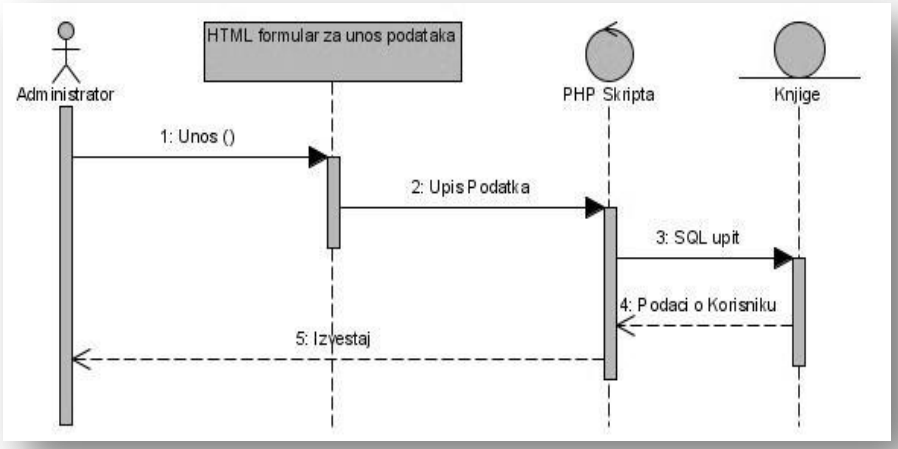
Za predstavljane dinamičkih modela sistema koriste se UML dijagrami, i to: dijagram sekvenci, dijagram saradnje, dijagram stanja i dijagram aktivnosti.

Dijagram sekvenci

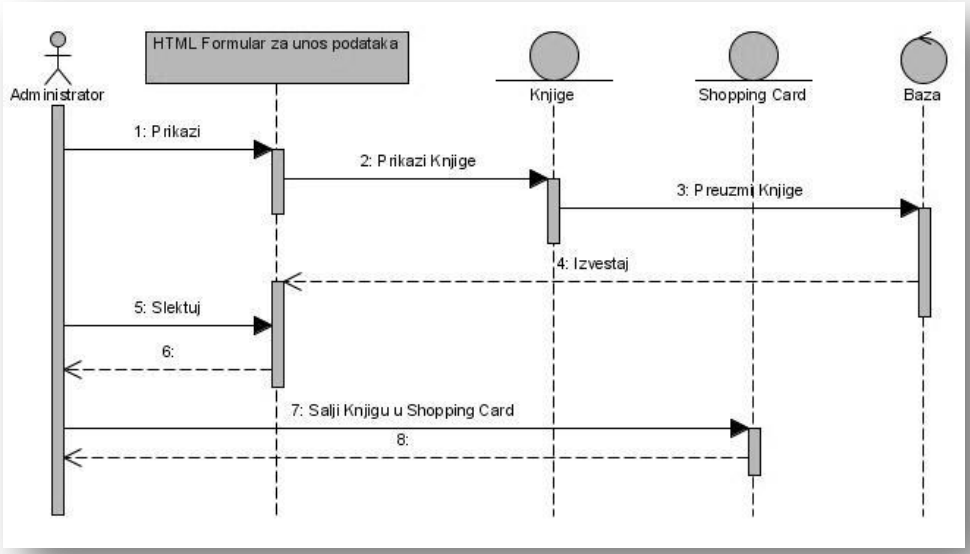
Dijagramom sekvenci moguće je prikazati komunikaciju između skupa objekata, ostvarenu pomoću poruka koje objekti međusobno razmjenjuju u cilju ostvarivanja očekivanog ponašanja.



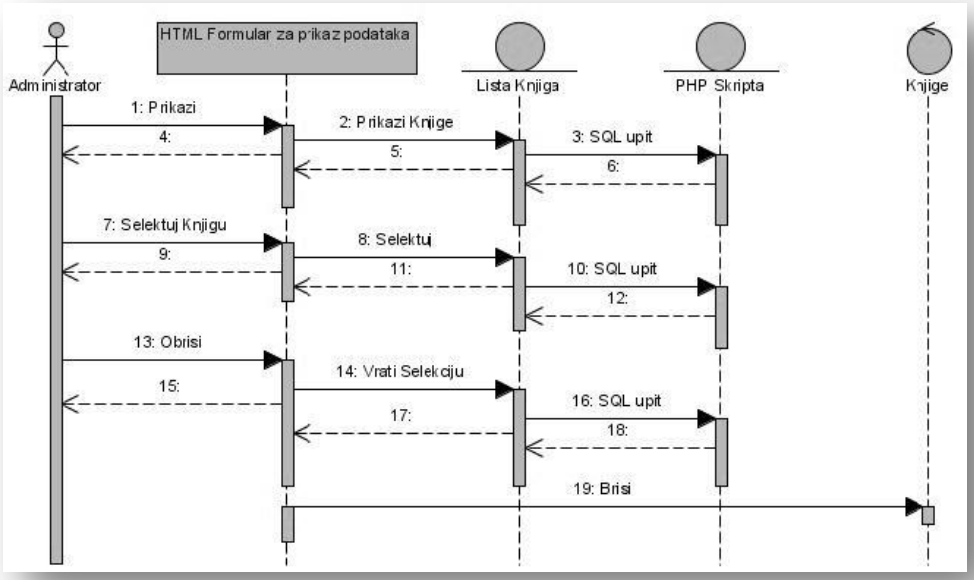
Dijagram 3 – dijagram sekvenci: provera podataka o korisniku



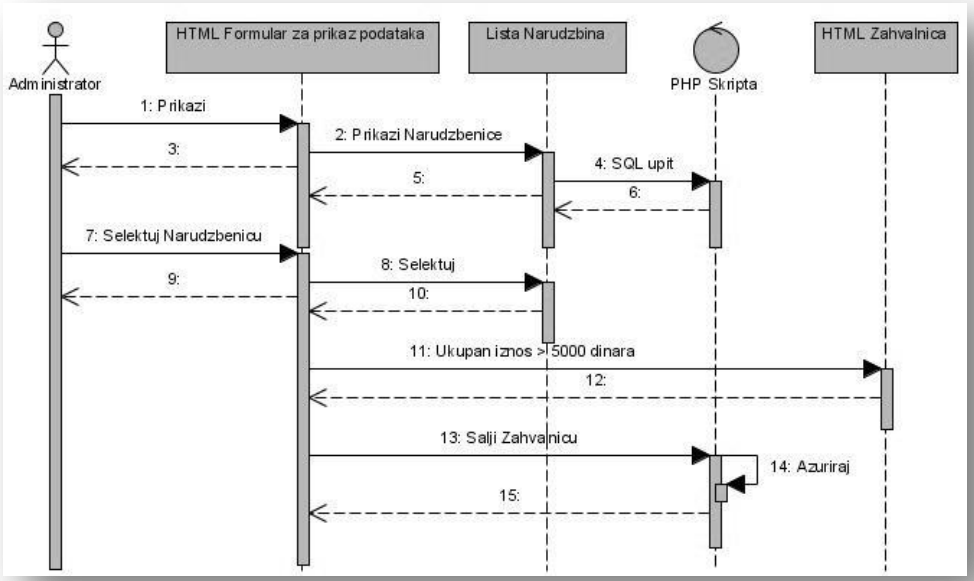
Dijagram 4– dijagram sekvenci: upis podataka o korisniku



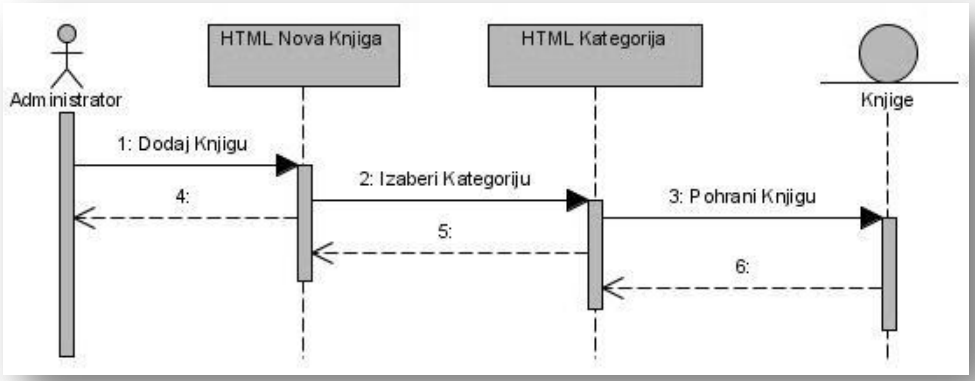
Dijagram 5 – dijagram sekvenci: podaci o knjigama,prikaz,pretraga



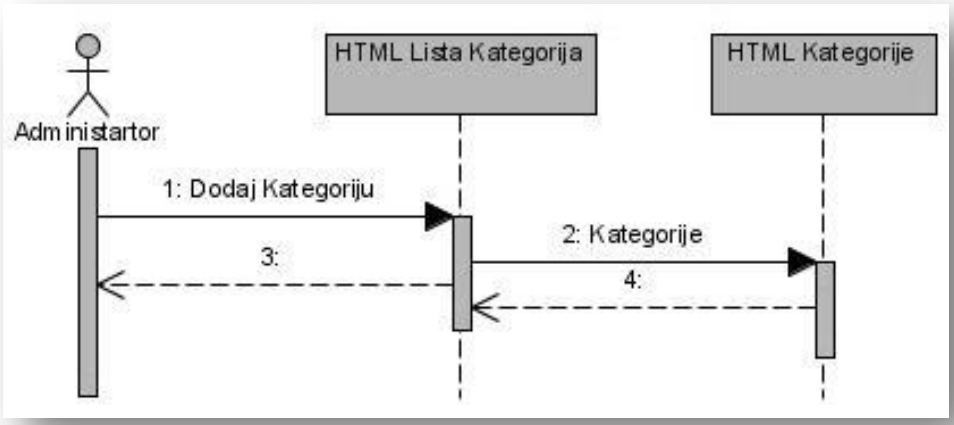
Dijagram 6– dijagram sekvenci: brisanje knjiga



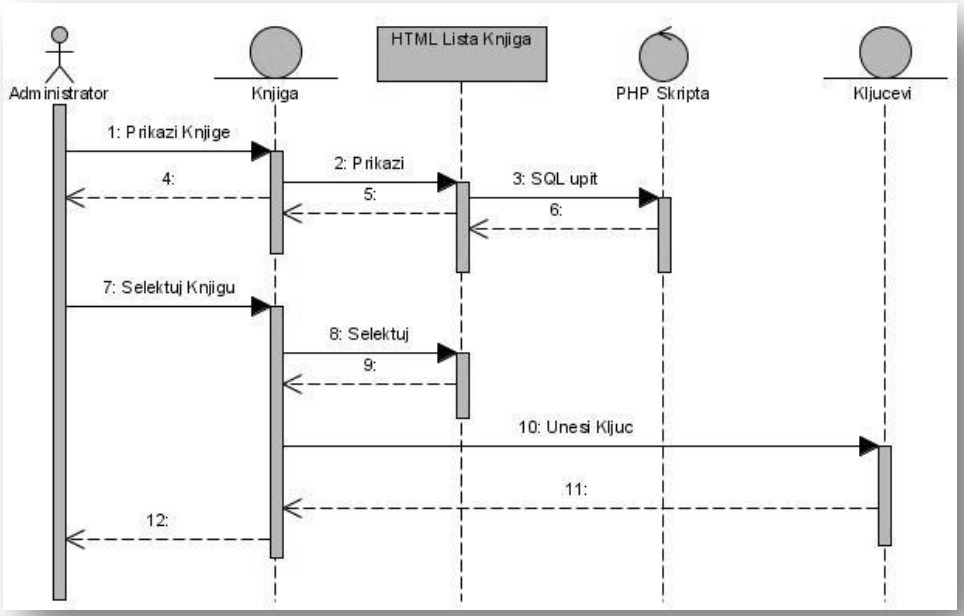
Dijagram 7– dijagram sekvenci: narudzbenica



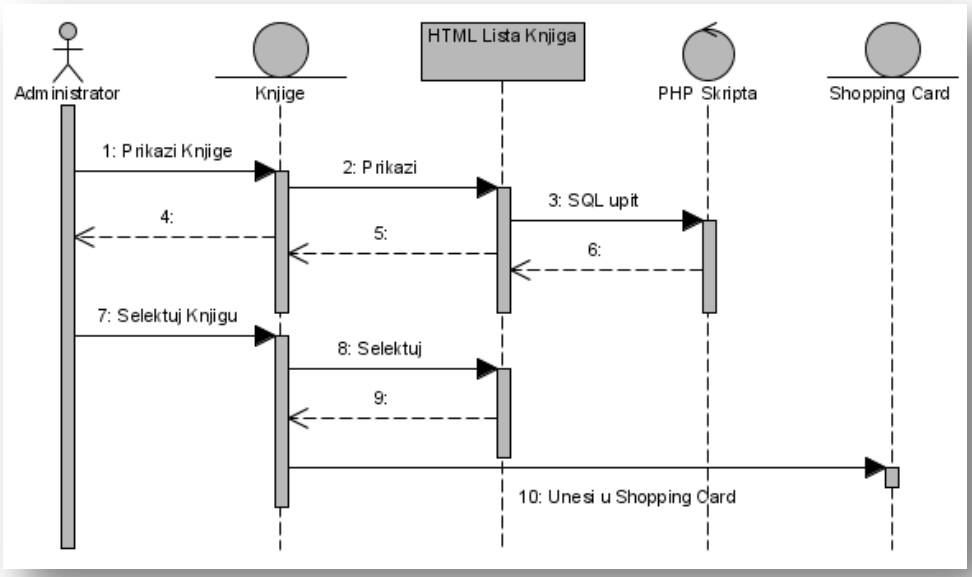
Dijagram 8– dijagram sekvenci: biranje kategorije



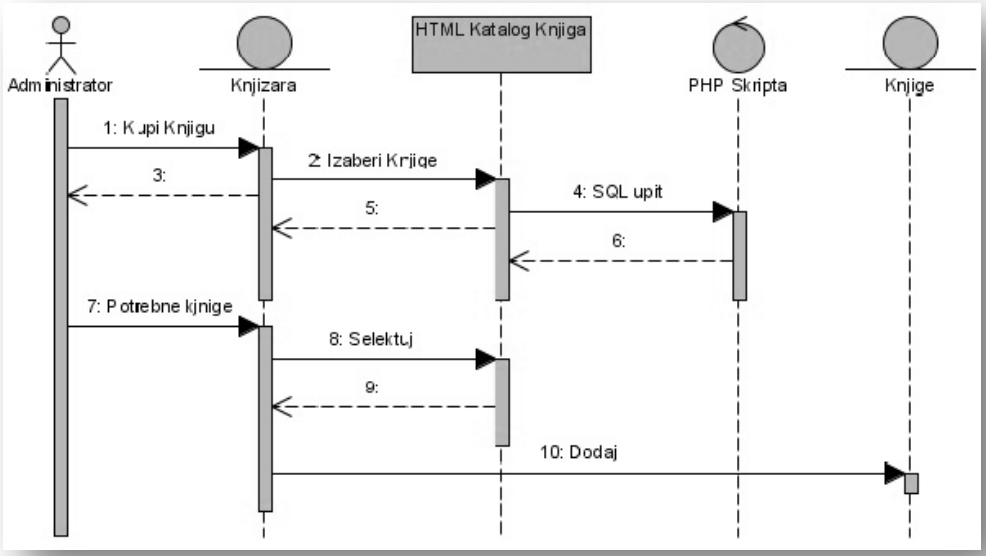
Dijagram 9– dijagram sekvenci: dodavanje kategorije



Dijagram 10– dijagram sekvenci: biranje knjiga

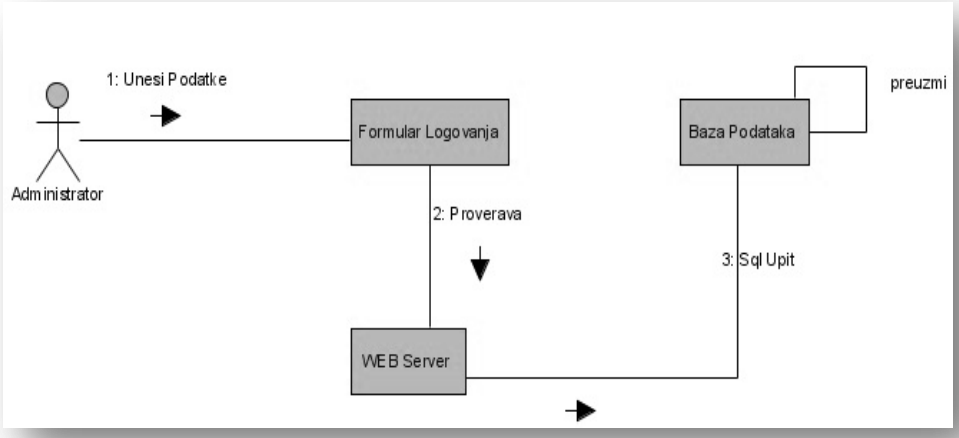


Dijagram 11– dijagram sekvenci: unos knjige u shopping card

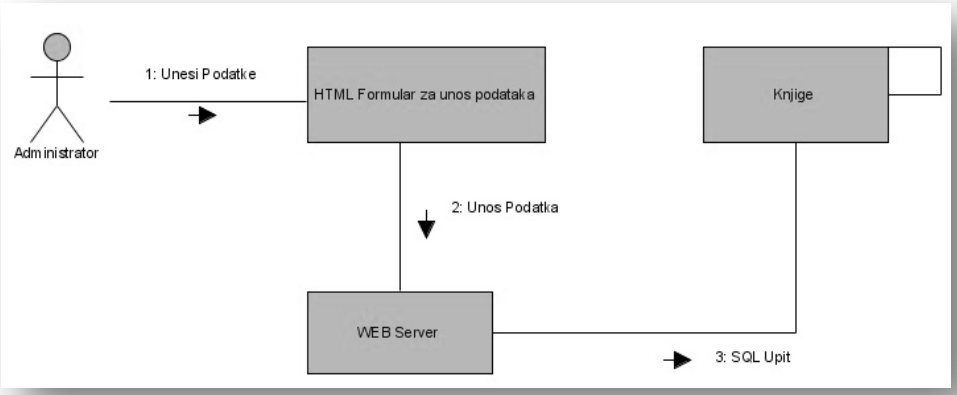


Dijagram 12– dijagram sekvenci: selekcija kataloga knjiga

Slično dijagramu sekvence, **dijagram kolaboracije** opisuje interakciju između objekata kroz niz poruka pri čemu akcenat nije na vremenu, kao kod dijagrama sekvence,već na relacijama između objekata.

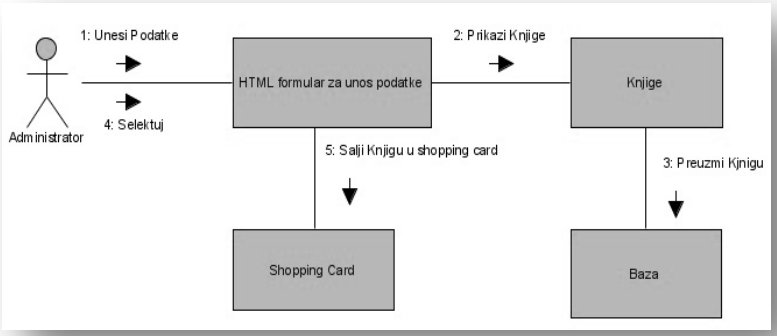


Dijagram 13– dijagram saradnje: logovanje

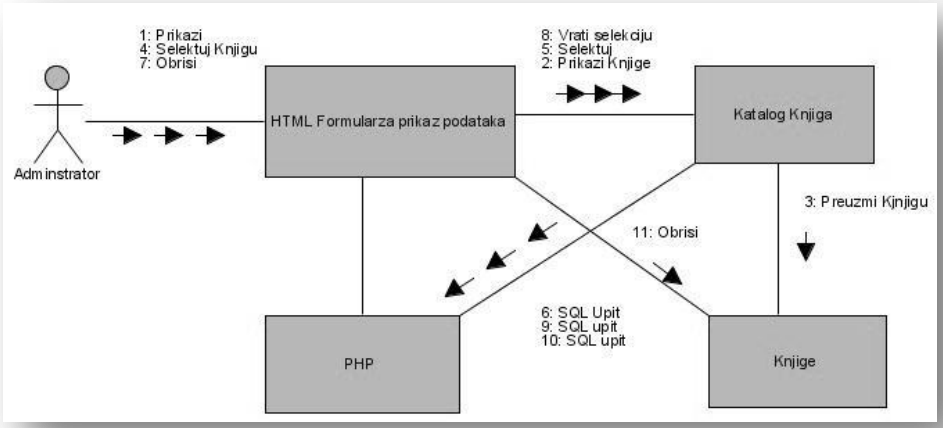


Dijagram 14– dijagram saradnje: unos podataka

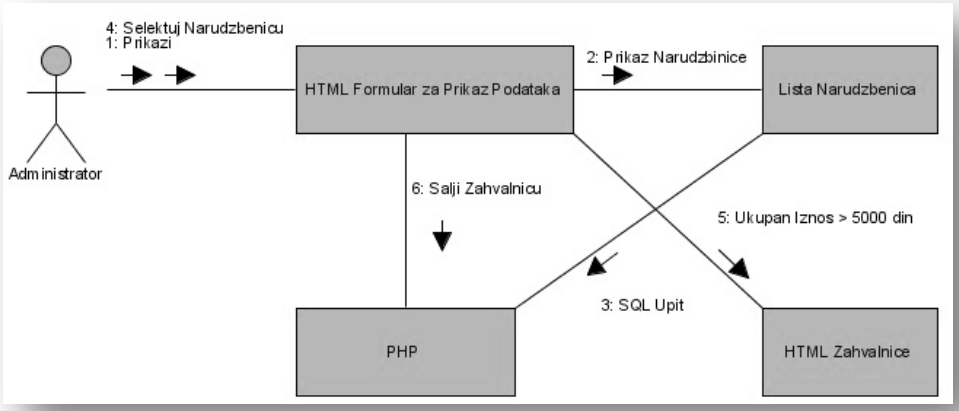
Služe za prikaz složenijih interakcija između objekata i njihove međusobne povezanosti.Fokusira se na kolaboracionoj strukturi i organizaciji između objekata.Modeluje link između objekata. Sastoji se od objekata, linkova, poruka



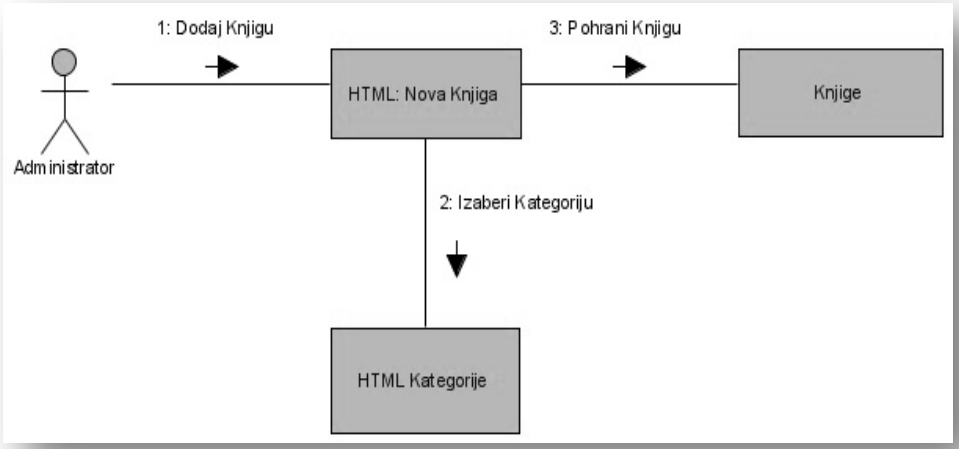
Dijagram 15– dijagram saradnje: prikaz knjiga



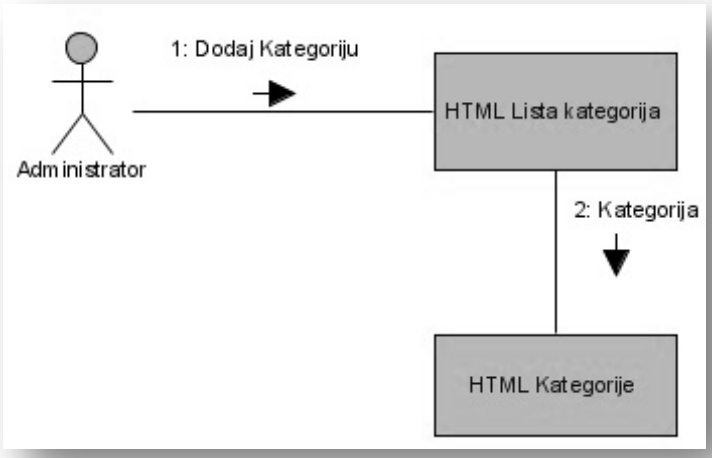
Dijagram 16– dijagram saradnje: brisanje knjiga



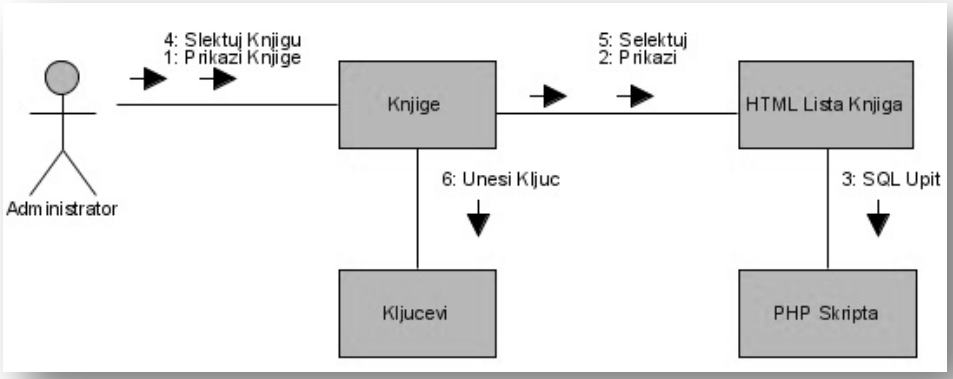
Dijagram 17– dijagram saradnje: narudzbenica



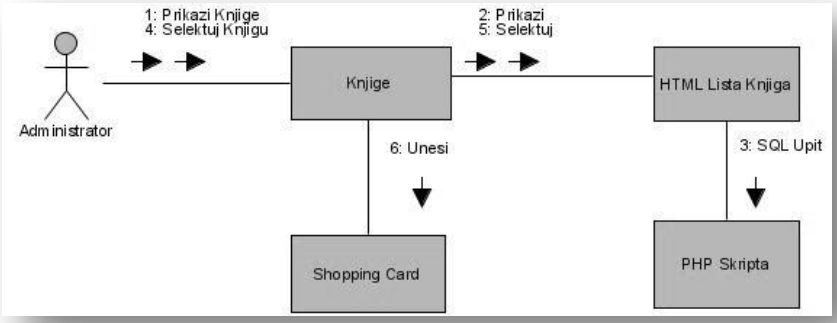
Dijagram 18– dijagram saradnje: biranje kategorije



Dijagram 19– dijagram saradnje: dodavanje kategorije



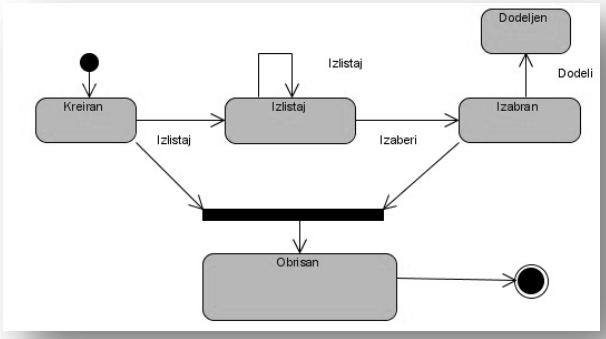
Dijagram 20– dijagram saradnje: selekcija knjiga



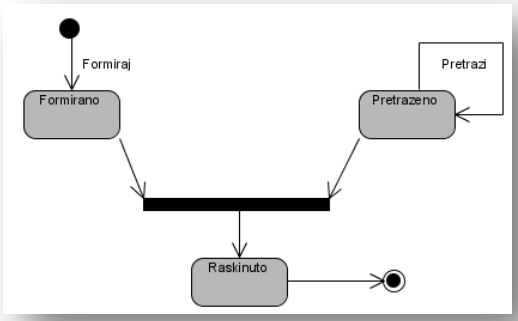
Dijagram 21– dijagram saradnje: smeštanje knjige u shopping card

Dijagrami stanja služe za modelovanje dinamičkih aspekta sistema.

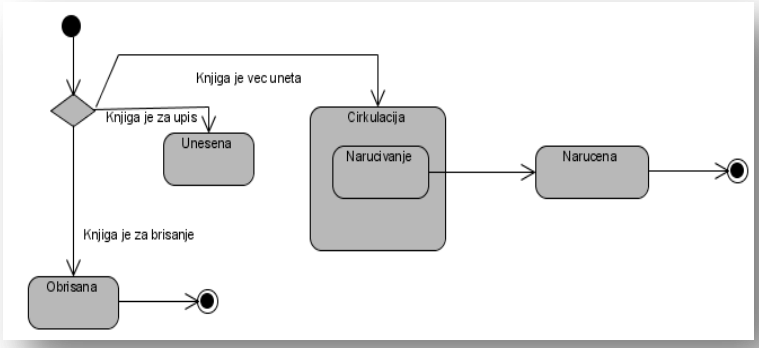
- Stanje je situacija u životu objekta kada on zadovoljava neki uslov. Objekat ostaje u nekom stanju u konačnom vremenskom intervalu.
- Dijagram stanja pokazuje odvijanje upravljanja od stanja do stanja. Obično modeluju objekte koji reaguju.
- Tranzicija pokazuje kretanje od jednog stanja ka drugom.To je relacija između 2 stanja koja kaže da će objekat u jednom stanju izvršiti neke akcije pa će ući u drugo stanje.



Dijagram 22– dijagram stanja: stanja proizvoda



Dijagram 23– dijagram stanja: pretraži

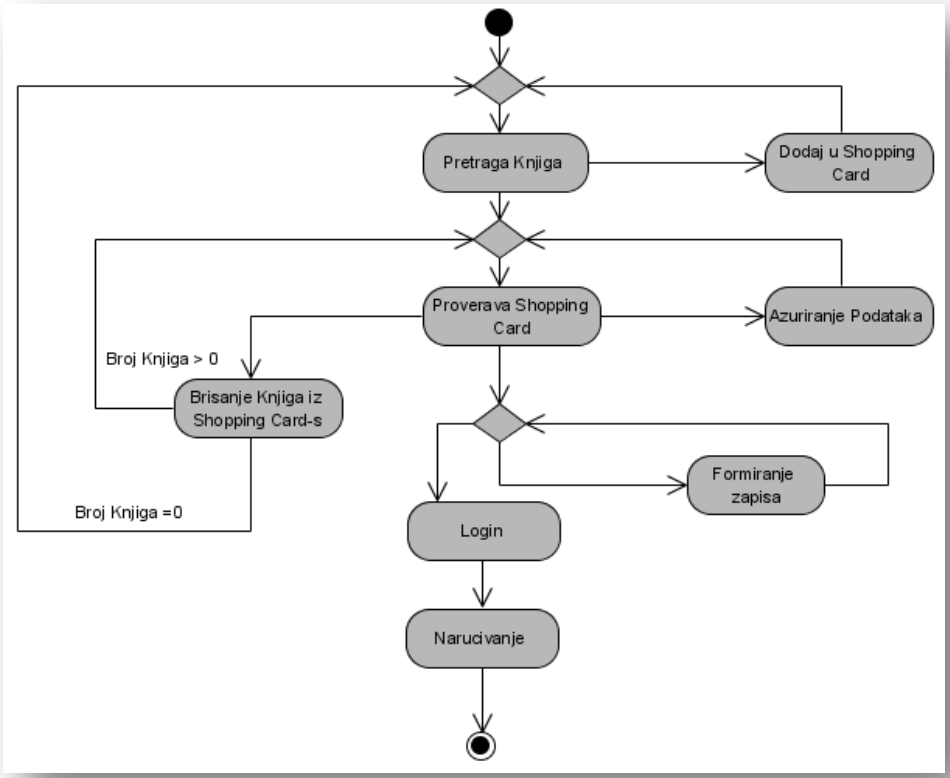


Dijagram 24– dijagram stanja: naruči

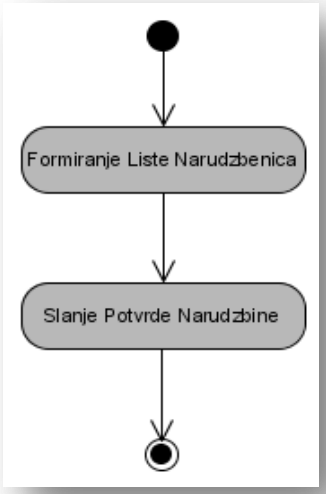
Dijagrami aktivnosti služe za modelovanje dinamičkih aspekata sistema. Oni služe da prikažu:

- proceduralnu logiku,
- poslovni proces ili tok posla.

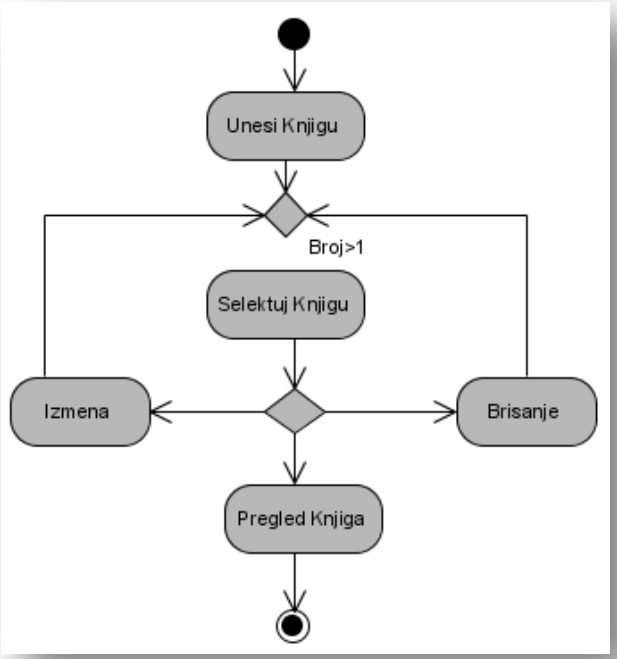
Slični su blok-dijagramima za opis algoritama (dodatno, podržavaju paralelno ponašanje). Dijagrami aktivnosti mogu se tretirati kao specijalan slučaj dijagrama stanja. U čvorovima ovog dijagrama prikazane su akcije. Akcije se implemetiraju kao metodi klasa ili neke podaktivnosti.



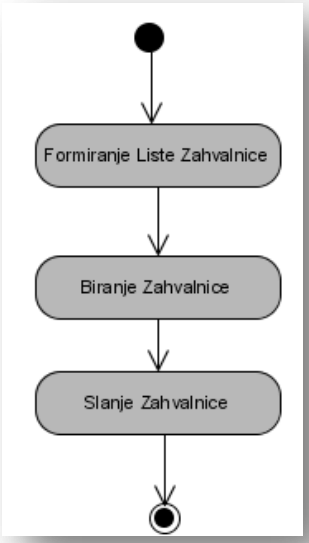
Dijagram 25– dijagram aktivnosti: naruči



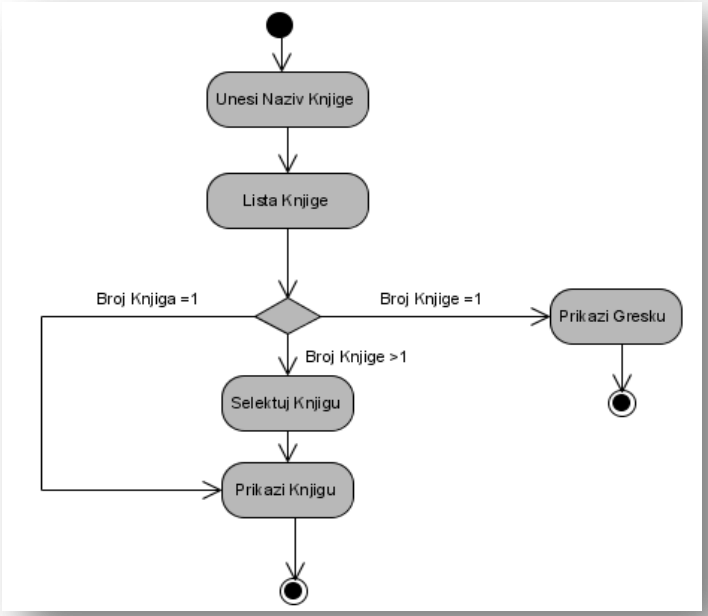
Dijagram 26– dijagram aktivnosti: potvrda



Dijagram 27– dijagram aktivnosti: izmena, brisanje knjiga



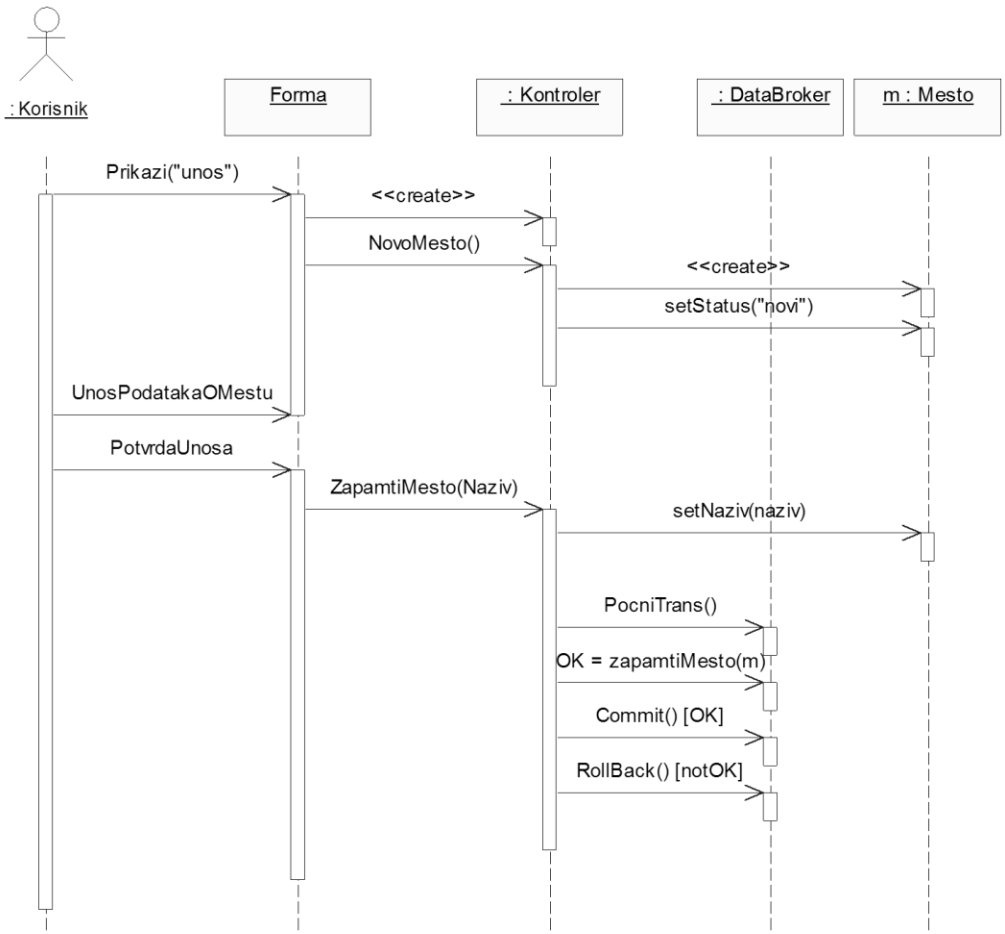
Dijagram 28– dijagram aktivnosti: slanje zahvalnice



Dijagram 29 – dijagram aktivnosti: biranje knjiga

Sada ćemo uočiti odnos između UML modelovanja i ekranskih formi koje su programirane u Javi. Ovde ću navesti samo neke dijagrame sekvenci kako bi uočili povezanost, tako da proces modelovanja predstavlja vizuelni plan kako će kasnije izgledati forma odnosno ceo informacioni sistem. Naveo sam par dijagrama iz projekta "informacioni sistem – prijemni ispit".

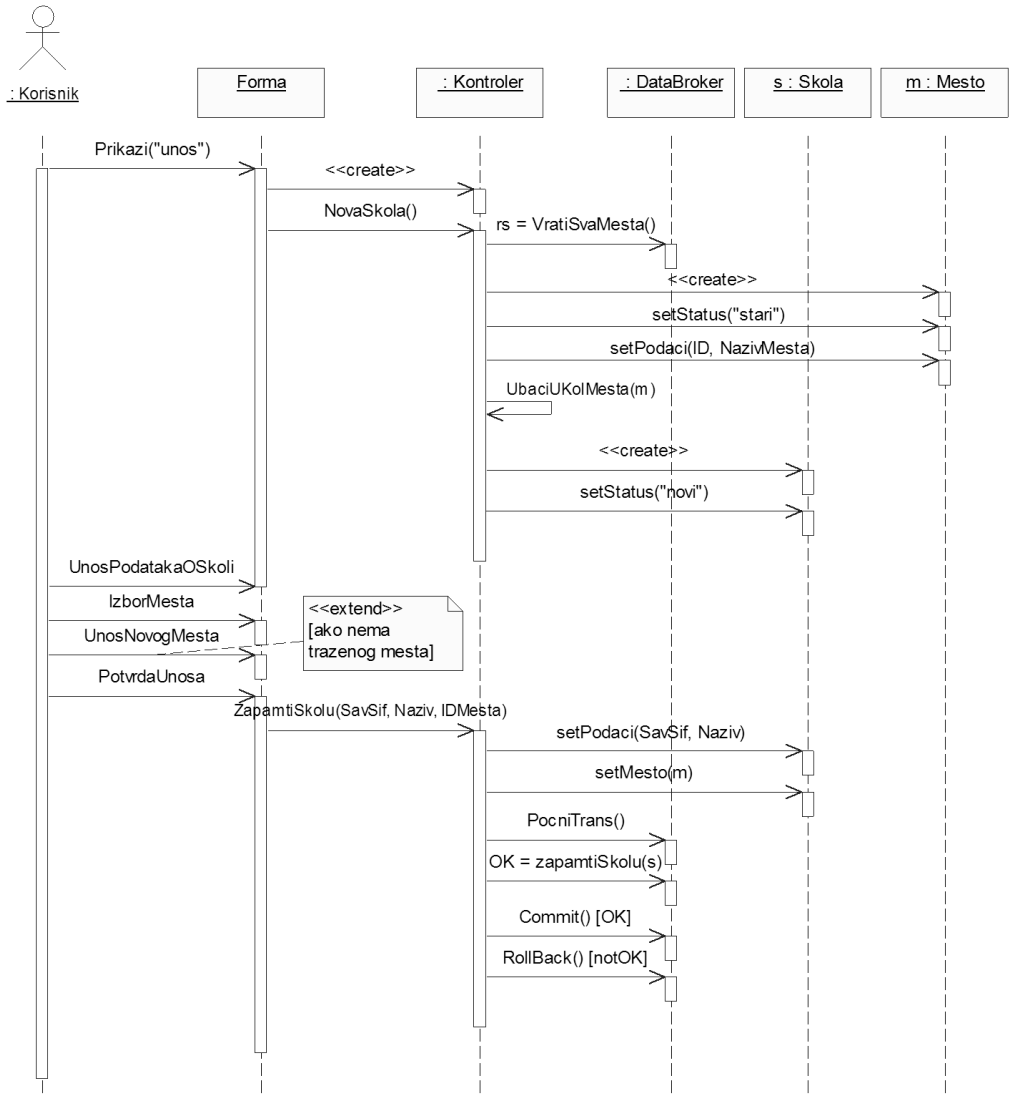
Dijagram sekvenci - Unos Novog Mesta:



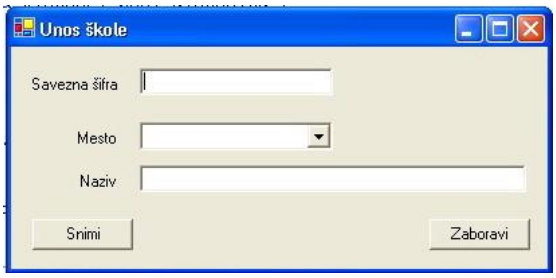
Ekranska forma – Unos mesta

Takođe uočite zavisnost i sledećeg dijagrama sa ekranskom formom.

Dijagram sekvenci - Unos Nove Skole

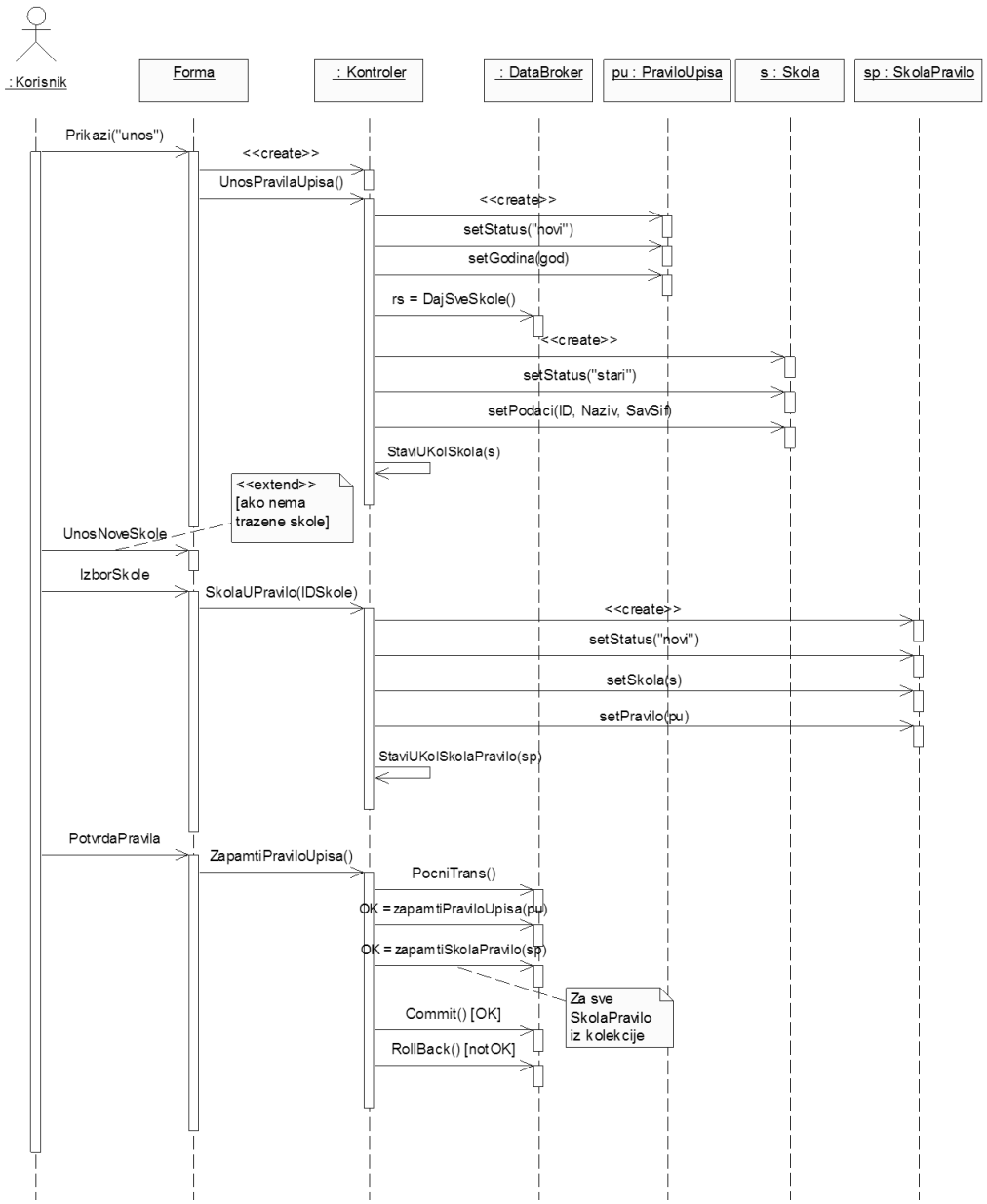


Ekranska forma – unos škole

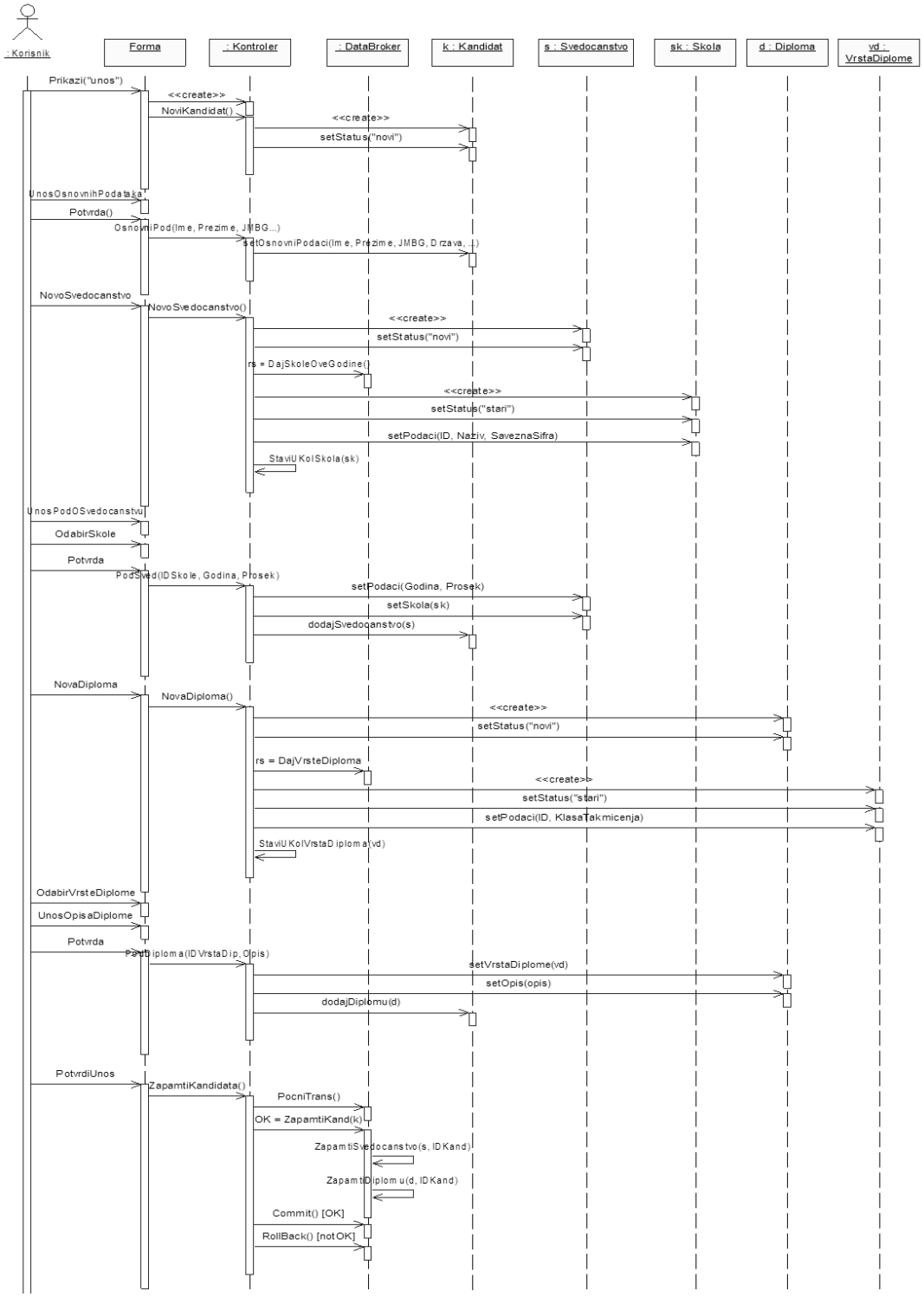


Sada su navedena dva dijagram sekvenci za složenije procese (unos pravila upisa i unos kandidata) a na posle sam naveo izgled ekranskih formi koji se baziraju na datim UML dijagramima.

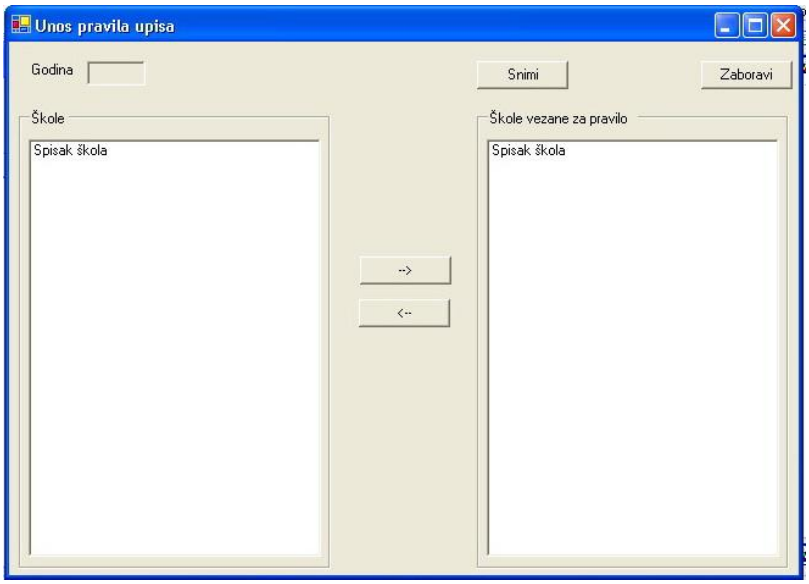
Dijagram sekvenci- Unos pravila Upisa



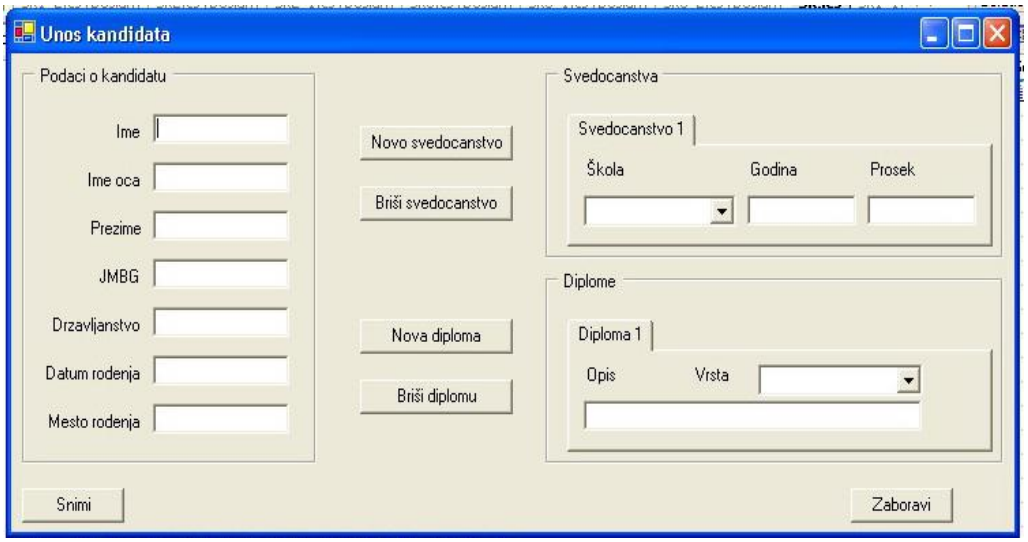
Dijagram sekvenci- Unos kandidata



Ekranska forma – unos pravila upisa



Ekranska forma – unos kandidata



ZAKLJUČAK

Dragi čitaoci i moji studenti, nadam se da sam vas upoznao sa programskim jezikom Java i UML alatom za modelovanje i da ste spremni da u daljem školovanju i radu primenite svoja znanja. Ono što mi preostaje je da u ovom poslednjem delu ove zbirke, tj. zaključku navedem par prednosti i razloga zbog čega se opredeliti baš za ove alate u daljem profesionalnom radu. Nadam se da mi ne treba puno da vas ubedim jer su prednosti zaista mnogobrojne.

Prvi razlog za popularnost Jave je njena cena – **potpuno je besplatna**!. Drugi razlog za popularnost je to što je Java programi mogu da se izvršavaju na skoro **svakom tipu računara**. Kažemo da su Java programi nezavisni od platforme na kojoj se izvršavaju.

Java može da se koristi za **razvoj raznovrsnih aplikacija**. Postoje jednostavni tekstualno-zasnovani programi koji se nazivaju konzolnim aplikacijama. Takvi programi podržavaju samo tekstualni unos i ispis na monitoru vašeg računara.
Takođe, možete da pravite aplikacije sa **grafičkim korisničkim interfejsom** (engl. Graphical User Interface – GUI). Ove aplikacije raspolažu sa menijima, paletama alatki, dugmadima, trakama za pomeranje sadržaja drugim kontrolama koje reaguju na miša. Primeri GUI aplikacija koje smo u ovoj zbirci radili su programi za obradu teksta, programi za rad sa tabelama i računarske igrice.
Takođe, možete praviti i aplikacije koje se nazivaju **apleti** (engl. Applets). To su male GUI aplikacije koje mogu da se izvršavaju unutar web stranice. Apleti daju dinamičnost web stranicama.

Drugo popularno svojstvo Jave je to što je ona **objektno orijentisana.** To je fini način da se kaže da su Java programi sačinjeni od više osnovnih delova (komponenti) koji mogu da se više puta koriste. To vas kao Java programera znači da možete da pravite i menjate velik programe bez većih komplikacija.

Prednost Jave je to što je i ona **prost jezik**, u poređenju sa drugim programskim jezicima i zbog toga se relativno lako uči. Ta jednostavnost je neophodna da bi se podržala nezavisnost Java aplikacija od tipa platforme (sposobnost da se izvršava na svakom tipu računara). Međutim, ta jednostavnost ne znači da Java nije moćan programski jezik. Možete pomoću Jave da uradite sve ono što možete da uradite sa bilo kojim mnogo složenijim programskim jezikom.

Analiza i rešavanje problema pomoću **UML-a** ima mnoge prednosti. UML standard koji se primenjuje kod objektno orijentisanog pristupa predviđa odgovarajuce poglede na sistem, s tim što se u svakom pogledu sistem može opisati sa statičkog (strukturnog) i dinamičkog aspekta. Koristi se za konstrukcija software-a kod koga treba odraditi plan ,nudi mogućnost vizualizacije u više dimenzija i nivoa detalja i prikladan je za nadogradnju nasljeđenih, starih sistema.

Prednosti UML-a su: otvoren standard, obuhvata celi životni ciklus oblikovanja programske potpore, temeljen je na iskustvu i potrebama zajednice oblikovatelja i korisnika programske potpore, podržavaju ga mnogi alati. UML notacija nudi razlicite dijagrame za razlicite svrhe, on je moćan i bogat opcijama. UML dijagrami imaju jasno definisanu semantiku, podržava apstrakcije, i ima široku primenu .

LITERATURA

[1] Milosavljević, B., Vidaković, M., ***Java i internet programiranje,*** *Grupa za informacione tehnologije*, Novi Sad, 2002

[2] Horton, I., ***Od početka... Java 2 JDK 5***, CET, Beograd, 200

[3] Martin Keschenau, ***Reverse Engineering of UML Specifications from Java Programs,*** RWTH Aachen University,2004.

[4] Grady Booch, Robert Maksimchuk, Michael W. Engle,Bobbi J. Young, Ph.D.,Jim Conallen. ***Object-Oriented Analysis and Design with Applications,*** Kelli A. Houston.

[5] Mr.Ivana Stanojević, Dr.Dušan Surla, ***Uvod u objedinjeni jezik modeliranja,*** PMF Novi Sad.

[6] Hartmann, Đorđević, Gocić, ***Uvod u objektno-orijentisano modeliranje (Softverski inženjering).***

[7] Markus Dahm, ***Byte Code Engineering with the BCEL API,*** Free University of Berlin Institute-computer science, 2001.

[8] Jon Kern and Christopher Garrett , ***Effective Sequence Diagram Generation,*** Borland ,2003.

[9] Nenad Jovanović, ***Objektno-orjentisano programiranje,*** *prezentacija.*

[10] Anglelina Njeguš, ***Metodološke osnove razvoja informacionih sistema,*** prezentacija, Univerzitet Singidunum.

[11]Elsa Estevez , Adegboyega Ojo, ***Object-Oriented Analysis and Design with UML.*** www.cd3wd.com/CD3WD_40/UNU_JAVA/19/report19.pdf

[12] Jon Oldevik, ***UML Model Transformation Tool- Overview and user guide documentation.***
umt-qvt.sourceforge.net/docs/UMT_documentation_v08.pdf

[13] Sead Mašović, Muzafer Saračević , Hamza Kamberović, ***Objektno-orijentisani pristup u simulaciji i metodologija simulacionog modeliranja,*** (Sept.2010). Festival informatičkih dostignuća - INFOFEST 2010 Budva, Crna Gora.

[14] Muzafer Saračević, Sead Mašović, ***Primena UML modelovanja i PHP jezika u izradi web aplikacije za e-učenje*** (Oktobar, 2010). Univerzitet METROPOLITAN - Elektronsko učenje na putu ka društvu znanja 2010, Beograd.

[15] Muzafer Saračević, Sead Mašović, Esad Međedović, ***Application of object-oriented analysis and design in navigation systems and transport networks*** (Sept, 2010), 10th International Conference "Research and Development in Mechanical Industry" RaDMI 2010.

[16] Muzafer Saračević, Mašović Sead, Lončarević Zoran: ***Primena UML dijagrama aktivnosti u predstavljanju Data Mining modela tehnikom genetskih algoritama,*** 18. telekomunikacioni forum TELFOR 2010.

[17] Muzafer Saračević, Mašović Sead, Kamberović Hamza: ***Tehnike Text Mining-a i njihova realizacija primenom objektno-orijentisane analize***, 18. telekomunikacioni forum TELFOR 2010.

[18] Sead Mašović, Muzafer Saračević, ***Modelovanje poslovnih procesa i primena Data Mining tehnika u e-učenju*** (Oktobar, 2010). Univerzitet METROPOLITAN - Elektronsko učenje na putu ka društvu znanja 2010, Beograd.

LISTING PAKETA U JAVI

java.applet	java.security	javax.naming.directory
java.awt	java.security.acl	javax.naming.event
java.awt.color	java.security.cert	javax.naming.ldap
java.awt.datatransfer	java.security.interfaces	javax.naming.spi
java.awt.dnd	java.security.spec	javax.net
java.awt.event	java.sql	javax.net.ssl
java.awt.font	java.text	javax.print
java.awt.geom	java.util	javax.print.attribute
java.awt.im	java.util.concurrent	javax.print.attribute.standard
java.awt.im.spi	java.util.concurrent.atomic	javax.print.event
java.awt.image	java.util.concurrent.locks	javax.rmi
java.awt.image.renderable	java.util.jar	javax.rmi.CORBA
java.awt.print	java.util.logging	javax.rmi.ssl
java.beans	java.util.prefs	javax.security.auth
java.beans.beancontext	java.util.regex	javax.security.auth.callback
java.io	java.util.zip	javax.security.auth.kerberos
java.lang	javax.accessibility	javax.security.auth.login
java.lang.annotation	javax.activity	javax.security.auth.spi
java.lang.instrument	javax.crypto	javax.security.auth.x500
java.lang.management	javax.crypto.interfaces	javax.security.cert
java.lang.ref	javax.crypto.spec	javax.security.sasl
java.lang.reflect	javax.imageio	javax.sound.midi
java.math	javax.imageio.event	javax.sound.midi.spi
java.net	javax.imageio.metadata	javax.sound.sampled
java.nio	javax.imageio.plugins.bmp	javax.sound.sampled.spi
java.nio.channels	javax.imageio.plugins.jpeg	javax.sql
java.nio.channels.spi	javax.imageio.spi	javax.sql.rowset
java.nio.charset	javax.imageio.stream	javax.sql.rowset.serial
java.nio.charset.spi	javax.management	javax.sql.rowset.spi
java.rmi	javax.management.loading	javax.swing
java.rmi.activation	javax.management.modelmbean	javax.swing.border
java.rmi.dgc	javax.management.monitor	javax.swing.colorchooser
java.rmi.registry	javax.management.openmbean	javax.swing.event
java.rmi.server	javax.management.relation	javax.swing.filechooser
javax.swing.plaf.basic	javax.management.remote	javax.swing.plaf
javax.swing.plaf.metal	javax.management.remote.rmi	javax.swing.text.html
javax.swing.plaf.multi	javax.management.timer	javax.swing.text.html.parser
javax.swing.plaf.synth	javax.naming	javax.swing.text.rtf
javax.swing.table	javax.transaction.xa	javax.swing.tree
javax.swing.text	javax.xml	javax.swing.undo
javax.xml.parsers	javax.xml.datatype	javax.transaction
javax.xml.transform	javax.xml.namespace	javax.xml.transform.dom
javax.xml.transform.sax	javax.xml.xpath	
javax.xml.transform.stream		
javax.xml.validation		

LISTING klasa u pojedinim paketima

java.applet	Interfaces AppletContext AppletStub AudioClip	Classes Applet
java.awt	Interfaces ActiveEvent Adjustable Composite CompositeContext ItemSelectable KeyEventDispatcher KeyEventPostProcessor LayoutManager LayoutManager2 MenuContainer Paint PaintContext PrintGraphics Shape Stroke Transparency Exceptions AWTException FontFormatException HeadlessException IllegalComponentStateException Errors AWTError	Classes AlphaComposite AWTEvent AWTEventMulticaster AWTKeyStroke AWTPermission BasicStroke BorderLayout BufferCapabilities BufferCapabilities.FlipContents Button Canvas CardLayout Checkbox CheckboxGroup CheckboxMenuItem Choice Color Component ComponentOrientation Container ContainerOrderFocusTraversalPolicy Cursor DefaultFocusTraversalPolicy DefaultKeyboardFocusManager Dialog Dimension DisplayMode Event EventQueue FileDialog FlowLayout FocusTraversalPolicy

		Font FontMetrics Frame GradientPaint Graphics Graphics2D GraphicsConfigTemplate GraphicsConfiguration GraphicsDevice GraphicsEnvironment GridBagConstraints GridBagLayout GridLayout Image ImageCapabilities Insets JobAttributes JobAttributes.DefaultSelectionType JobAttributes.DestinationType JobAttributes.DialogType JobAttributes.MultipleDocumentHandling Type JobAttributes.SidesType KeyboardFocusManager Label List MediaTracker Menu MenuBar MenuComponent MenuItem MenuShortcut MouseInfo PageAttributes PageAttributes.ColorType PageAttributes.MediaType PageAttributes.OrientationRequestedType PageAttributes.OriginType PageAttributes.PrintQualityType Panel Point PointerInfo Polygon PopupMenu

		PrintJob Rectangle RenderingHints RenderingHints.Key Robot Scrollbar ScrollPane ScrollPaneAdjustable SystemColor TextArea TextComponent TextField TexturePaint Toolkit Window
java.io	Interfaces Closeable DataInput DataOutput Externalizable FileFilter FilenameFilter Flushable ObjectInput ObjectInputValidation ObjectOutput ObjectStreamConstants Serializable Exceptions CharConversionException EOFException FileNotFoundException InterruptedIOException InvalidClassException InvalidObjectException IOException NotActiveException NotSerializableException ObjectStreamException OptionalDataException StreamCorruptedException SyncFailedException UnsupportedEncodingException UTFDataFormatException WriteAbortedException	Classes BufferedInputStream BufferedOutputStream BufferedReader BufferedWriter ByteArrayInputStream ByteArrayOutputStream CharArrayReader CharArrayWriter DataInputStream DataOutputStream File FileDescriptor FileInputStream FileOutputStream FilePermission FileReader FileWriter FilterInputStream FilterOutputStream FilterReader FilterWriter InputStream InputStreamReader LineNumberInputStream LineNumberReader ObjectInputStream ObjectInputStream.GetField

		ObjectOutputStream ObjectOutputStream.PutField ObjectStreamClass ObjectStreamField OutputStream OutputStreamWriter PipedInputStream PipedOutputStream PipedReader PipedWriter PrintStream PrintWriter PushbackInputStream PushbackReader RandomAccessFile Reader SequenceInputStream SerializablePermission StreamTokenizer StringBufferInputStream StringReader StringWriter Writer
java.math	Enums RoundingMode	Classes BigDecimal BigInteger MathContext
javax.sql	Interfaces ConnectionEventListener ConnectionPoolDataSource DataSource PooledConnection RowSet RowSetInternal RowSetListener RowSetMetaData RowSetReader RowSetWriter XAConnection XADataSource	Classes ConnectionEvent RowSetEvent
javax.swing	Interfaces Action BoundedRangeModel	Classes AbstractAction AbstractButton

	ButtonModel CellEditor ComboBoxEditor ComboBoxModel DesktopManager Icon JComboBox.KeySelectionManager ListCellRenderer ListModel ListSelectionModel MenuElement MutableComboBoxModel Renderer RootPaneContainer Scrollable ScrollPaneConstants SingleSelectionModel SpinnerModel SwingConstants UIDefaults.ActiveValue UIDefaults.LazyValue WindowConstants Enums JTable.PrintMode Exceptions UnsupportedLookAndFeelException	AbstractCellEditor AbstractListModel AbstractSpinnerModel ActionMap BorderFactory Box Box.Filler BoxLayout ButtonGroup CellRendererPane ComponentInputMap DebugGraphics DefaultBoundedRangeModel DefaultButtonModel DefaultCellEditor DefaultComboBoxModel DefaultDesktopManager DefaultFocusManager DefaultListCellRenderer DefaultListCellRenderer.UIResource DefaultListModel DefaultListSelectionModel DefaultSingleSelectionModel FocusManager GrayFilter ImageIcon InputMap InputVerifier InternalFrameFocusTraversalPolicy JApplet JButton JCheckBox JCheckBoxMenuItem JColorChooser JComboBox JComponent JDesktopPane JDialog JEditorPane JFileChooser JFormattedTextField JFormattedTextField.AbstractFormatter JFormattedTextField.AbstractFormatterFactory JFrame

		JInternalFrame JInternalFrame.JDesktopIcon JLabel JLayeredPane JList JMenu JMenuBar JMenuItem JOptionPane JPanel JPasswordField JPopupMenu JPopupMenu.Separator JProgressBar JRadioButton JRadioButtonMenuItem JRootPane JScrollBar JScrollPane JSeparator JSlider JSpinner JSpinner.DateEditor JSpinner.DefaultEditor JSpinner.ListEditor JSpinner.NumberEditor JSplitPane JTabbedPane JTable JTextArea JTextField JTextPane JToggleButton JToggleButton.ToggleButtonModel JToolBar JToolBar.Separator JToolTip JTree JTree.DynamicUtilTreeNode JTree.EmptySelectionModel JViewport JWindow KeyStroke LayoutFocusTraversalPolicy LookAndFeel

		MenuSelectionManager OverlayLayout Popup PopupFactory ProgressMonitor ProgressMonitorInputStream RepaintManager ScrollPaneLayout ScrollPaneLayout.UIResource SizeRequirements SizeSequence SortingFocusTraversalPolicy SpinnerDateModel SpinnerListModel SpinnerNumberModel Spring SpringLayout SpringLayout.Constraints SwingUtilities Timer ToolTipManager TransferHandler UIDefaults UIDefaults.LazyInputMap UIDefaults.ProxyLazyValue UIManager UIManager.LookAndFeelInfo ViewportLayout
javax.swing.event	Interfaces AncestorListener CaretListener CellEditorListener ChangeListener DocumentEvent DocumentEvent.ElementChange DocumentListener HyperlinkListener InternalFrameListener ListDataListener ListSelectionListener MenuDragMouseListener MenuKeyListener MenuListener MouseInputListener PopupMenuListener TableColumnModelListener	Classes AncestorEvent CaretEvent ChangeEvent DocumentEvent.EventType EventListenerList HyperlinkEvent HyperlinkEvent.EventType InternalFrameAdapter InternalFrameEvent ListDataEvent ListSelectionEvent MenuDragMouseEvent MenuEvent MenuKeyEvent MouseInputAdapter

	TableModelListener TreeExpansionListener TreeModelListener TreeSelectionListener TreeWillExpandListener UndoableEditListener	PopupMenuEvent SwingPropertyChangeSupport TableColumnModelEvent TableModelEvent TreeExpansionEvent TreeModelEvent TreeSelectionEvent UndoableEditEvent

Lista zadataka

I

Lista dijagrama