

Laslo Kraus

JAVA

**REŠENI ZADACI  
IZ PROGRAMSKOG JEZIKA  
JAVA**

AKADEMSKA MISAO

## Ispravke uz drugo izdanje

Mesto	Pogrešno	Ispravno
strana: 17	4 linearne jednačine $x = -\frac{b}{c}$	4 linearne jednačine $x = -\frac{c}{b}$
strana: 58 red: ↑ 17	<pre>if (b &lt; 0) { a = -a; b = -b; } for (int i=0; i &lt; maxProst &amp;&amp;      prosti[i]*prosti[i]&lt;=a &amp;&amp;</pre>	<pre>if (b &lt; 0) { a = -a; b = -b; } long absA = a&lt;0 ? -a : a; for (int i=0; i &lt; maxProst &amp;&amp;      prosti[i]*prosti[i]&lt;=absA &amp;&amp;</pre>
strana: 25 red: ↑ 1	Decimalan broj? s	Decimalan broj? 9999
strana: 59 red: ↓ 21	<pre>if (rez != "") rez += (a &gt; 0) ? "+" :                "-";</pre>	<pre>if (rez != "" &amp;&amp; a &gt; 0) rez += "+";                "-";</pre>
strana: 61 red: ↑ 5	double rastojanje (Krug k)	<b>public</b> double rastojanje (Krug k)
strana: 67 red: ↓ 9	protected String	private String
strana: 71 red: ↓ 6	double ter = Citaj.Int();	double ter = Citaj.Double();
strana: 75	<pre> classDiagram     class Figura {         +Figura()         +premesti()         +postavi()         +O(), P()         +citaj(), toString()     }     T "1" --&gt; "1" Figura : ~T   </pre>	<pre> classDiagram     class Figura {         +Figura()         +pomeri()         +postavi()         +O(), P()         +citaj(), toString()     }     T "1" --&gt; "1" Figura : -T   </pre>
strana: 78		
strana: 79	<pre> classDiagram     class Tacka {         -x, y         +Tacka()         +x(), y()         +toString()         +pisiniZ()     }     tem "*" --&gt; "*" Tacka : -tem   </pre>	<pre> classDiagram     class Tacka {         -x, y         +Tacka()         +x(), y()         +toString()         +toString()     }     tem "*" --&gt; "*" Tacka : #tem   </pre>
strana: 87 red: ↑ 12	public double x, y, z;	private double x, y, z;
strana: 87 red: ↑ 8	double intenzitet () { ... }	public double intenzitet () { ... }
strana: 112 red: ↑ 20	import verizan;	import verizan.*;
strana: 116 red: ↓ 20	mat[i][j] += m2.mat[i][j];	mat[i][j] -= m2.mat[i][j];
strana: 119 red: ↓ 9	12.0 14.0 16.0 18.0 23.0 25.0 27.0 29.0 34.0 36.0 38.0 40.0	10.0 10.0 10.0 10.0 19.0 19.0 19.0 19.0 28.0 28.0 28.0 28.0
strana: 126 red: ↓ 14	interaktivni program	program
strana: 137		« Treba izbaciti. »
strana: 140 red: ↑ 19	double a, b; char vrs;	double a, b;
strana: 153	<pre> classDiagram     interface Zbirka     interface Iterator     Zbirka &lt;--&gt; Iterator     Znak   </pre>	<pre> classDiagram     interface Zbirka     interface Iterator     Zbirka &lt;--&gt; Iterator     Znak   </pre>

**Laslo Kraus**

# **REŠENI ZADACI**

**IZ**

**PROGRAMSKOG JEZIKA**

# **Java**

**AKADEMSKA MISAO**

Beograd, 2007

Laslo Kraus

REŠENI ZADACI  
IZ PROGRAMSKOG JEZIKA  
**JAVA**

Drugo, prerađeno izdanje

*Recenzenti*

Dr Igor Tartalja

Dr Jelica Protić

*Izdavač*

AKADEMSKA MISAO

Bul. kralja Aleksandra 73, Beograd

*Štampa*

Planeta print, Beograd

*Tiraž*

500 primeraka

ISBN 86-7466-281-1

---

NAPOMENA: Fotokopiranje ili umnožavanje na bilo koji način ili ponovno objavljivanje ove knjige - u celini ili u delovima - nije dozvoljeno bez prethodne izričite saglasnosti i pismenog odobrenja izdavača.

---

## P r e d g o v o r

Ova zbirka zadataka je pomoćni udžbenik za učenje programiranja na jeziku *Java*. Zbirka je namenjena za upotrebu u fakultetskoj nastavi ali može da se koristi i za samostalno produbljivanje znanja iz programiranja.

Rešenja svih zadataka su potpuna u smislu da priloženi programi mogu da se izvršavaju na računaru. Pored samih tekstova programa priloženo je samo malo objašnjenja, prvenstveno u obliku slika i formula. Očekuje se da će izvođač nastave dati dodatna usmena objašnjenja slušaocima. Uz malo više napora zadaci mogu da se shvate i samostalno. Uz svaki program dat je i primer izvršavanja da bi se olakšalo razumevanje rada programa.

Kroz zadatke, pored elemenata samog jezika, prikazani su osnovni principi objektno orijentisanog programiranja (sakrivanje podataka, ponovno korišćenje koda, nasleđivanje i polimorfizam), konkurentnog programiranja (rad s nitima) i izrade programa zasnovanih na grafičkoj korisničkoj površi (rad s prozorima). Prikazani su i najčešće korišćeni postupci u programiranju: pretraživanje i uređivanje nizova, obrada znakovnih podataka, rad s bitovima, rad s dinamičkim strukturama podataka (kao što su liste i stabla) i obrada datoteka. Posebna pažnja posvećena je i inženjerskim aspektima programiranja: preglednosti, razumljivosti i efikasnosti.

Izvorni tekstovi svih programa iz ove zbirke mogu da se preuzmu preko Interneta sa adrese [galeb.etf.bg.ac.yu/~kraus/knjige/](http://galeb.etf.bg.ac.yu/~kraus/knjige/). Svoja zapažanja čitaoci mogu da upute elektronskom poštom na adresu [kraus@etf.bg.ac.yu](mailto:kraus@etf.bg.ac.yu).

Beograd, januar 2007.

*Laslo Kraus*

# Sadržaj

Predgovor .....	3
Sadržaj .....	4
Preporučena literatura .....	6
<b>1 Operatori .....</b>	<b>7</b>
Zadatak 1.1 Ispisivanje teksta na glavnom izlazu .....	8
Zadatak 1.2 Izračunavanje zbira dva cela broja .....	9
Zadatak 1.3 Izračunavanje obima i površine kruga .....	10
Zadatak 1.4 Izračunavanje površine trougla .....	11
Zadatak 1.5 Pakovanje i raspakivanje vremena .....	12
<b>2 Naredbe .....</b>	<b>13</b>
Zadatak 2.1 Nalaženje najmanjeg od tri broja .....	14
Zadatak 2.2 Uređivanje tri broja .....	15
Zadatak 2.3 Rešavanje sistema od dve linearne jednačine .....	16
Zadatak 2.4 Rešavanje kvadratne jednačine .....	17
Zadatak 2.5 Tabeliranje vrednosti izraza .....	18
Zadatak 2.6 Izračunavanje vrednosti složenijih izraza .....	19
Zadatak 2.7 Tabeliranje vrednosti složenijih izraza .....	21
Zadatak 2.8 Izračunavanje aritmetičke srednje vrednosti i standardne devijacije niza brojeva .....	22
Zadatak 2.9 Značaj redosleda sabiranja niza realnih brojeva .....	23
Zadatak 2.10 Određivanje datuma za naredni dan .....	24
Zadatak 2.11 Ispisivanje celih brojeva u binarnom brojevnom sistemu .....	25
<b>3 Nizovi .....</b>	<b>27</b>
Zadatak 3.1 Tabeliranje vrednosti polinoma .....	28
Zadatak 3.2 Izračunavanje srednje vrednosti elemenata niza .....	29
Zadatak 3.3 Nalaženje vrednosti najmanjeg elementa u nizu .....	30
Zadatak 3.4 Obrtanje redosleda elemenata niza .....	31
Zadatak 3.5 Izračunavanje binomnih koeficijenata .....	32
Zadatak 3.6 Nalaženje fuzije dva uređena niza .....	33
Zadatak 3.7 Uređivanje niza .....	34
Zadatak 3.8 Umetanje niza u drugi niz .....	36
Zadatak 3.9 Medusobna zamena najmanjeg i najvećeg elementa matrice .....	37
Zadatak 3.10 Transponovanje pravougaone matrice .....	38

<b>4 Klase .....</b>	<b>39</b>
Zadatak 4.1 Tačke u ravni .....	40
Zadatak 4.2 Kompleksni brojevi .....	41
Zadatak 4.3 Numerisani trouglovi .....	43
Zadatak 4.4 Stekovi ograničenih kapaciteta .....	45
Zadatak 4.5 Nizovi kompleksnih brojeva .....	48
Zadatak 4.6 Uređeni skupovi brojeva .....	50
Zadatak 4.7 Liste brojeva .....	52
Zadatak 4.8 Redovi brojeva neograničenih kapaciteta .....	56
Zadatak 4.9 Racionalni brojevi .....	58
Zadatak 4.10 Tačke i krugovi koji ne smeju da se preklapaju u ravni .....	60
<b>5 Izvedene klase .....</b>	<b>63</b>
Zadatak 5.1 Valjci i kante .....	64
Zadatak 5.2 Osobe, đaci i zaposleni .....	66
Zadatak 5.3 Vozila, teretna vozila i putnička vozila .....	69
Zadatak 5.4 Predmeti, sfere i kvadri .....	72
Zadatak 5.5 Geometrijske figure, krugovi, kvadrati i trouglovi u ravni .....	75
Zadatak 5.6 Tačake, linije, duži, izlomljene linije i poligoni u ravni .....	79
Zadatak 5.7 Prosti i složeni otpornici; redne i paralelne veze otpornika .....	83
Zadatak 5.8 Vektori, brzine, pokretni objekti i tačke u prostoru .....	87
Zadatak 5.9 Uporedivi objekti, razne vrste uređivača objekata i celi brojevi .....	90
Zadatak 5.10 Izrazi, konstante, promenljive, dodele vrednosti i aritmetičke operacije .....	97
Zadatak 5.11 Naredbe, proste naredbe, sekvene i ciklusi .....	102
<b>6 Izuzeci .....</b>	<b>107</b>
Zadatak 6.1 Vektor realnih brojeva sa zadatim opsezima indeksa .....	108
Zadatak 6.2 Verižni razlomci .....	111
Zadatak 6.3 Matrice realnih brojeva .....	114
Zadatak 6.4 Police za predmete .....	120
Zadatak 6.5 Liste objekata i greške .....	126
Zadatak 6.6 Predmeti koji mogu da se kopiraju, tela, sfere, kvadri i sklopovi .....	130
Zadatak 6.7 Funkcije sa izračunavanjem određenog integrala .....	134
Zadatak 6.8 Funkcije za koje mogu da se stvaraju izvodi, monomi, eksponencijalne funkcije i zbirni funkcija .....	138
Zadatak 6.9 Mašine, mašine za sfere, mašine za kvadre i radnici .....	142
Zadatak 6.10 Časovnici, pokretni radnici, kupci, prodavci i radnje .....	146
Zadatak 6.11 Zbirke, iteratori, neuredeni i uređeni nizovi i liste .....	153
<b>7 Niti .....</b>	<b>163</b>
Zadatak 7.1 Vektori s konkurentnim izračunavanjem zbiru i skalarnog proizvoda .....	164
Zadatak 7.2 Skladišta, aktivni proizvođači, potrošači i izveštaci .....	166
Zadatak 7.3 Uporedivi znakovi, aktivni časovnici, uređivači i nizovi znakova .....	171
<b>8 Grafička korisnička površ .....</b>	<b>175</b>
Zadatak 8.1 Ispisivanje teksta u prozoru .....	176
Zadatak 8.2 Izračunavanje zbiru dva broja .....	177
Zadatak 8.3 Izračunavanje rastojanja između dve tačke u prostoru .....	179
Zadatak 8.4 Jednostavan kalkulator za cele brojeve .....	181

<b>Zadatak 8.5</b>	Klasa za komunikaciju s grafičkom korisničkom površij i klasa kompleksnih brojeva s grafičkim prikazom.....	186
<b>Zadatak 8.6</b>	Uređivanje niza celih brojeva.....	191
<b>Zadatak 8.7</b>	Crtanje Lisažuovih figura.....	193
<b>Zadatak 8.8</b>	Apstraktne funkcije, polinomi, oscilacije, logaritmi i grafici funkcija .....	195
<b>Zadatak 8.9</b>	Popunjene figure, krugovi, pravougaonici, mnogouglovi, crteži i platna.....	201
<b>Zadatak 8.10</b>	Skladišta, aktivni proizvodači i potrošači .....	208
<b>Zadatak 8.11</b>	Aktivni časovnici.....	212
<b>Zadatak 8.12</b>	Simuliranje trke automobila (aktivna i grafička vozila i staze) .....	214
<b>Zadatak 8.13</b>	Vatromet (klase: Vektor, Figura, Petarda, Krug, Zvezda, Top, Scena, Vatromet) .....	220
<b>Zadatak 8.14</b>	Simuliranje rada samoposluge (klase: Ulaz, Kupac, Red, Kasa, Samoposluga) .....	226
<b>Zadatak 8.15</b>	Igra XoX (klase tabli, igrača, čoveka, računara i igara).....	234
<b>Zadatak 8.16</b>	Prikazivači, konzole, prozori, histogrami, skupovi prikazivača, razne vrste uređivača .....	240
<b>Zadatak 8.17</b>	Crtanje ping-pong loptice u apletu.....	250
<b>9</b>	Datoteke .....	253
<b>Zadatak 9.1</b>	Klasa za čitanje podataka iz tekstualnog ulaznog toka, obrada sekvencijalne tekstualne datoteke .....	254
<b>Zadatak 9.2</b>	Obrada rečenica u tekstualnoj datoteci.....	257
<b>Zadatak 9.3</b>	Obrada sekvencijalne binarne datoteke.....	259
<b>Zadatak 9.4</b>	Klasa rečnika, snimanje objekata u datoteku .....	261

## Preporučena literatura

- [1] Herbert Schildt: **Java™ J2SE™ 5: kompletan priručnik**, Mikro knjiga, Beograd, 2006.
- [2] Ken Arnold, James Gosling, David Holmes: **Programski jezik Java™, Prevod trećeg izdanja**, CET, Beograd, 2001.
- [3] James Gosling, Bill Joy, Guy Steele, Gilad Bracha: **The Java™ Language Specification, third edition**, Addison Wesley Longman, Inc., Reading, Massachusetts, 2005. (dostupno u .html i .pdf verziji na adresi <http://java.sun.com/docs/books/jls/>)
- [4] Bruce Eckel: **Misliti na Javi, prevod drugog izdanja**, Mikro knjiga, Beograd, 2002.
- [5] Martin Fowler: **UML ukratko – Kratak vodič kroz standardni jezik za modelovanje objekata, prevod trećeg izdanja**, Mikro knjiga, Beograd, 2004.

# **1 Operatori**

### Zadatak 1.1 Ispisivanje teksta na glavnom izlazu

Napisati na jeziku Java program koji ispisuje tekst na glavnom izlazu računara.

Rešenje:

```
// Pozdravl.java - Ispisivanje pozdrava.

public class Pozdravl {
    public static void main (String[] varg) {
        System.out.println ("Pozdrav svima!");
    }
}
```

Obrada programa:

1) iz komandnog reda:

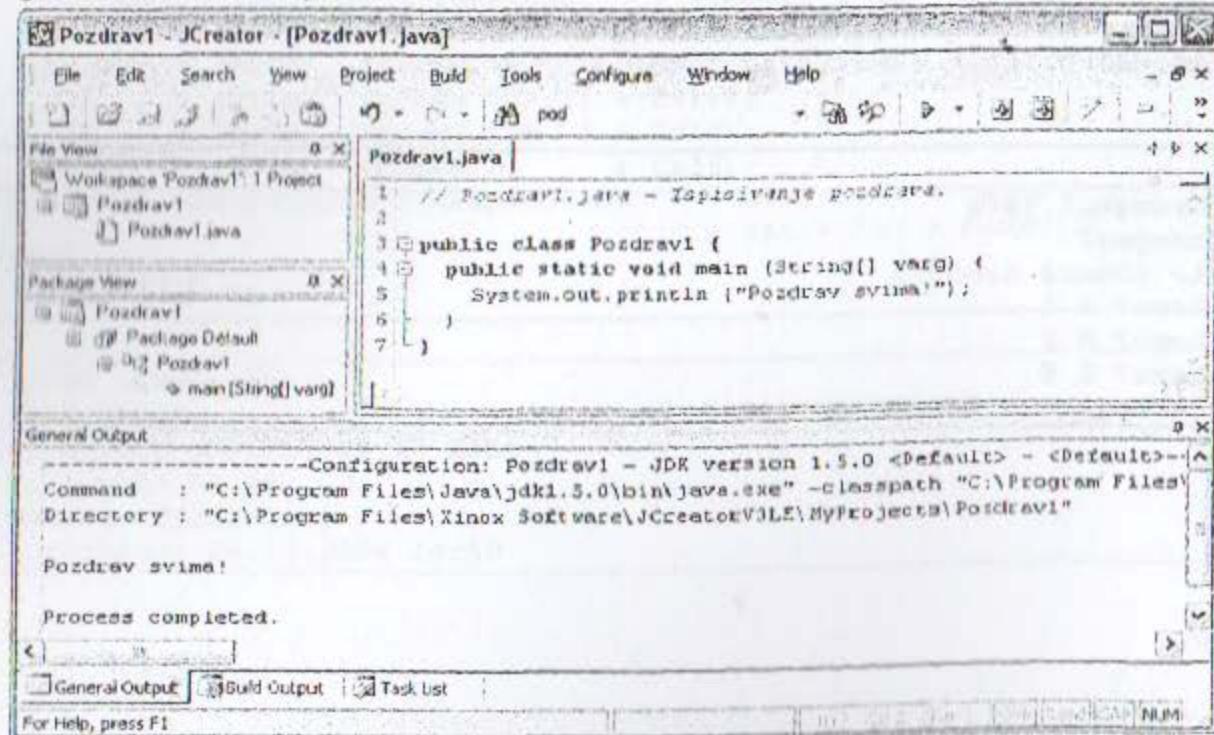
% edit ImeDatot.java	1. Unos izvornog teksta
% javac ImeDatot.java	2. Prevodenje u <b>ImeKlase.class</b>
% java ImeKlase	3. Izvršavanje (voditi računa o malim i velikim slovima)

Ime datoteke izvornog teksta (.java) ne mora da bude jednako imenu klase. U datoteci može da bude i više klasa, ali najviše jedna javna klasa. Ime javne klase mora da bude jednako imenu datoteke.

Prevod svake klase se stavlja u zasebnu datoteku (.class) čije je ime jednako imenu klase, vodeći računa o malim i velikim slovima.

```
% edit Pozdravl.java
% javac Pozdravl.java
% java Pozdravl
Pozdrav svima!
```

2) u programskom okruženju JCreator:



### Zadatak 1.2 Izračunavanje zbiru dva cela broja

Napisati na jeziku Java program koji pročita dva cela broja s glavnog ulaza računara, izračuna njihov zbir i ispiše rezultat na glavnom izlazu računara.

**Rešenje:**

```
// Zbir1.java - Izračunavanje zbiru dva cela broja.

public class Zbir1 {
    public static void main (String[] varg) {
        System.out.print ("Unesite dva cela broja: ");
        int a = Citaj.Int (), b = Citaj.Int ();
        int c = a + b;
        System.out.println ("Zbir unetih brojeva: " + c);
    }
}
```

```
$ javac Zbir1.java
$ java Zbir1
Unesite dva cela broja: 123 -45
Zbir unetih brojeva: 78
```

### Čitanje podataka s glavnog ulaza (nije standardno):

Preskaču se beline i čita se do sledećeg belog znaka:

Citaj.Byte()	byte	
Citaj.Short()	short	
Citaj.Int()	int	
Citaj.Long()	long	
Citaj.Float()	float	
Citaj.Double()	double	
Citaj.Boolean()	boolean	
Citaj.Char()	char	(prvi nebeli znak)
Citaj.String()	String	(jedna reč)

Bez preskakanja belih znakova:

Citaj.getCh()	char	(prvi znak, uključujući i bele znakove)
Citaj.Line()	String	(do kraja reda, uključujući i bele znakove)

Preskakanje svih znakova do kraja reda:

Citaj.getNL()

Ispitivanje da li je pročitan znak za kraj:

Citaj.end() boolean

### Zadatak 1.3 Izračunavanje obima i površine kruga

Napisati na jeziku Java program koji pročita poluprečnik kruga s glavnog ulaza računara i ispiše obim i površinu tog kruga na glavnom izlazu računara.

**Rešenje:**

```
// Krugl.java - Izračunavanje obima i površine kruga.

public class Krugl {
    public static void main (String[] args) {
        System.out.print ("Poluprecnik? ");
        double r = Citaj.Double ();
        System.out.println ("Obim      = " + (2*r*Math.PI));
        System.out.println ("Povrsina = " + (r*r*Math.PI));
    }
}
```

```
% javac Krugl.java
% java Krugl
Poluprecnik? 10
Obim      = 62.83185307179586
Povrsina = 314.1592653589793
```

---

**Zadatak 1.4 Izračunavanje površine trougla**

Napisati na jeziku Java program za izračunavanje površine trougla u ravni ako su zadate koordinate temena. Potrebne podatke čitati s glavnog ulaza računara, a rezultat ispisati na glavnom izlazu računara.

**Rešenje:**

$$A(x_A, y_A), \quad B(x_B, y_B), \quad C(x_C, y_C), \\ a = \sqrt{(x_B - x_C)^2 + (y_B - y_C)^2}, \quad b = \sqrt{(x_C - x_A)^2 + (y_C - y_A)^2}, \quad c = \sqrt{(x_A - x_B)^2 + (y_A - y_B)^2}, \\ s = (a + b + c) / 2, \quad P = \sqrt{s(s-a)(s-b)(s-c)}.$$

```
// Trougaol.java - Površina trougla u ravni.

public class Trougaol {
    public static void main (String[] varg) {

        // Temena trougla:
        System.out.println ("Koordinate temena trougla");
        System.out.print (" - prvo teme? ");
        double xA = Citaj.Double (), yA = Citaj.Double ();
        System.out.print (" - drugo teme? ");
        double xB = Citaj.Double (), yB = Citaj.Double ();
        System.out.print (" - treće teme? ");
        double xC = Citaj.Double (), yC = Citaj.Double ();

        // Stranice trougla:
        double a = Math.sqrt (Math.pow (xB-xC, 2) + Math.pow (yB-yC, 2));
        double b = Math.sqrt (Math.pow (xC-xA, 2) + Math.pow (yC-yA, 2));
        double c = Math.sqrt (Math.pow (xA-xB, 2) + Math.pow (yA-yB, 2));

        // Površina trougla:
        double s = (a + b + c) / 2;
        double P = Math.sqrt (s * (s-a) * (s-b) * (s-c));
        System.out.println ("Povrsina trougla: " + P);
    }
}
```

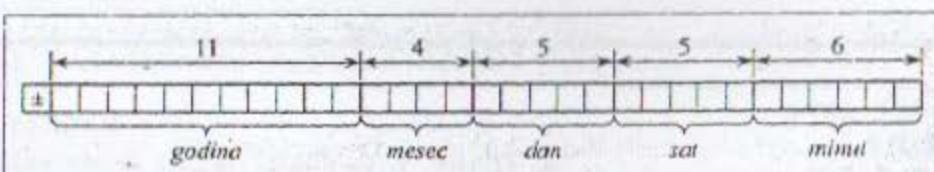
```
% javac Trougaol.java
% java Trougaol
Koordinate temena trougla
- prvo teme? 1 1
- drugo teme? 5 2
- treće teme? 3 6
Povrsina trougla: 9.000000000000007
```

### Zadatak 1.5 Pakovanje i raspakivanje vremena

Vreme se zadaje pomoću broja godina, meseci, dana, sati i minuta. Napisati na jeziku Java program za pakovanje i obrnuti proces za raspakivanje podataka o vremenu u jednu 32-bitnu celobrojnu promenljivu. Potrebne podatke čitati s glavnog ulaza računara, a rezultate ispisivati na glavnom izlazu računara.

**Rešenje:**

minut:	0 - 59	6 bitova (0 - 63)
sat:	0 - 23	5 bitova (0 - 31)
dan:	1 - 31	5 bitova (0 - 31)
mesec:	1 - 12	4 bita (0 - 15)
godina:		11 bitova (0 - 2047)
		31 bit (ne koristi se bit za predznak)



```
// vreme.java - Pakovanje i raspakivanje vremena.

public class Vreme {
    public static void main (String[] varg) {
        System.out.print ("Dan, mesec, godina? ");
        byte dan = Citaj.Byte (), mesec = Citaj.Byte ();
        short godina = Citaj.Short ();
        System.out.print ("Sat, minut? ");
        byte sat = Citaj.Byte (), minut = Citaj.Byte ();
        int vreme = godina << 20 | mesec << 16 | dan << 11 | sat << 6 | minut;
        System.out.println ("Pakovano: " + vreme);
        godina = (short) (vreme >>> 20);
        mesec = (byte) (vreme >>> 16 & 0x0f);
        dan = (byte) (vreme >>> 11 & 0x1f);
        sat = (byte) (vreme >>> 6 & 0x1f);
        minut = (byte) (vreme & 0x3f);
        System.out.println ("Raspakovano: " + dan + "." + mesec + "." + godina +
                            " " + sat + ";" + minut);
    }
}
```

```
% javac Vreme.java
% java Vreme
Dan, mesec, godina? 29 10 2004
Sat, minut? 14 59
Pakovano: 2102062011
Raspakovano: 29.10.2004 14:59
```

## 2 Naredbe

**Zadatak 2.1 Nalaženje najmanjeg od tri broja**

Napisati na jeziku Java program za nalaženje najmanjeg od tri cela broja.

**Rešenje:**

```
// min1.java - Najmanji od tri broja.

public class Min1 {
    public static void main (String[] args) {
        System.out.print ("Tri broja? ");
        int a = Citaj.Int (), b = Citaj.Int (), c = Citaj.Int ();
        int min = a;
        if (b < min) min = b;
        if (c < min) min = c;
        System.out.println ("Najmanji= " + min);
    }
}
```

```
% javac Min1.java
% java Min1
Tri broja? 4 2 7
Najmanji= 2
```

---

### Zadatak 2.2 Uređivanje tri broja

Napisati na jeziku Java program za uređivanje tri realna broja po neopadajućem redosledu.

**Rešenje:**

```
// Uredil.java - Uređivanje tri broja.

public class Uredil {
    public static void main (String[] varg) {
        System.out.print ("Tri broja? ");
        double a = Citaj.Double (), b = Citaj.Double (), c = Citaj.Double ();
        if (a > b) { double p = a; a = b; b = p; }
        if (a > c) { double p = a; a = c; c = p; }
        if (b > c) { double p = b; b = c; c = p; }
        System.out.println ("Uredjeno= " + a + " " + b + " " + c);
    }
}

% javac Uredil.java
% java Uredil
Tri broja? 8 2 5
Uredjeno= 2.0 5.0 8.0
```

### Zadatak 2.3 Rešavanje sistema od dve linearne jednačine

Napisati na jeziku Java program za rešavanje sistema od dve linearne jednačine s dve nepoznate.

**Rešenje:**

Jednačine:	Determinante:	Rešenja:
$a_1x + b_1y = c_1$	$D = \begin{vmatrix} a_1 & b_1 \\ a_2 & b_2 \end{vmatrix} = a_1b_2 - a_2b_1$	$D \neq 0 : x = \frac{D_x}{D}, y = \frac{D_y}{D}$
$a_2x + b_2y = c_2$	$D_x = \begin{vmatrix} c_1 & b_1 \\ c_2 & b_2 \end{vmatrix} = c_1b_2 - c_2b_1$	$D = D_x = D_y = 0 : \text{neodređeno}$
	$D_y = \begin{vmatrix} a_1 & c_1 \\ a_2 & c_2 \end{vmatrix} = a_1c_2 - a_2c_1$	inače: protivrečno

```
// LinJed.java - Rešavanje sistema od dve linearne jednačine.
```

```
public class LinJed {
    public static void main (String[] varg) {
        System.out.print ("Koeficijenti prve jednacine? ");
        double a1 = Citaj.Double (), b1 = Citaj.Double (), c1 = Citaj.Double ();
        System.out.print ("Koeficijenti druge jednacine? ");
        double a2 = Citaj.Double (), b2 = Citaj.Double (), c2 = Citaj.Double ();
        double D = a1 * b2 - a2 * b1,
               Dx = c1 * b2 - c2 * b1,
               Dy = a1 * c2 - a2 * c1;
        if (D != 0) {
            double x = Dx / D, y = Dy / D;
            System.out.println ("x= " + x);
            System.out.println ("y= " + y);
        } else
            if (Dx==0 && Dy==0)
                System.out.println ("Sistem je neodredjen.");
            else
                System.out.println ("Sistem je protivrecan.");
    }
}
```

```
$ javac LinJed.java
$ java LinJed
Koeficijenti prve jednacine? 1 2 3
Koeficijenti druge jednacine? 4 5 6
x= -1.0
y= 2.0
$ java LinJed
Koeficijenti prve jednacine? 1 2 3
Koeficijenti druge jednacine? 2 4 6
Sistem je neodredjen.
$ java LinJed
Koeficijenti prve jednacine? 1 2 3
Koeficijenti druge jednacine? 2 4 5
Sistem je protivrecan.
```

#### Zadatak 2.4 Rešavanje kvadratne jednačine

Napisati na jeziku Java program za rešavanje kvadratne jednačine.

Rešenje:

```
// KvadJed.java - Rešavanje kvadratne jednačine.

public class KvadJed {
    public static void main (String[] varg) {
        final int REALNI = 0, DVOSTRUKI = 1, // Šifre vrsta rešenja jednačine.
                  KOMPLEKSNI = 2, LINEARNA = 3, POGRESNA = 4;

        System.out.print ("Koeficijenti kvadratne jednacine? ");
        double a = Citaj.Double (), b = Citaj.Double (), c = Citaj.Double ();
        double x1 = 0, x2 = 0, // Realni delovi korena.
              y1 = 0, y2 = 0; // Imaginarni delovi korena.
        int vrsta; // Vrsta rešenja jednačine.

        if (a != 0) {
            double d = b * b - 4 * a * c; // Diskriminanta jednačine.
            if (d > 0) {
                vrsta = REALNI;
                x1 = (- b + Math.sqrt (d)) / (2 * a);
                x2 = (- b - Math.sqrt (d)) / (2 * a);
            } else if (d == 0) {
                vrsta = DVOSTRUKI;
                x1 = - b / (2 * a);
            } else {
                vrsta = KOMPLEKSNI;
                x1 = - b / (2 * a);
                y1 = Math.sqrt (-d) / (2 * a);
                x2 = x1;
                y2 = - y1;
            }
        } else
            if (b != 0) {
                vrsta = LINEARNA;
                x1 = - c / b;
            } else
                vrsta = POGRESNA;

        if (vrsta == REALNI)
            System.out.println ("Realni koren su " + x1 + " i " + x2);
        else if (vrsta == DVOSTRUKI)
            System.out.println ("Dvostruki realni koren je " + x1);
        else if (vrsta == KOMPLEKSNI)
            System.out.println ("Kompleksni koren su (" + x1 + ", " + y1 +
                                ") i (" + x2 + ", " + y2 + ")");
        else if (vrsta == LINEARNA)
            System.out.println ("Resenje linearne jednacine je " + x1);
        else if (vrsta == POGRESNA)
            System.out.println ("Podaci nemaju smisla!");
    }
}
```

Jednačina:	$ax^2 + bx + c = 0$
Diskriminanta:	$d = b^2 - 4ac$
Vrste rešenja:	
1 različita realna	$x_{1,2} = \frac{-b \pm \sqrt{d}}{2a}$ $(a \neq 0 \wedge d > 0)$
2 dvostruka realna	$x_{1,2} = \frac{-b}{2a}$ $(a \neq 0 \wedge d = 0)$
3 različita kompleksna	$z_{1,2} = \frac{-b \pm i\sqrt{-d}}{2a}$ $(a \neq 0 \wedge d < 0)$
4 linearna jednačina	$x = -b/c$ $(a = 0 \wedge b \neq 0)$
5 nema rešenja	$(a = 0 \wedge b = 0)$

```
% java KvadJed
Koeficijenti kvadratne jednacine? 1 1 1
Kompleksni koren su (-0.5, 0.8660254037844386) i (-0.5, -0.8660254037844386)
```

**Zadatak 2.5 Tabeliranje vrednosti izraza**

Napisati na jeziku Java program za tabeliranje vrednosti izraza

$$y = \frac{x^2 - 2x - 2}{x^2 + 1}$$

za svako  $x_{\min} \leq x \leq x_{\max}$  s korakom  $\Delta x$ .

**Rešenje:**

---

```
// Tabel1.java - Tabeliranje vrednosti izraza.

public class Tabel1 {
    public static void main (String[] varg) {
        System.out.print ("xmin, xmax, dx? ");
        double xmin = Citaj.Double (), xmax = Citaj.Double (),
               dx = Citaj.Double ();
        System.out.print ("\nx\ty\n=====\\n");
        for (double x=xmin; x<=xmax; x+=dx) {
            double y = (x*x - 2*x - 2) / (x*x + 1);
            System.out.println (x + "\t" + y);
        }
    }
}
```

---

```
% javac Tabel1.java
% java Tabel1
xmin, xmax, dx? -2 2 .2

x      y
=====
-2.0   1.2
-1.8   1.141509433962264
-1.6   1.056179775280899
-1.4000000000000001   0.9324324324324326
-1.2000000000000002   0.7540983606557379
-1.0000000000000002   0.5000000000000003
-0.8000000000000003   0.14634146341463478
-0.6000000000000003   -0.3235294117647051
-0.4000000000000003   -0.8965517241379302
-0.2000000000000003   -1.4999999999999991
-2.7755575615628914E-16 -1.9999999999999993
0.1999999999999973   -2.269230769230769
0.3999999999999974   -2.2758620689655173
0.5999999999999998   -2.0882352941176476
0.7999999999999998   -1.804878048780488
0.9999999999999998   -1.5000000000000004
1.1999999999999997   -1.213114754098361
1.3999999999999997   -0.9594594594594599
1.5999999999999996   -0.7415730337078656
1.7999999999999996   -0.5566037735849061
1.9999999999999996   -0.4000000000000003
```

---

### Zadatak 2.6 Izračunavanje vrednosti složenijih izraza

Napisati na jeziku Java programe za izračunavanje vrednosti sledećih izraza:

a)  $s = 1 + 2 + 3 + \dots + n = \sum_{i=1}^n i$

b)  $s = n! = 1 \cdot 2 \cdot 3 \cdot \dots \cdot n = \prod_{i=1}^n i$

c)  $s = 1! + 2! + 3! + \dots + n! = 1 + 1 \cdot 2 + 1 \cdot 2 \cdot 3 + \dots + 1 \cdot 2 \cdot 3 \cdot \dots \cdot n = \sum_{i=1}^n i! = \sum_{i=1}^n \prod_{j=1}^i j$

d)  $s = \frac{1!}{1} + \frac{2!}{2} + \frac{3!}{3} + \dots + \frac{n!}{n} = \sum_{i=1}^n \frac{i!}{\sum_{j=1}^i j}$

e)  $s = 1 - \frac{1+2}{1+2^2} + \frac{1+2+3}{1+2^2+3^2} - \dots + (-1)^{n-1} \frac{1+2+3+\dots+n}{1+2^2+3^2+\dots+n^2} = \sum_{i=1}^n (-1)^{i-1} \frac{\sum_{j=1}^i j}{\sum_{j=1}^i j^2}$

Rešenje:

a)

```
// Izraz1.java - Izračunavanje zbiru uzastopnih prirodnih brojeva.

public class Izraz1 {
    public static void main (String[] varg) {
        System.out.print ("n? "); int n = Citaj.Int ();
        double s = 0;
        for (int i=1; i<=n; s+=i++);
        System.out.println ("s= " + s);
    }
}
```

```
% java Izraz1
n? 5
s= 15.0
```

b)

```
// izraz2.java - Izračunavanje proizvoda uzastopnih prirodnih brojeva.

public class Izraz2 {
    public static void main (String[] varg) {
        System.out.print ("n? "); int n = Citaj.Int ();
        long s = 1;
        for (int i=1; i<=n; s*=i++);
        System.out.println ("s= " + s);
    }
}
```

```
% java Izraz2
n? 20
s= 2432902008176640000
```

c)

```
// Izraz3.java - Izračunavanje zbira faktorijela.

public class Izraz3 {
    public static void main (String[] varg) {
        System.out.print ("n? "); int n = Citaj.Int ();
        long s = 0;
        for (int f=1, i=1; i<=n; i++) {
            f *= i;
            s += f;
        }
        System.out.println ("s= " + s);
    }
}
```

```
# java Izraz3
n? 5
s= 153
```

Sažetije: `for (int f=1, i=1; i<=n; s+=(f*=i++)) ;`

d)

```
// Izraz4.java - Izračunavanje složenog zbiraa.

public class Izraz4 {
    public static void main (String[] varg) {
        System.out.print ("n? "); int n = Citaj.Int ();
        double s = 0, g = 0; long f = 1;
        for (int i=1; i<=n; i++) {
            f *= i;
            g += 1. / (i+1);
            s += f / g;
        }
        System.out.println ("s= " + s);
    }
}
```

```
# java Izraz4
n? 5
s= 111.39838092941542
```

Sažetije: `for (int i=1; i<=n; s+=(f*=i++)/(g+=1./i)) ;`

e)

```
// izraz5.java - Izračunavanje složenog zbiraa.

public class Izraz5 {
    public static void main (String[] varg) {
        System.out.print ("n? "); int n = Citaj.Int ();
        double s = 0;
        for (int f=0, g=0, i=1, z=1; i<=n; i++) {
            f += i;
            g += i * i;
            s += z * (double)f / g;
            z = -z;
        }
        System.out.println ("s= " + s);
    }
}
```

```
# java Izraz5
n? 5
s= 0.767965367965368
```

### Zadatak 2.7 Tabeliranje vrednosti složenijih izraza

Napisati na jeziku Java program za tabeliranje vrednosti izraza

$$\text{a)} \quad s = x + x^2 + x^3 + \dots + x^n = \sum_{i=1}^n x^i \quad \text{b)} \quad s = x - x^2 + x^3 - \dots + (-1)^{n-1} x^n = \sum_{i=1}^n (-1)^{i-1} x^i$$

za svako  $x_{\min} \leq x \leq x_{\max}$  s korakom  $\Delta x$ .

**Rešenje:**

a)

```
// Tabela2.java - Tabeliranje vrednosti izraza.

public class Tabela2 {
    public static void main (String[] varg) {
        System.out.print ("n? "); int n = Citaj.Int ();
        System.out.print ("xmin, xmax, dx? ");
        double xmin = Citaj.Double (), xmax = Citaj.Double (),
               dx = Citaj.Double ();
        System.out.print ("\n      x      s\n=====\\n");
        for (double x=xmin; x<=xmax; x+=dx) {
            double s = 0, p = 1;
            for (int i=1; i<=n; i++) s += (p *= x);
            System.out.format ("%8.3f%8.3f\\n", x, s);
        }
    }
}
```

```
# java Tabela2
n? 5
xmin, xmax, dx? -1 1 .25

      x      s
=====
-1.000  -1.000
-0.750  -0.530
-0.500  -0.344
-0.250  -0.200
0.000   0.000
0.250   0.333
0.500   0.969
0.750   2.288
1.000   5.000
```

b)

```
// Tabela3.java - Tabeliranje vrednosti izraza.

public class Tabela3 {
    public static void main (String[] varg) {
        System.out.print ("n? "); int n = Citaj.Int ();
        System.out.print ("xmin, xmax, dx? ");
        double xmin = Citaj.Double (), xmax = Citaj.Double (),
               dx = Citaj.Double ();
        System.out.print ("\n      x      s\n=====\\n");
        for (double x=xmin; x<=xmax; x+=dx) {
            double s = 0, p = -1;
            for (int i=1; i<=n; i++) s += (p *= -x);
            System.out.format ("%8.3f%8.3f\\n", x, s);
        }
    }
}
```

```
# java Tabela3
n? 5
xmin, xmax, dx? -1 1 .25

      x      s
=====
-1.000  -5.000
-0.750  -2.288
-0.500  -0.969
-0.250  -0.333
0.000   0.000
0.250   0.200
0.500   0.344
0.750   0.530
1.000   1.000
```

**Zadatak 2.8 Izračunavanje aritmetičke srednje vrednosti i standardne devijacije niza brojeva**

Napisati na jeziku Java program za izračunavanje aritmetičke srednje vrednosti i standardne devijacije niza realnih brojeva. Program treba da čita i obrađuje nizove i ispisuje dobijene rezultate sve dok za broj elemenata niza ne pročita nulu ili negativnu vrednost.

**Rešenje:**

$$s = \frac{1}{n} \sum_{i=1}^n a_i \quad \text{i} \quad d = \sqrt{\frac{1}{n} \sum_{i=1}^n a_i^2 - s^2}$$

```
// Statist.java - Srednja vrednost i standardna devijacija
// niza realnih brojeva.

public class Statist {
    public static void main (String[] args) {
        System.out.print ("\nBroj elemenata niza? "); int n = Citaj.Int ();
        while (n > 0) {
            System.out.print ("Elementi niza? ");
            double s = 0, d = 0;
            for (int i=1; i<=n; i++)
                { double a = Citaj.Double (); s += a; d += a * a; }
            s /= n; d = Math.sqrt (d / n - s * s);
            System.out.println ("Srednja vrednost: " + s);
            System.out.println ("Standardna devijacija: " + d);
            System.out.print ("\\nBroj elemenata niza? "); n = Citaj.Int ();
        }
    }
}
```

```
* java Statist

Broj elemenata niza? 10
Elementi niza? 0 1 2 3 4 5 6 7 8 9
Srednja vrednost: 4.5
Standardna devijacija: 2.8722813232690143

Broj elemenata niza? 10
Elementi niza? 4 4 4 4 4 5 5 5 5 5
Srednja vrednost: 4.5
Standardna devijacija: 0.5

Broj elemenata niza? 10
Elementi niza? 0 0 0 0 0 9 9 9 9 9
Srednja vrednost: 4.5
Standardna devijacija: 4.5

Broj elemenata niza? 0
```

### Zadatak 2.9 Značaj redosleda sabiranja niza realnih brojeva

Napisati na jeziku Java program koji izračunava  $s = \sum_{i=1}^n \frac{1}{i^2}$  po opadajućem i po rastućem redosledu vrednosti sabirska za proizvoljan broj vrednosti parametra  $n$ .

Rešenje:

```
// Redosled.java - Značaj redosleda sabiranja realnih brojeva.

public class Redosled {
    public static void main (String[] varg) {
        while (true) {
            int n = Citaj.Int ();
            if (n <= 0) break;
            float s1 = 0; for (int i=1; i<=n; i++) s1 += 1. / i / i;
            float s2 = 0; for (int i=n; i>=1; i--) s2 += 1. / i / i;
            System.out.println (n + "\t" + s1 + "\t" + s2);
        }
    }
}
```

Prethodni program je pogodan za obradu unapred pripremljenih podataka koji se čitaju (skočivši preko glavnog ulaza) iz neke datoteke – nema pitanja pre čitanja, a pročitani podaci se kasnije ispisuju.

Redosled.pod

100	200	500	1000	2000	5000	10000	20000	50000	100000	200000	500000	1000000	2000000	5000000	0
-----	-----	-----	------	------	------	-------	-------	-------	--------	--------	--------	---------	---------	---------	---

100	1.634984	1.6349839
200	1.6399467	1.6399465
500	1.642936	1.642936
1000	1.6439348	1.6439345
2000	1.6444321	1.6444342
5000	1.6447253	1.6447341
10000	1.6447253	1.644834
20000	1.6447253	1.6448841
50000	1.6447253	1.644914
100000	1.6447253	1.644924
200000	1.6447253	1.644929
500000	1.6447253	1.644932
1000000	1.6447253	1.644933
2000000	1.6447253	1.6449336
5000000	1.6447253	1.6449339

$$\lim_{n \rightarrow \infty} \sum_{i=1}^n \frac{1}{i^2} = \frac{\pi^2}{6} = 1.64493430668$$

Realni brojevi jednostrukе tačnosti imaju 6 - 7 značajnih cifara.  $\Rightarrow 1 + 10^{-8} = 1$

Ako je moguće, nizove brojeva treba sabirati po rastućem redosledu vrednosti elemenata (zbir velikih vrednosti će biti dovoljno velik da utiče na rezultat), a ne po opadajućem redosledu (zbir malih vrednosti može da bude dovoljno velik da utiče na rezultat).

### Zadatak 2.10 Određivanje datuma za naredni dan

Napisati na jeziku Java program za određivanje narednog datuma u odnosu na zadati dan. Program treba da čita datume i da ispisuje rezultate sve dok za jednu od komponenata datuma ne prečita nulu.

**Rešenje:**

```
// Sutra.java - Određivanje datuma sutrašnjeg dana.

public class Sutra {
    public static void main (String[] args) {
        while (true) {
            System.out.print ("Danas? ");
            int dan = Citaj.Int (), mes = Citaj.Int (), god = Citaj.Int ();
            if (dan==0 || mes==0 || god==0) break;

            // Broj dana u tekućem mesecu:
            int d = 0;
            switch (mes) {
                case 1: case 3: case 5: case 7: case 8: case 10: case 12:
                    d = 31; break;
                case 4: case 6: case 9: case 11:
                    d = 30; break;
                case 2:
                    if (god%4==0 && god%100!=0 || god%400==0) d = 29; else d = 28;
                    break;
            }

            // Obrazovanje datuma za sledeći dan:
            if (dan < d) dan++;
            else {
                dan = 1;
                if (mes < 12) mes++;
                else { mes = 1; god++; }
            }
            System.out.println ("Sutra= " + dan + "." + mes + "." + god + "\n");
        }
    }
}
```

Godina je prestupna ako je deljiva sa 4 i ne sa 100. Ipak, ako je deljiva sa 400, ona je prestupna.

```
% java Sutra
Danas? 3 11 2004
Sutra= 4.11.2004

Danas? 30 11 2004
Sutra= 1.12.2004

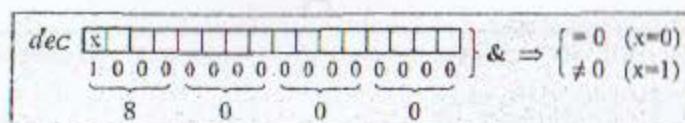
Danas? 31 12 2004
Sutra= 1.1.2005

Danas? 0 0 0
```

### Zadatak 2.11 Ispisivanje celih brojeva u binarnom brojevnom sistemu

Napisati na jeziku Java program koji čita decimalne cele brojeve i ispisuje njihovu vrednost u binarnom obliku. Za interno predstavljanje celih brojeva koristiti po 16 bitova. Program treba da čita brojeve i da ispisuje rezultate sve dok ne pročita vrednost 9999.

**Rešenje:**



```
// Binar.java - Ispisivanje u binarnom brojevnom sistemu.
```

```
public class Binar {
    public static void main (String[] varg) {
        while (true) {
            System.out.print ("Decimalan broj? "); short dec = Citaj.Short ();
            if (dec == 9999) break;
            System.out.print ("Binaran broj:    ");
            for (int i=1; i<=16; i++) {
                int bit = (dec & 0x8000) >>> 15;
                System.out.print (bit);
                if (i % 4 == 0) System.out.print (' ');
                dec <<= 1;
            }
            System.out.println ();
        }
    }
}
```

```
$ java Binar
Decimalan broj? 1
Binaran broj: 0000 0000 0000 0001
Decimalan broj? -1
Binaran broj: 1111 1111 1111 1111
Decimalan broj? 32767
Binaran broj: 0111 1111 1111 1111
Decimalan broj? -32768
Binaran broj: 1000 0000 0000 0000
Decimalan broj? 1234
Binaran broj: 0000 0100 1101 0010
Decimalan broj? -765
Binaran broj: 1111 1101 0000 0011
Decimalan broj? s
```

Upravljanje podacima je jedan od ključnih delova baza podataka. Upravljanje podacima uključuje razne operacije na podacima, uključujući upravljanje podataka u bazi podataka, upravljanje podataka u aplikaciji i upravljanje podataka u mreži. Upravljanje podacima je važno za dobro funkcionisanje baze podataka i aplikacija.

### **3 Nizovi**

**Zadatak 3.1** Tabeliranje vrednosti polinoma

Napisati na jeziku Java program za tabeliranje polinoma koji je zadat pomoću niza koeficijenata, za svaki  $x_{\min} \leq x \leq x_{\max}$  s korakom  $\Delta x$ .

**Rešenje:**

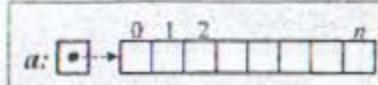
$$p(x) = \sum_{i=0}^n a_i x^i = a_n x^n + a_{n-1} x^{n-1} + \dots + a_1 x + a_0 = (\dots ((a_n x + a_{n-1})x + a_{n-2})x + \dots + a_1)x + a_0$$

```
// Polinom.java - Tabeliranje polinoma zadatog pomoću koeficijenata
```

```

public class Polinom {
    public static void main (String[] varg) {
        System.out.print ("Red polinoma? ");
        int n = Citaj.Int ();
        float[] a = new float [n+1];
        System.out.print ("Koeficijenti? ");
        for (int i=n; i>=0; i--) a[i] = Citaj.Float ();
        System.out.print ("xmin,xmax,dx? ");
        float xmin = Citaj.Float (), xmax = Citaj.Float (), dx = Citaj.Float ();
        System.out.print ("\nx\tp(x)\n=====\\n");
        for (float x=xmin; x<=xmax; x+=dx) {
            float p = a[n];
            for (int i=n-1; i>=0; p=p*x+a[i--]);
            System.out.println (x + "\t" + p);
        }
    }
}

```



```
% java Polinom  
Red polinoma? 3  
Koeficijenti? 1 -2 3 4  
xmin,xmax,dx? -4 4 .5
```

$x$	$p(x)$
-4.0	-104.0
-3.5	-73.875
-3.0	-50.0
-2.5	-31.625
-2.0	-18.0
-1.5	-8.375
-1.0	-2.0
-0.5	1.875
0.0	4.0
0.5	5.125
1.0	6.0
1.5	7.375
2.0	10.0
2.5	14.625
3.0	22.0
3.5	32.875
4.0	48.0

### Zadatak 3.2 Izračunavanje srednje vrednosti elemenata niza

Napisati na jeziku Java program za izračunavanje srednje vrednosti onih elemenata niza celih brojeva koji su deljivi sa tri. Program treba da pročita niz, izvrši traženu obradu, ispiše dobijeni rezultat i ponavlja prethodne korake sve dok za dužinu niza ne pročita nedozvoljenu vrednost.

Rešenje:

```
// SrVredl.java - Srednja vrednost brojeva deljivih sa tri.

public class SrVredl {
    public static void main (String[] varg) {
        while (true) {
            System.out.print ("n? "); int n = Citaj.Int ();
            if (n < 0) break;
            int[] a = new int [n];
            System.out.print ("A? ");
            for (int i=0; i<n; i++) a[i] = Citaj.Int ();
            double s = 0; int k = 0;
            for (int i=0; i<n; i++)
                if (a[i] % 3 == 0 ) { s += a[i]; k++; }
            if (k != 0) s /= k;
            System.out.println ("s= " + s);
        }
    }
}
```

```
% java SrVredl
n? 10
A? 1 2 3 4 5 6 7 8 9 0
s= 4.5
n? 5
A? 3 6 9 12 15
s= 9.0
n? 5
A? 1 1 1 1 1
s= 0.0
n? -1
```

**Zadatak 3.3 Nalaženje vrednosti najmanjeg elementa u nizu**

Napisati program na jeziku Java za nalaženje vrednosti najmanjeg elementa u nizu realnih brojeva. Program treba da čita i obrađuje nizove sve dok za dužinu niza ne pročita nedozvoljenu vrednost.

**Rešenje:**

Prvi element niza se proglaši za najmanji. Posle se traži da li u nizu postoji još manja vrednost.

```
// Min2.java - Najmanji element niza.

public class Min2 {
    public static void main (String[] varg) {
        while (true) {
            System.out.print ("n? "); int n = Citaj.Int ();
            if (n<=0) break;
            double[] a = new double [n];
            System.out.print ("A? "); for (int i=0; i<n; a[i++]=Citaj.Double ());
            double min = a[0];
            for (int i=1; i<n; i++) if (a[i] < min) min = a[i];
            System.out.println ("min= " + min + "\n");
        }
    }
}
```

```
% java Min2
n? 5
A? 8 3 6 2 3
min= 2.0

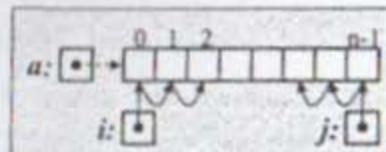
n? -1
```

---

#### Zadatak 3.4 Obrtanje redosleda elemenata niza

Napisati program na jeziku Java za obrtanje redosleda elemenata (tj. za zamenu prvog elementa njim, drugog s pretposlednjim itd.) zadatog niza realnih brojeva. Program treba da čita i obraće sve dok za dužinu niza ne pročita nedozvoljenu vrednost.

**Rešenje:**



```
// Obrni.java - Obrtanje redosleda elemenata niza.

public class Obrni {
    public static void main (String[] varg) {
        while (true) {
            System.out.print ("n? "); int n = Citaj.Int ();
            if (n<=0) break;
            double[] a = new double [n];
            System.out.print ("A? ");
            for (int i=0; i<n; a[i++]=Citaj.Double ());
            for (int i=0, j=n-1; i<j; i++, j--) {
                double b = a[i]; a[i] = a[j]; a[j] = b;
            }
            System.out.print ("A=");
            for (int i=0; i<n; System.out.print (" "+a[i++]));
            System.out.print ("\n\n");
        }
    }
}
```

```
% java Obrni
n? 5
A? 1 2 3 4 5
A= 5.0 4.0 3.0 2.0 1.0

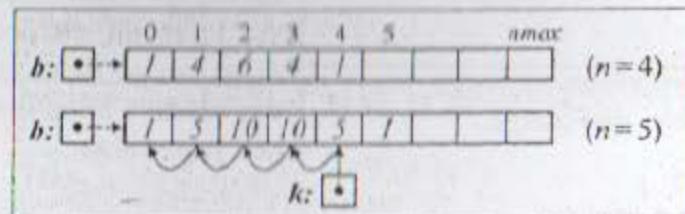
n? 6
A? 1 2 3 4 5 6
A= 6.0 5.0 4.0 3.0 2.0 1.0

n? 0
```

### Zadatak 3.5 Izračunavanje binomnih koeficijenata

Binomni koeficijenti su definisani sa  $B_{n,k} = B_{n-1,k-1} + B_{n-1,k}$  za  $0 < k < n$ , odnosno  $B_{n,0} = B_{n,n} = 1$  ( $n=0, 1, 2, \dots$ ). Napisati na jeziku Java program za ispisivanje svih binomnih koeficijenata za svako  $0 \leq n \leq n_{\max}$ .

**Rešenje:**



```
// BinKo.java - Ispisivanje binomnih koeficijenata.
```

```
public class BinKo {
    public static void main (String[] varg) {
        System.out.print ("nmax? "); int nmax = Citaj.Int ();
        int[] b = new int [nmax+1];
        System.out.println ();
        for (int n=0; n<=nmax; n++) {
            b[n] = 1;
            for (int k=n-1; k>0; k--) b[k] += b[k-1];
            for (int k=0; k<=n; System.out.print (b[k++]+ " "));
            System.out.println ();
        }
    }
}
```

```
% java BinKo
nmax? 12

1
1 1
1 2 1
1 3 3 1
1 4 6 4 1
1 5 10 10 5 1
1 6 15 20 15 6 1
1 7 21 35 35 21 7 1
1 8 28 56 70 56 28 8 1
1 9 36 84 126 126 84 36 9 1
1 10 45 120 210 252 210 120 45 10 1
1 11 55 165 330 462 462 330 165 55 11 1
1 12 66 220 495 792 924 792 495 220 66 12 1
```

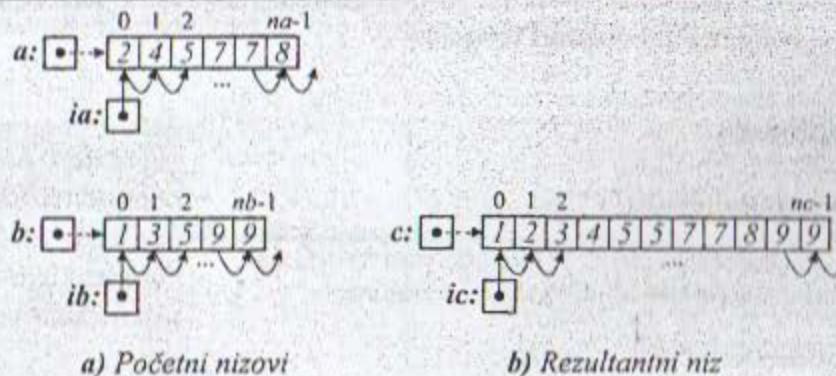
### Zadatak 3.6 Nalaženje fuzije dva uređena niza

Napisati na jeziku Java program za nalaženje fuzije dva niza celih brojeva koji su uređeni po neopadajućem redosledu u treći, na isti način ureden, niz. Program treba da obrađuje proizvoljan broj kompleta ulaznih podataka.

#### Rešenje:

Uporeduju se po jedan element iz oba niza, manji se stavlja na kraj rezultata i uoči se sledeći element niza kome je taj manji pripadao.

Kad se dođe do kraja jednog od nizova, ostali elementi drugog niza se dodaju na kraj rezultata.



a) Početni nizovi

b) Rezultantni niz

```
// Fuzija.java - Fuzija dva uređena niza celih brojeva.

public class Fuzija {
    public static void main (String[] varg) {
        while (true) {
            System.out.print ("na? "); int na = Citaj.Int ();
            if (na < 0) break;
            int[] a = new int [na];
            System.out.print ("A ? "); for (int i=0; i<na; a[i++]=Citaj.Int ());
            if (na == 0) System.out.println ();
            System.out.print ("nb? "); int nb = Citaj.Int ();
            if (nb < 0) break;
            int[] b = new int [nb];
            System.out.print ("B ? "); for (int i=0; i<nb; b[i++]=Citaj.Int ());
            if (nb == 0) System.out.println ();
            int nc = na + nb; int[] c = new int [nc];
            int ia = 0, ib = 0, ic = 0;
            while (ia<na && ib<nb)
                c[ic++] = (a[ia] < b[ib]) ? a[ia++] : b[ib++];
            while (ia < na) c[ic++] = a[ia++];
            while (ib < nb) c[ic++] = b[ib++];
            System.out.print ("C = ");
            for (int i=0; i<nc; System.out.print(c[i++]+ " "));
            System.out.print ("\n\n");
        }
    }
}
```

```
$ java Fuzija
na? 6
A ? 1 2 3 4 5 6
nb? 5
B ? 4 5 6 7 8
C = 1 2 3 4 4 5 5 6 6 7 8
na? 4
A ? 1 2 3 4
nb? 0
```

```
B ?
C = 1 2 3 4
na? 0
A ?
nb? 6
B ? 1 2 3 4 5 6
C = 1 2 3 4 5 6
na? -1
```

### Zadatak 3.7 Uređivanje niza

Napisati na jeziku Java programe za uređivanje niza brojeva po neopadajućem redosledu

- metodom izbora, i
- metodom umetanja.

Program treba da pročita dužinu niza, popuni niz slučajnim jednocifrenim decimalnim celim brojevima, ispiše dobijeni niz, uredi niz, ispiše dobijeni rezultat i ponavlja prethodne korake sve dok za dužinu niza ne pročita nedozvoljenu vrednost.

**Rešenje:**

#### a) Metoda izbora

```
// Uredi2.java - Uređivanje niza
// brojeva metodom izbora.

public class Uredi2 {
    public static void main
        (String[] args) {
        while (true) {

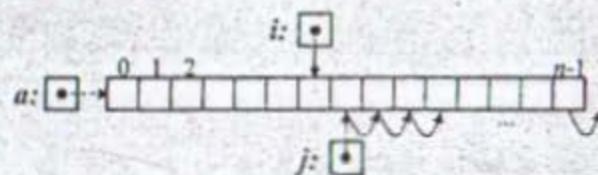
            // Čitanje dužine niza:
            System.out.print ("\nDuzina niza? ");
            int n = Citaj.Int ();
            if (n <= 0) break;

            // Stvaranje i ispisivanje niza:
            int[] a = new int [n]; System.out.print ("\nPocetni niz:\n\n");
            for (int i=0; i<n; i++) {
                System.out.print ((a[i] = (int)(Math.random() * 10)) + " ");
                if (i%30==29 || i==n-1) System.out.println ();
            }

            // Uređivanje niza:
            for (int i=0; i<n-1; i++)
                for (int j=i+1; j<n; j++)
                    if (a[i] > a[j]) { int b = a[i]; a[i] = a[j]; a[j] = b; }

            // Ispisivanje uređenog niza:
            System.out.print ("\nUredjeni niz:\n\n");
            for (int i=0; i<n; i++) {
                System.out.print (a[i] + " ");
                if (i%30==29 || i==n-1) System.out.println ();
            }
        }
    }
}
```

U  $i$ -tom prolazu ( $i=0,1,\dots,n-2$ ) na  $i$ -to mesto se dovodi (bir) najmanji broj među elementima niza od  $i$ -tog mesta do kraja niza. To se postiže upoređivanjem  $a_i$  s elementima iza  $i$ -tog mesta i zamenom elemenata po potrebi.



```
% java Uredi2
```

Duzina niza? 100

Pocetni niz:

```
7 4 6 6 1 2 2 4 6 7 0 3 3 8 3 8 4 1 2 5 5 1 6 0 8 6 3 0 5 2
7 4 3 7 0 5 0 5 4 3 4 6 4 6 8 9 6 0 8 0 8 0 5 8 0 5 5 8 5 5
6 8 6 6 6 9 6 8 9 2 5 4 7 0 9 9 8 2 3 8 9 1 3 8 3 9 6 2 7 6
3 7 8 2 8 6 8 6 6 3
```

Uredjeni niz:

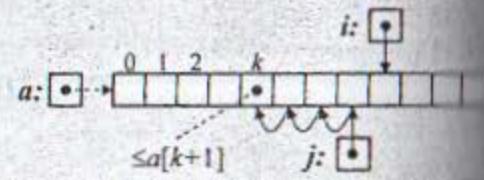
```
0 0 0 0 0 0 0 0 0 1 1 1 1 2 2 2 2 2 2 3 3 3 3 3 3 3 3  
3 3 3 4 4 4 4 4 4 4 5 5 5 5 5 5 5 5 5 6 6 6 6 6 6 6 6  
6 6 6 6 6 6 6 6 6 7 7 7 7 7 7 7 8 8 8 8 8 8 8 8 8 8 8 8  
8 8 8 9 9 9 9 9 9 9 9
```

Duzina niza? -1

### b) Metoda umetanja

```
// Uredi3.java - Uredjivanje niza  
// brojeva metodom umetanja.  
  
public class Uredi3 {  
    public static void main  
        (String[] args) {  
    while (true) {  
  
        // Čitanje dužine niza:  
        System.out.print ("\nDuzina niza? "); int n = Citaj.Int ();  
        if (n <= 0) break;  
  
        // Stvaranje i ispisivanje niza:  
        int[] a = new int [n]; System.out.print ("\nPocetni niz:\n\n");  
        for (int i=0; i<n; i++) {  
            System.out.print ((a[i] = (int)(Math.random() * 10)) + " ");  
            if (i%30==29 || i==n-1) System.out.println ();  
        }  
  
        // Uredjivanje niza:  
        for (int i=1; i<n; i++)  
            for (int j=i-1; j>=0 && a[j]>a[j+1]; j--)  
                { int b = a[j]; a[j] = a[j+1]; a[j+1] = b; }  
  
        // Ispisivanje uređenog niza:  
        System.out.print ("\nUredjeni niz:\n\n");  
        for (int i=0; i<n; i++) {  
            System.out.print (a[i] + " ");  
            if (i%30==29 || i==n-1) System.out.println ();  
        }  
    }  
}
```

U  $i$ -tom prolazu ( $i=1,2,\dots,n-1$ )  $i$ -ti element se uvrstava na odgovarajuće mesto između već uredenih elemenata. To se postiže razmjenom po dva uzastopna elementa tog mesta i njihovom medusobnom zamjenom dok je potrebno.



% java Uredi3

Duzina niza? 25

Pocetni niz:

```
7 1 1 9 9 2 7 5 5 1 0 1 7 8 6 4 4 4 3 0 6 6 3 9 4
```

Uredjeni niz:

```
0 0 1 1 1 1 2 3 3 4 4 4 4 5 5 6 6 6 7 7 7 8 9 9 9
```

Duzina niza? -1

### Zadatak 3.8 Umetanje niza u drugi niz

Napisati na jeziku Java program za umetanje u niz celih brojeva sadržaja drugog takvog niza, počev od zadate pozicije (ispred prvog elementa ako je pozicija manja od nule, odnosno iza poslednjeg elementa ako je pozicija veća od dužine odredišnog niza). Program treba da obrađuje proizvoljan broj kompleta podataka.

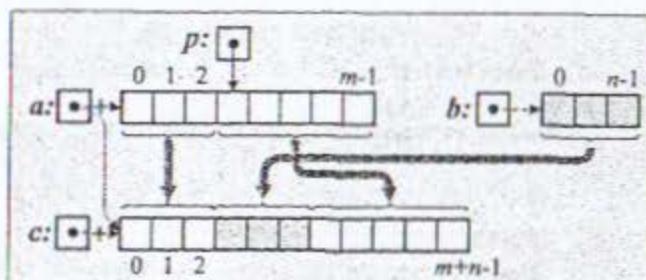
**Rešenje:**

```
// Umetni.java - Umetanje elemenata niza u drugi niz.

public class Umetni {
    public static void main (String[] args) {
        while (true) {
            System.out.print ("m? "); int m = Citaj.Int ();
            if (m < 0) break;
            int[] a = new int [m];
            System.out.print ("A? "); for (int i=0; i<m; a[i++]=Citaj.Int ());
            if (m == 0) System.out.println ();
            System.out.print ("n? "); int n = Citaj.Int ();
            if (n < 0) break;
            int[] b = new int [n];
            System.out.print ("B? "); for (int j=0; j<n; b[j++]=Citaj.Int ());
            if (n == 0) System.out.println ();
            System.out.print ("p? "); int p = Citaj.Int ();
            if (p < 0) p = 0;
            if (p > m) p = m;

            int[] c = new int [m+n];
            int i = 0, j = 0, k = 0;
            while (i < p) c[k++] = a[i++];
            while (j < n) c[k++] = b[j++];
            while (i < m) c[k++] = a[i++];
            a = c; c = null; m += n;

            System.out.print ("A= ");
            for (i=0; i<m; System.out.print(a[i++]+ " "));
            System.out.print ("\n\n");
        }
    }
}
```



```
% java Umetni
m? 5
A? 0 1 2 3 4
n? 3
B? 11 22 33
p? 2
A= 0 1 11 22 33 2 3 4

m? 5
A? 0 1 2 3 4
n? 3
B? 11 22 33
p? -1
A= 11 22 33 0 1 2 3 4
```

```
p? -2
A= 11 22 33 0 1 2 3 4

m? 5
A? 0 1 2 3 4
n? 3
B? 11 22 33
p? 8
A= 0 1 2 3 4 11 22 33

m? -1
```

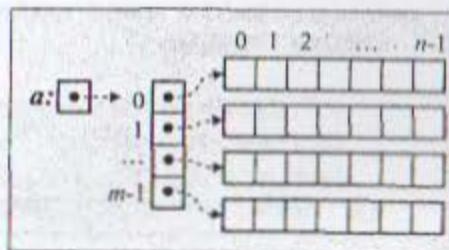
### Zadatak 3.9 Međusobna zamena najmanjeg i najvećeg elementa matrice

Napisati na jeziku Java program za međusobnu zamenu najmanjeg i najvećeg elementa date matrice realnih brojeva. Program treba da obrađuje proizvoljan broj kompleta podataka.

**Rešenje:**

```
// MinMax.java - Međusobna zamena najmanjeg i najvećeg elementa matrice.

public class MinMax {
    public static void main (String[] args) {
        while (true) {
            System.out.print ("m, n? ");
            int m = Citaj.Int (), n = Citaj.Int ();
            if (m<=0 || n<=0) break;
            float[][] a = new float [m][n];
            for (int i=0; i<m; i++) {
                System.out.print (i+1 + ". vrsta? ");
                for (int j=0; j<n; a[i][j++]=Citaj.Float ());
            }
            float min = a[0][0], max = min;
            int imin = 0, jmin = 0, imax = 0, jmax = 0;
            for (int i=0; i<m; i++)
                for (int j=0; j<n; j++)
                    if (a[i][j] < min) { min = a[i][j]; imin = i; jmin = j; }
                    else if (a[i][j] > max) { max = a[i][j]; imax = i; jmax = j; }
            float b = a[imin][jmin];
            a[imin][jmin] = a[imax][jmax];
            a[imax][jmax] = b;
            System.out.println ();
            for (int i=0; i<m; i++) {
                for (int j=0; j<n; System.out.print(a[i][j++]+ " "));
                System.out.println ();
            }
            System.out.println ();
        }
    }
}
```



```
% java MinMax
m, n? 3 4
1. vrsta? 1 2 3 4
2. vrsta? 2 3 4 5
3. vrsta? 3 4 5 6
6.0 2.0 3.0 4.0
2.0 3.0 4.0 5.0
3.0 4.0 5.0 1.0
m, n? 0 0
```

### Zadatak 3.10 Transponovanje pravougaone matrice

Napisati na jeziku Java program za transponovanje pravougaone matrice celih brojeva. Program treba da obrađuje proizvoljan broj kompleta podataka.

**Rešenje:**

```
// Transpon.java - Transponovanje matrice.

public class Transpon {
    public static void main (String[] varg) {
        while (true) {

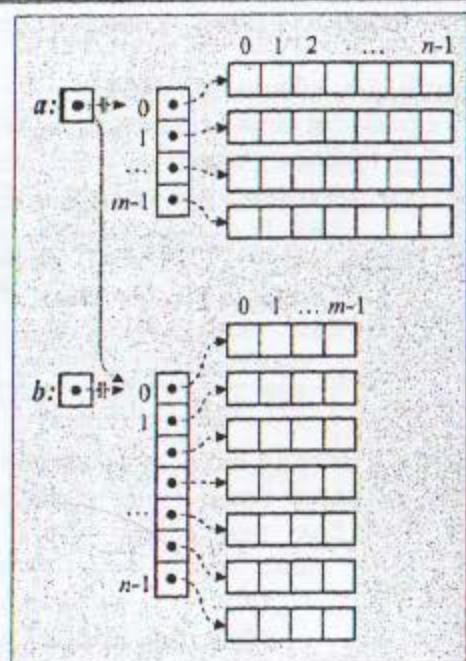
            // Čitanje dimenzija matrice:
            System.out.print ("\nBroj vrsta i kolona? ");
            int m = Citaj.Int (), n = Citaj.Int ();
            if (m<=0 || n<=0) break;

            // Čitanje elemenata matrice:
            int[][] a = new int [m][n];
            for (int i=0; i<m; i++) {
                System.out.print (i+1 + ". vrsta? ");
                for (int j=0; j<n; a[i][j++]=Citaj.Int ());
            }

            // Obrazovanje transponovane matrice:
            int[][] b = new int [n][m];
            for (int i=0; i<n; i++)
                for (int j=0; j<m; j++) b[i][j] = a[j][i];

            // Zamena stare matrice novom matricom:
            a = b; b = null; int p = m; m = n; n = p;

            // Ispisivanje rezultata:
            System.out.print ("\nTransponovana matrica:\n");
            for (int i=0; i<m; i++) {
                for (int j=0; j<n; System.out.print (a[i][j++]+ " "));
                System.out.println ();
            }
        }
    }
}
```



```
% java Transpon

Broj vrsta i kolona? 3 4
1. vrsta? 1 2 3 4
2. vrsta? 5 6 7 8
3. vrsta? 9 0 1 2

Transponovana matrica:
1 5 9
2 6 0
3 7 1
4 8 2

Broj vrsta i kolona? 0 0
```

## **4 Klase**

### Zadatak 4.1 Tačke u ravni

Napisati na jeziku Java klasu tačaka u ravni. Predviđeni postavljanje koordinata, dohvatanje koordinata, izračunavanje rastojanja do zadate tačke, čitanje tačke, pisanje tačke i glavnu funkciju za ispitivanje klase.

**Rešenje:**

```
// Tackal.java - Klasa tačaka u ravni.

public class Tackal {
    private double x, y; // Koordinate.

    public void postavi (double a, double b) // Postavljanje
    { x = a; y = b; } // koordinata.

    public double x () { return x; } // Apscisa.

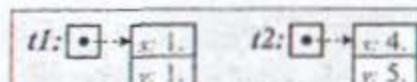
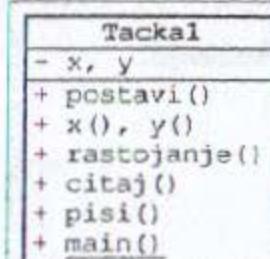
    public double y () { return y; } // Ordinata.

    public double rastojanje (Tackal t) // Rastojanje do tačke.
    { return Math.sqrt (Math.pow (x-t.x, 2) + Math.pow (y-t.y, 2)); }

    public void citaj () // Čitanje tačke.
    { x = Citaj.Double (); y = Citaj.Double (); }

    public void pisi () // Pisanje tačke.
    { System.out.print ("(" + x + "," + y + ")"); }

    public static void main (String[] varg) { // Ispitivanje klase tačaka.
        System.out.print ("t1? ");
        double x = Citaj.Double (),
               y = Citaj.Double ();
        Tackal t1 = new Tackal (); t1.postavi (x, y);
        System.out.print ("t2? ");
        Tackal t2 = new Tackal (); t2.citaj ();
        System.out.print ("t1=" + t1.x () + "," + t1.y () + ", t2=");
        t2.pisi (); System.out.println ();
        System.out.println ("Rastojanje=" + t1.rastojanje (t2));
    }
}
```



```
$ java Tackal
t1? 1 1
t2? 4 5
t1=(1.0,1.0), t2=(4.0,5.0)
Rastojanje=5.0
```

### Zadatak 4.2 Kompleksni brojevi

Napisti na jeziku Java klasu kompleksnih brojeva. Predvideti:

- stvaranje kompleksnog broja,
- dohvatanje delova kompleksnog broja,
- postavljanje delova kompleksnog broja,
- izračunavanje absolutne vrednosti i argumenta kompleksnog broja,
- nalaženje konjugovanog kompleksnog broja,
- izvođenje aritmetičkih operacija,
- izračunavanje  $e^x$ ,  $\log x$  i  $x^y$ ,
- čitanje kompleksnog broja,
- sastavljanje tekstualnog oblika kompleksnog broja, i
- glavnu funkciju za ispitivanje klase.

Rešenje:

```
// Kompl1.java - Klasa kompleks-
// nih brojeva.

public class Kompl1 {
    private double re, im;

    public Kompl1 () {} // Stvaranje kompleksnog broja.

    public Kompl1 (double r) { re = r; }

    public Kompl1 (double r, double i) { re = r; im = i; }

    public double re () { return re; } // Realni deo.

    public double im () { return im; } // Imaginarni deo.

    public double abs () // Apsolutna vrednost.
        { return Math.sqrt (re*re + im*im); }

    public double arg () // Argument.
        { return (re!=0 || im!=0) ? Math.atan2(im,re) : 0; }

    public Kompl1 postavi (double r, double i) // Postavljanje delova.
        { re = r; im = i; return this; }

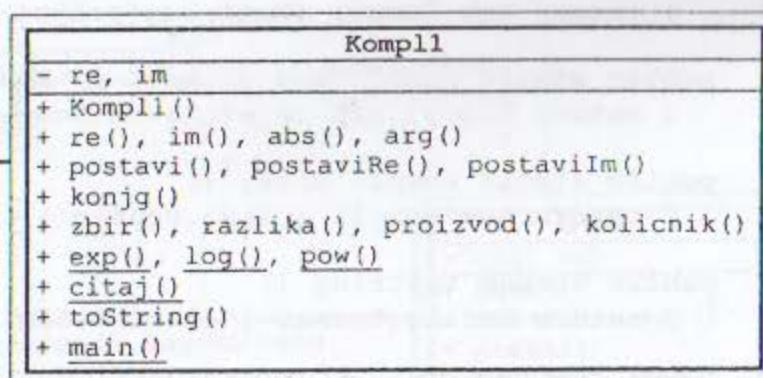
    public Kompl1 postaviRe (double r)
        { re = r; return this; }

    public Kompl1 postaviIm (double i)
        { im = i; return this; }

    public Kompl1 postavi (Kompl1 z)
        { re = z.re; im = z.im; return this; }

    public Kompl1 konjg () // Konjugovan broj.
        { return new Kompl1 (re, -im); }

    public Kompl1 zbir (Kompl1 b) // a += b
        { return new Kompl1 (re+b.re, im+b.im); }
```



```

public Kompl1 razlika (Kompl1 b) // a - b
    { return new Kompl1 (re-b.re, im-b.im); }

public Kompl1 proizvod (Kompl1 b) // a * b
    { return new Kompl1 (re*b.re-im*b.im, im*b.re+re*b.im); }

public Kompl1 kolicnik (Kompl1 b) { // a / b
    double c = b.re*b.re + b.im*b.im;
    return new Kompl1 ((re*b.re+im*b.im)/c, (im*b.re-re*b.im)/c);
}

public static Kompl1 exp (Kompl1 a) { // exp (a)
    double abs = Math.exp (a.re), arg = a.im;
    return new Kompl1 (abs*Math.cos(arg), abs*Math.sin(arg));
}

public static Kompl1 log (Kompl1 a) // log (a)
    { return new Kompl1 (Math.log(a.abs()), a.arg()); }

public static Kompl1 pow (Kompl1 a, Kompl1 b) // pow (a, b)
    { return Kompl1.exp (b.proizvod(Kompl1.log(a))); }

public static Kompl1 citaj () // Čitanje.
    { return new Kompl1 (Citaj.Double(), Citaj.Double()); }

public String toString () // Tekstualni oblik.
    { return String.format ("(%f,%f)", re, im); }

// Glavna funkcija za ispitivanje klase Kompl1. =====
public static void main (String[] varg) {
    System.out.print ("Prvi broj (re,im)? ");
    Kompl1 x = new Kompl1 (Citaj.Double(), Citaj.Double());
    System.out.print ("Drugi broj (re,im)? ");
    Kompl1 y = Kompl1.citaj ();
    System.out.println ("x = (" + x.re() + "," + x.im() + ")");
    System.out.println ("y = " + y);
    System.out.println ("x+y = " + x.zbir (y));
    System.out.println ("x-y = " + x.razlika (y));
    System.out.println ("x*y = " + x.proizvod (y));
    System.out.println ("x/y = " + x.kolicnik (y));
    System.out.println ("exp(x) = " + Kompl1.exp (x));
    System.out.println ("log(x) = " + Kompl1.log (x));
    System.out.println ("x^y = " + Kompl1.pow (x, y));
}
}

```

```

% javac Kompl1.java
% java Kompl1
Prvi broj (re,im)? 1 2
Drugi broj (re,im)? 3 4
x = (1.0,2.0)
y = (3.000,4.000)
x+y = (4.000,6.000)
x-y = (-2.000,-2.000)
x*y = (-5.000,10.000)
x/y = (0.440,0.080)
exp(x) = (-1.131,2.472)
log(x) = (0.805,1.107)
x^y = (0.129,0.034)

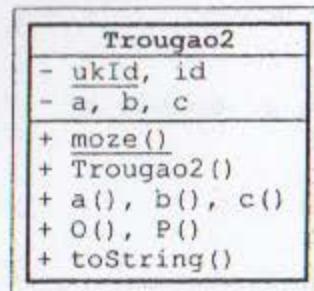
```

### Zadatak 4.3 Numerisani trouglovi

Napisati na jeziku Java klasu trouglova od kojih svaki mora da ima različit, automatski generisani identifikacioni broj. Predvideti ispitivanje da li tri duži mogu biti stranice trougla, stvaranje trougla sa zadatim dužinama, dohvatanje dužina stranica, izračunavanje obima, izračunavanje površine i ispis tekstualnog oblika trougla.

Napisati na jeziku Java program (klasu s glavnom funkcijom) koji pročita dinamički niz trouglova, niz po neopadajućem redosledu površina trouglova i ispiše dobijeni rezultat.

Rešenje:



```
// Trougao2.java - Klasa trouglova.

public class Trougao2 {

    private static int ukId = 0; // Poslednje korišćeni identifikator.
    private int id = ++ukId; // Identifikator trougla.
    private double a, b, c; // Stranice trougla.

    // Da li su stranice prihvatile vrednosti?
    public static boolean moze (double a, double b, double c)
    { return a>0 && b>0 && c>0 && a+b>c && b+c>a && c+a>b; }

    public Trougao2 (double aa, double bb, double cc) // Stvaranje.
    {
        if (! moze (aa, bb, cc)) System.exit (1);
        a = aa; b = bb; c = cc;
    }

    public double a () { return a; } // Dohvatanje stranica.
    public double b () { return b; }
    public double c () { return c; }

    public double O () { return a + b + c; } // Obim trougla.

    public double P () { // Površina trougla.
        double s = O () / 2;
        return Math.sqrt (s * (s-a) * (s-b) * (s-c));
    }

    public String toString () // Tekstualni oblik.
    { return "Troug" + id + "(" + a + "," + b + "," + c + ")"; }
}
```

```
// Trougao2T.java - Ispitivanje klase trouglova.

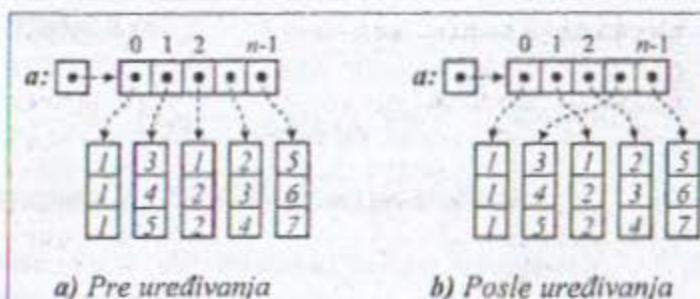
public class Trougao2T {
    public static void main (String[] args) {
        System.out.print ("Broj trouglova? "); int n = Citaj.Int ();
        Trougao2[] niz = new Trougao2 [n];
        for (int i=0; i<n; ) {
            System.out.print (i+1 + ". trougao? ");
            double a = Citaj.Double (), b = Citaj.Double (), c = Citaj.Double ();
            if (Trougao2.moze (a,b,c)) niz[i++] = new Trougao2 (a, b, c);
            else System.out.println ("*** Neprihvativne stranice!");
        }
        for (int i=0; i<n-1; i++)
            for (int j=i+1; j<n; j++)
                if (niz[j].P () < niz[i].P ())
                    { Trougao2 pom = niz[i]; niz[i] = niz[j]; niz[j] = pom; }
        System.out.print ("\nUredjeni niz trouglova:\n");
        for (int i=0; i<n; i++)
            System.out.println (i+1 + ": " + niz[i] + " P=" + niz[i].P ());
    }
}
```

Dovoljno je tražiti prevodenje samo klase s glavnom funkcijom. Sve ostale korišćene klase biće automatski prevedene. Ako već postoji prevod neke od potrebnih klasa, koristiće se taj prevod.

```
% javac Trougao2T.java
% java Trougao2T
Broj trouglova? 5
1. trougao? 1 1 1
2. trougao? 3 4 5
3. trougao? 1 2 2
4. trougao? 1 3 5
*** Neprihvativne stranice!
4. trougao? 2 3 4
5. trougao? 5 6 7
```

Uredjeni niz trouglova:

```
1: Troug1(1.0,1.0,1.0) P=0.4330127018922193
2: Troug3(1.0,2.0,2.0) P=0.9682458365518543
3: Troug4(2.0,3.0,4.0) P=2.9047375096555625
4: Troug2(3.0,4.0,5.0) P=6.0
5: Troug5(5.0,6.0,7.0) P=14.696938456699069
```



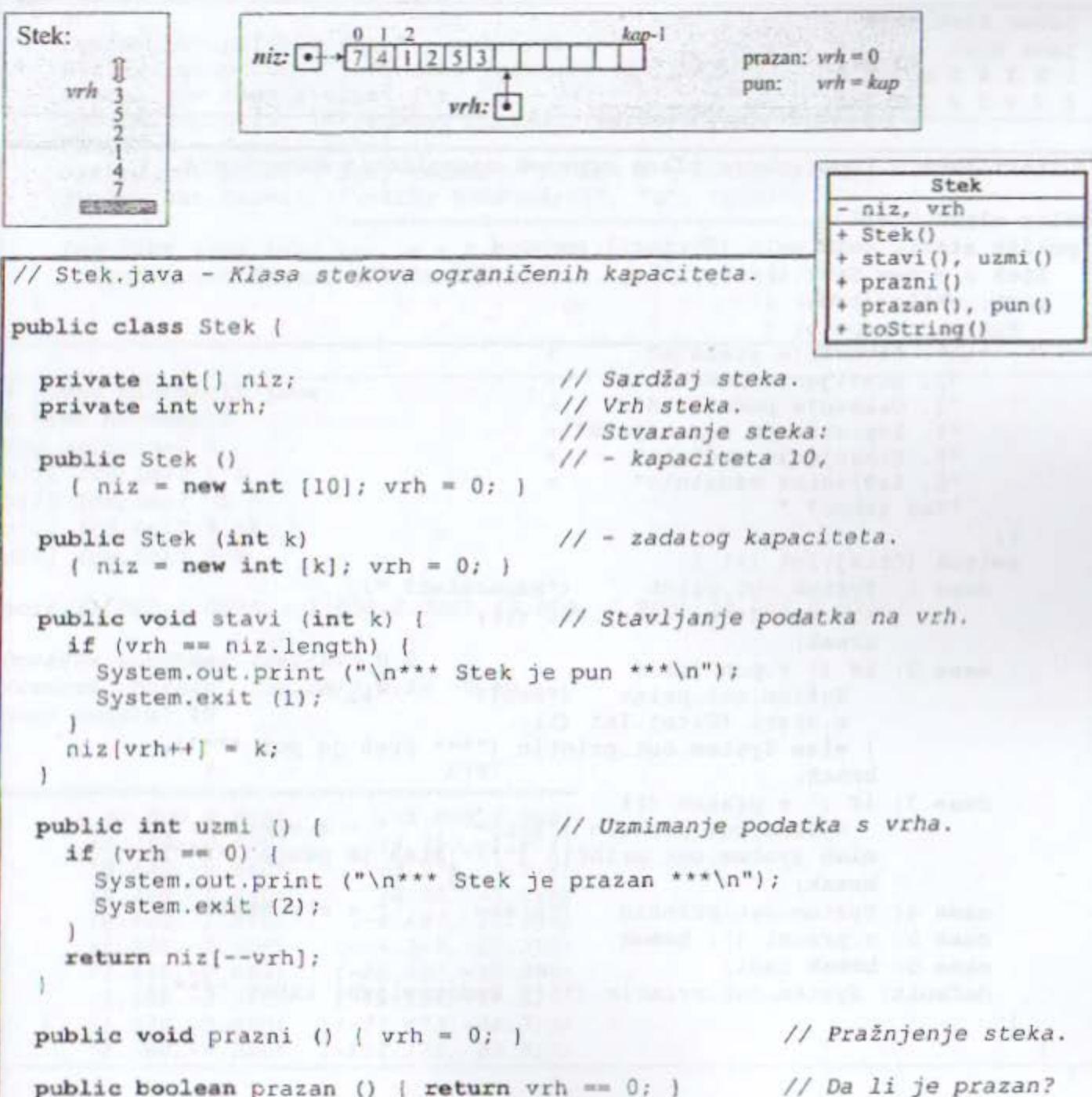
#### Zadatak 4.4 Stekovi ograničenih kapaciteta

Stekovi su linearne strukture podataka kod kojih se podaci dodaju i uzimaju na istom kraju. Mogu imati ograničene ili neograničene kapacitete. Napisati na jeziku Java klasu stekova celih brojeva ograničenih kapaciteta. Predvideti:

- stvaranje praznog steka sa zadatim kapacitetom (podrazumevano 10),
- dodavanje jednog podatka i uzimanje jednog podatka,
- ispitivanje da li je stek pun i da li je prazan,
- pražnjenje steka, i
- sastavljanje tekstualnog oblika steka.

Napisati na jeziku Java interaktivni program za ispitivanje prethodne klase.

**Rešenje:**



```

public boolean pun () { return vrh == miz.length; } // Da li je pun?

public String toString () { // Tekstualni oblik
    String s = "";
    for (int i=vrh; i>0; s+=miz[--i]+" ");
    return s;
}

// Glavna funkcija za ispitivanje klase Stek.
public static void main (String[] varg) {
    Stek s = new Stek ();
    while (! s.pun ()) { s.stavi (Citaj.Int()); }
    while (! s.prazan ()) { System.out.print (s.uzmi() + " "); }
    System.out.println ();
}
}

```

```

% javac Stek.java
% java Stek
0 1 2 3 4 5 6 7 8 9
9 8 7 6 5 4 3 2 1 0

```

```
// StekT.java - Ispitivanje klase stekova ograničenih kapaciteta.
```

```

public class StekT {
    public static void main (String[] varg) {
        Stek s = new Stek ();
        radi: while (true) {
            System.out.print (
                "\n1. Stvaranje steka\n" +
                "2. Stavljanje podatka\n" +
                "3. Uzimanje podatka\n" +
                "4. Ispisivanje sadrzaja\n" +
                "5. Praznjenje steka\n" +
                "0. Zavrsetak rada\n\n" +
                "Vas izbor? "
            );
            switch (Citaj.Int ()) {
                case 1: System.out.print ("Kapacitet? ");
                    s = new Stek (Citaj.Int ());
                    break;
                case 2: if (! s.pun ()) {
                    System.out.print ("Broj? ");
                    s.stavi (Citaj.Int ());
                } else System.out.println ("*** Stek je pun ***");
                    break;
                case 3: if (! s.prazan ()) {
                    System.out.println ("Broj= " + s.uzmi ());
                } else System.out.println ("*** Stek je prazan ***");
                    break;
                case 4: System.out.println ("Stek= " + s); break;
                case 5: s.prazni (); break;
                case 0: break radi;
                default: System.out.println ("*** Nedozvoljeni izbor ***");
            }
        }
    }
}

```

```
% javac StekT.java  
% java StekT  
  
1. Stvaranje steka  
2. Stavljanje podatka  
3. Uzimanje podatka  
4. Ispisivanje sadrzaja  
5. Praznjenje steka  
0. Zavrsetak rada  
  
Vas izbor? 1  
Kapacitet? 3  
...  
Vas izbor? 2  
Broj? 1  
...  
Vas izbor? 2  
Broj? 2  
...  
Vas izbor? 2  
Broj? 3  
...
```

```
Vas izbor? 2  
*** Stek je pun ***  
...  
Vas izbor? 4  
Stek= 3 2 1  
...  
Vas izbor? 3  
Broj= 3  
...  
Vas izbor? 3  
Broj= 2  
...  
Vas izbor? 3  
Broj= 1  
...  
Vas izbor? 3  
*** Stek je prazan ***  
...  
Vas izbor? 55  
*** Nedozvoljeni izbor ***  
...  
Vas izbor? 0
```

### Zadatak 4.5 Nizovi kompleksnih brojeva

Napisati na jeziku Java klasu nizova kompleksnih brojeva. Predviđeti:

- stvaranje niza zadate dužine (podrazumevano 10) s elementima jednakim nuli,
- dohvatanje dužine niza,
- postavljanje i dohvatanje vrednosti zadatog elementa niza,
- sastavljanje tekstualnog oblika niza,
- izračunavanje vrednosti polinoma, čiji su koeficijenti elementi niza, za zadatu vrednost nezavisne promenljive, i
- sastavljanje tekstualnog oblika niza kada se on tumači kao polinom.

Napisati na jeziku Java program za ispitivanje prethodne klase.

**Rešenje:**

```
// NizKompl.java - Klasa
// nizova kompleksnih
// brojeva.

public class NizKompl {
    private Kompl[] niz; // Elementi niza.

    public NizKompl (int n) { // Stvaranje niza:
        niz = new Kompl [n];
        for (int i=0; i<n; niz[i++] = new Kompl());
    }

    public NizKompl () { this (10); } // - dužine 10.

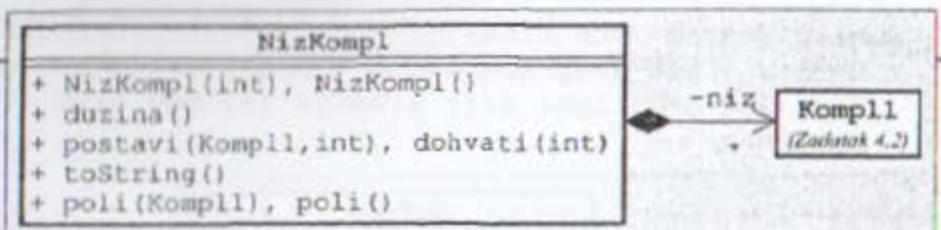
    public int duzina () { return niz.length; } // Dohvatanje dužine.

    public NizKompl postavi (Kompl z, int i) { // Postavljanje elementa.
        if (i<0 || i>=niz.length) System.exit (1);
        niz[i] = new Kompl (z.re(), z.im());
        return this;
    }

    public Kompl dohvati (int i) { // Dohvatanje elementa.
        if (i<0 || i>=niz.length) System.exit (1);
        return niz[i];
    }

    public String toString () { // Tekstualni oblik niza.
        String s = "";
        for (int i=0; i<niz.length; s+=niz[i++]+ " ");
        return s;
    }

    public Kompl poli (Kompl z) { // Vrednost polinoma.
        Kompl p = new Kompl ();
        for (int i=niz.length-1; i>=0; i--)
            p = p.proizvod(z).zbit(niz[i]);
        return p;
    }
}
```



```

public String poli () { // Tekstualni oblik polinoma.
    String s = "poli[";

    for (int i=niz.length-1; i>=0; i--)
        { s += niz[i]; if (i > 0) s += ","; }

    return s + "]";
}
}

```

// NizKomplT.java - Ispitivanje klase niza kompleksnih brojeva.

```

public class NizKomplT {
    public static void main (String[] varg) {
        System.out.print ("Red polinoma? "); int n = Citaj.Int ();
        NizKompl poli = new NizKompl (n+1);
        for (int i=n; i>=0; i--) {
            System.out.print ("p[" + i + "] (re,im)? ");
            poli.postavi (Kompl1.citaj(), i);
        }
        System.out.println ("\n" + poli.poli());
        System.out.print ("\nPocetna vrednost (re,im)? ");
        Kompl1 z = Kompl1.citaj ();
        System.out.print ("Vrednost koraka (re,im)? ");
        Kompl1 d = Kompl1.citaj ();
        System.out.print ("Broj tacaka? "); int k = Citaj.Int ();
        System.out.format ("\n%15s %20s\n", "z", "p(z)",
                           "=====");
        for (int i=0; i<k; i++, z = z.zbir(d))
            System.out.format ("%20s %20s\n", z, poli.poli(z));
    }
}

```

```

$ javac NizKomplT.java
$ java NizKomplT
Red polinoma? 3
p[3] (re,im)? 1 1
p[2] (re,im)? -1 0
p[1] (re,im)? 3 -2
p[0] (re,im)? 5 0

poli{{1.000,1.000),(-1.000,0.000),(3.000,-2.000),(5.000,0.000) }

Pocetna vrednost (re,im)? 0 0
Vrednost koraka (re,im)? 0.24 -0.48
Broj tacaka? 10

      z          p(z)
=====
(0.000,0.000)      (5.000,0.000)
(0.240,-0.480)     (4.753,-1.814)
(0.480,-0.960)     (3.774,-3.914)
(0.720,-1.440)     (0.983,-7.046)
(0.960,-1.920)     (-4.697,-11.956)
(1.200,-2.400)     (-14.344,-19.392)
(1.440,-2.880)     (-29.037,-30.099)
(1.680,-3.360)     (-49.854,-44.825)
(1.920,-3.840)     (-77.873,-64.315)
(2.160,-4.320)     (-114.173,-89.317)

```

### Zadatak 4.6 Uređeni skupovi brojeva

Napisati na jeziku Java klasu uređenih skupova realnih brojeva. Predviđeti:

- stvaranje praznog skupa, skupa koji sadrži jedan broj,
- nalaženje unije, preseka i razlike dva skupa,
- sastavljanje tekstualnog oblika skupa,
- čitanje skupa s glavnog ulaza, i
- dohvatanje broja elemenata skupa.

Napisati na jeziku Java klasu program za ispitivanje prethodne klase.

**Rešenje:**

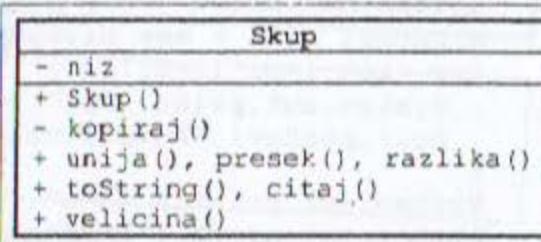
```
// Skup.java - Klasa uređenih skupova.

class Skup {
    private double[] niz; // Elementi skupa.
                           // Stvaranje skupa:
    public Skup () { niz = new double [0]; } // - praznog,
                                             
    public Skup (double b)                      // - od jednog broja.
        { niz = new double [1]; niz[0] = b; }

    private void kopiraj (double[] a, int n) { // Kopiranje niza u skup.
        niz = new double [n];
        for (int i=0; i<n; niz[i]=a[i++]);
    }

    public void unija (Skup s1, Skup s2) { // Unija dva skupa.
        double[] a = new double [s1.niz.length+s2.niz.length]; int n = 0;
        for (int i=0, j=0; i<s1.niz.length || j<s2.niz.length; ) {
            a[n++] = i==s1.niz.length ? s2.niz[j++] :
                       j==s2.niz.length ? s1.niz[i++] :
                       s1.niz[i]<s2.niz[j] ? s1.niz[i++] :
                       s1.niz[i]>s2.niz[j] ? s2.niz[j++] :
                                         s1.niz[i++];
            if (j<s2.niz.length && s2.niz[j]==a[n-1]) j++;
        }
        kopiraj (a, n);
    }

    public void presek (Skup s1, Skup s2) { // Presek dva skupa.
        double[] a = new double [s1.niz.length<s2.niz.length ?
                               s1.niz.length : s2.niz.length];
        int n = 0;
        for (int i=0, j=0; i<s1.niz.length && j<s2.niz.length; )
            if (s1.niz[i] < s2.niz[j]) i++;
            else if (s1.niz[i] > s2.niz[j]) j++;
            else
                a[n++] = s1.niz[i++]; j++;
        kopiraj (a, n);
    }
}
```



```

public void razlika (Skup s1, Skup s2) { // Razlika dva skupa.
    double[] a = new double [s1.niz.length]; int n = 0;
    for (int i=0, j=0; i<s1.niz.length; )
        if (j == s2.niz.length) a[n++] = s1.niz[i++];
        else if (s1.niz[i] < s2.niz[j]) a[n++] = s1.niz[i++];
        else if (s1.niz[i] > s2.niz[j]) j++;
        else { i++; j++; }
    kopiraj (a, n);
}

public String toString () { // Tekstualni oblik skupa.
    String s = "{";
    for (int i=0; i<niz.length; i++)
        { s += niz[i]; if (i < niz.length-1) s += ","; }
    return s + "}";
}

public static Skup citaj () { // Čitanje skupa.
    Skup s = new Skup (); int n = Citaj.Int ();
    for (int i=0; i<n; i++) s.unija (s, new Skup (Citaj.Double()));
    return s;
}

public int velicina () { return niz.length; } // Veličina skupa.
}

```

// SkupT.java - Ispitivanje klase uređenih skupova.

```

public class SkupT {
    public static void main (String[] varg) {
        char jos;
        do {
            System.out.print ("niz1? "); Skup s1 = Skup.citaj ();
            System.out.print ("niz2? "); Skup s2 = Skup.citaj ();
            System.out.println ("s1 = " + s1);
            System.out.println ("s2 = " + s2);
            Skup s = new Skup ();
            s.unija (s1, s2); System.out.println ("s1+s2=" + s );
            s.presek (s1, s2); System.out.println ("s1*s2=" + s );
            s.razlika (s1, s2); System.out.println ("s1-s2=" + s );
            System.out.print ("\nJos? "); jos = Citaj.Char ();
        } while (jos=='d' || jos=='D');
    }
}

```

```

% javac SkupT
% java SkupT
niz1? 4 1 2 3 4
niz2? 5 3 4 5 6 7
s1 ={1.0, 2.0, 3.0, 4.0}
s2 ={3.0, 4.0, 5.0, 6.0, 7.0}
s1+s2={1.0, 2.0, 3.0, 4.0, 5.0, 6.0, 7.0}
s1*s2={3.0, 4.0}
s1-s2={1.0, 2.0}

```

<pre>Jos? d niz1? 6 9 3 5 1 7 3 niz2? 4 6 2 8 8 s1 ={1.0, 3.0, 5.0, 7.0, 9.0} s2 ={2.0, 6.0, 8.0} s1+s2={1.0, 2.0, 3.0, 5.0, 6.0, 7.0} s1*s2={} s1-s2={1.0, 3.0, 5.0, 7.0, 9.0}</pre>
<pre>Jos? n</pre>

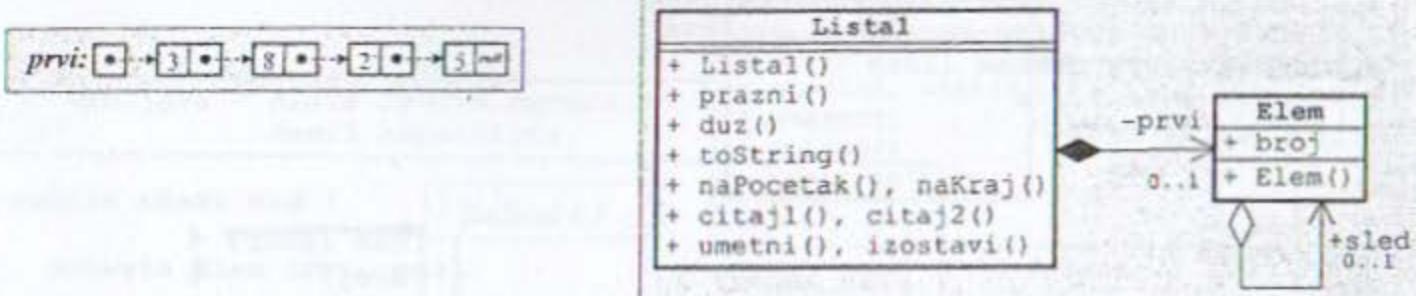
### Zadatak 4.7 Liste brojeva

Napisati na jeziku Java klasu listi celih brojeva. Previđeni:

- stvaranje prazne liste i liste od jednog broja,
- pražnjenje liste,
- određivanje broja elemenata liste,
- sastavljanje tekstualnog oblika liste,
- dodavanje broja na početak liste,
- dodavanje broja na kraj liste,
- čitanje liste s glavnog ulaza dodajući brojeve na početak liste,
- čitanje liste s glavnog izlaza dodajući brojeve na kraj liste,
- umetanje broja u uređenu listu,
- brisanje svih elemenata liste, i
- izostavljanje iz liste svakog pojavljivanja datog broja.

Napisati na jeziku Java interaktivni program (s menijem) za ispitivanje prethodne klase.

Rešenje:



// Elem.java - Element liste celih brojeva.

```

public class Elemt {
    public int broj; // Sadržaj.
    public Elemt sled; // Sledeći element.
    public Elemt (int b, Elemt s) // Inicijalizacija.
        { broj = b; sled = s; }
    public Elemt (int b) { broj = b; }
}
  
```

// Listal.java - Klasa listi celih brojeva.

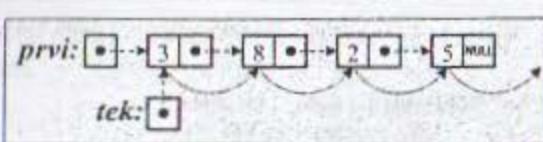
```

public class Listal {
    private Elemt prvi; // Početak liste.
    // Inicijalizacija:
    // - prazna lista,
    public Listal () {}
    public Listal (int b) // - lista od jednog broja.
        { prvi = new Elemt (b); }
    public void prazni () // Pražnjenje liste.
        { prvi = null; }
  
```

```

public int duz () { // Broj elemenata liste.
    int n = 0;
    for (Elem tek=prvi; tek!=null;
         tek=tek.sled)
        n++;
    return n;
}

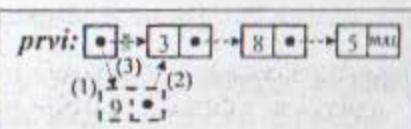
```



```

public String toString () { // Tekstualni oblik.
    String s = "";
    for (Elem tek=prvi; tek!=null; tek=tek.sled)
        s += tek.broj + " ";
    return s;
}

```

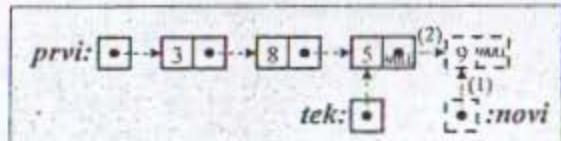


```

public void naPocetak (int b) { // Dodavanje
    prvi = new Elem (b, prvi); // na početak.
}

public void naKraj (int b) { // Dodavanje na kraj.
    Elek novi = new Elek (b);
    if (prvi == null) prvi = novi;
    else {
        Elek tek = prvi;
        while (tek.sled != null)
            tek = tek.sled;
        tek.sled = novi;
    }
}

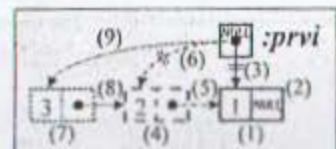
```



```

public void citaj1 (int n) { // Čitanje liste
    prvi = null; // stavljajući brojeve
    for (int i=0; i<n; i++) // na početak.
        prvi = new Elek (Citaj.Int (), prvi);
}

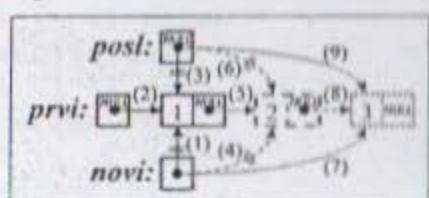
```



```

public void citaj2 (int n) { // Čitanje liste stavljajući
    Elek posl = prvi = null; // brojeve na kraj.
    for (int i=0; i<n; i++) {
        Elek novi = new Elek (Citaj.Int ());
        if (prvi == null) prvi = novi;
        else posl.sled = novi;
        posl = novi;
    }
}

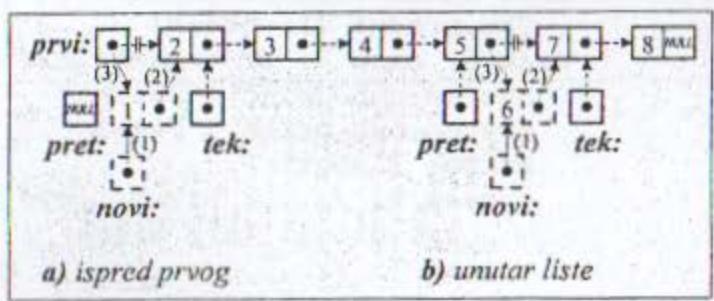
```



```

public void umetni (int b) { // Umetanje u uređenu listu.
    Elek tek = prvi, pret = null;
    while (tek != null &&
           tek.broj < b)
        ( pret = tek; tek = tek.sled; )
    Elek novi = new Elek (b, tek);
    if (pret == null) prvi = novi;
    else pret.sled = novi;
}

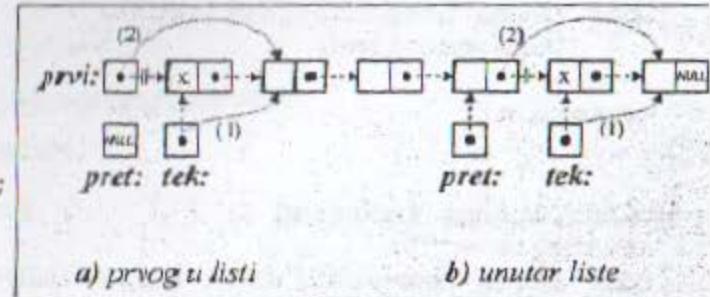
```



```

public void izostavi (int b) { // Izostavljanje svakog
    Elek tek = prvi, pret = null; // pojavljivanja broja.
    while (tek != null)
        if (tek.broj != b) {
            pret = tek; tek = tek.sled;
        } else {
            tek = tek.sled;
            if (pret == null) prvi = tek;
            else pret.sled = tek;
        }
    }
}

```



// ListalT.java - Ispitivanje klase listi celih brojeva.

```

public class ListalT {
    public static void main (String[] varg) {
        Listal lst = new Listal ();
        radi: while (true) {
            System.out.print (
                "\n1. Dodavanje broja na pocetak liste\n"
                "2. Dodavanje broja na kraj liste\n"
                "3. Umetanje broja u uredjenu listu\n"
                "4. Izostavljanje broja iz liste\n"
                "5. Brisanje svih elemenata liste\n"
                "6. Citanje uz obrtanje redosleda brojeva\n"
                "7. Citanje uz cuvanje redosleda brojeva\n"
                "8. Odredjivanje duzine liste\n"
                "9. Ispisivanje liste\n"
                "0. Zavrsetak rada\n\n"
                "Vas izbor? "
            );
            int izbor = Citaj.Int ();
            switch (izbor) {
                case 1: case 2: case 3: case 4:
                    System.out.print ("Broj?      "); int broj = Citaj.Int ();
                    switch (izbor) {
                        case 1: // Dodavanje broja na početak liste:
                            lst.naPocetak (broj); break;
                        case 2: // Dodavanje broja na kraj liste:
                            lst.naKraj (broj); break;
                        case 3: // Umetanje broja u uredenu listu:
                            lst.umetni (broj); break;
                        case 4: // Izostavljanje broja iz liste:
                            lst.izostavi (broj); break;
                    } break;
                case 5: // Pražnjenje liste:
                    lst.prazni (); break;
                case 6: case 7: // Čitanje liste ...
                    System.out.print ("Duzina?      "); int n = Citaj.Int ();
                    System.out.print ("Elementi?   ");
                    switch (izbor) {
                        case 6: // ... uz obrtanje redosleda brojeva:
                            lst.citaj1 (n); break;
                        case 7: // ... uz čuvanje redosleda brojeva:
                            lst.citaj2 (n); break;
                    } break;
            }
        }
    }
}

```

```
        case 8: // Određivanje dužine liste:  
            System.out.println ("Duzina=      " + lst.duz()); break;  
        case 9: // Ispisivanje liste:  
            System.out.println ("Lista=      " + lst); break;  
        case 0: // Završetak rada:  
            break radi;  
        default:// Pogrešan izbor:  
            System.out.println ("*** Nedozvoljen izbor! ***"); break;  
    }  
}
```

```

8 javac ListalT.java
8 java ListalT

1. Dodavanje broja na pocetak liste
2. Dodavanje broja na kraj liste
3. Umetanje broja u uredjenu listu
4. Izostavljanje broja iz liste
5. Brisanje svih elemenata liste
6. Citanje uz obrtanje redosleda brojeva
7. Citanje uz cuvanje redosleda brojeva
8. Odredjivanje duzine liste
9. Ispisivanje liste
0. Zavrsetak rada

Vas izbor? 1
Broj? 1
...
Vas izbor? 1
Broj? 2
...
Vas izbor? 1
Broj? 3
...
Vas izbor? 2
Broj? 4
...
Vas izbor? 2
Broj? 5
...
Vas izbor? 2
Broj? 6
...
Vas izbor? 8
Duzina= 6
...
Vas izbor? 9
Lista= 3 2 1 4 5 6
...
Vas izbor? 5
...
Vas izbor? 3
Broj? 5
...
Vas izbor? 3
Broj? 2
...
Vas izbor? 3
Broj? 4
...
Vas izbor? 3
Broj? 1
...
Vas izbor? 3
Broj? 5
...
Vas izbor? 3
Broj? 1
...
Vas izbor? 3
Broj? 5
...
Vas izbor? 3
Broj? 2
...
Vas izbor? 9
Lista= 1 2 2 4 5 5 6
...
Vas izbor? 4
Broj? 5
...
Vas izbor? 9
Lista= 1 2 3 4 5 6
...
Vas izbor? 6
Duzina? 5
Elementi? 1 2 3 4 5
...
Vas izbor? 9
Lista= 5 4 3 2 1
...
Vas izbor? 7
Duzina? 5
Elementi? 1 2 3 4 5
...
Vas izbor? 9
Lista= 1 2 3 4 5 6
...
Vas izbor? 11
*** Nedozvoljen
...
Vas izbor? 0

```

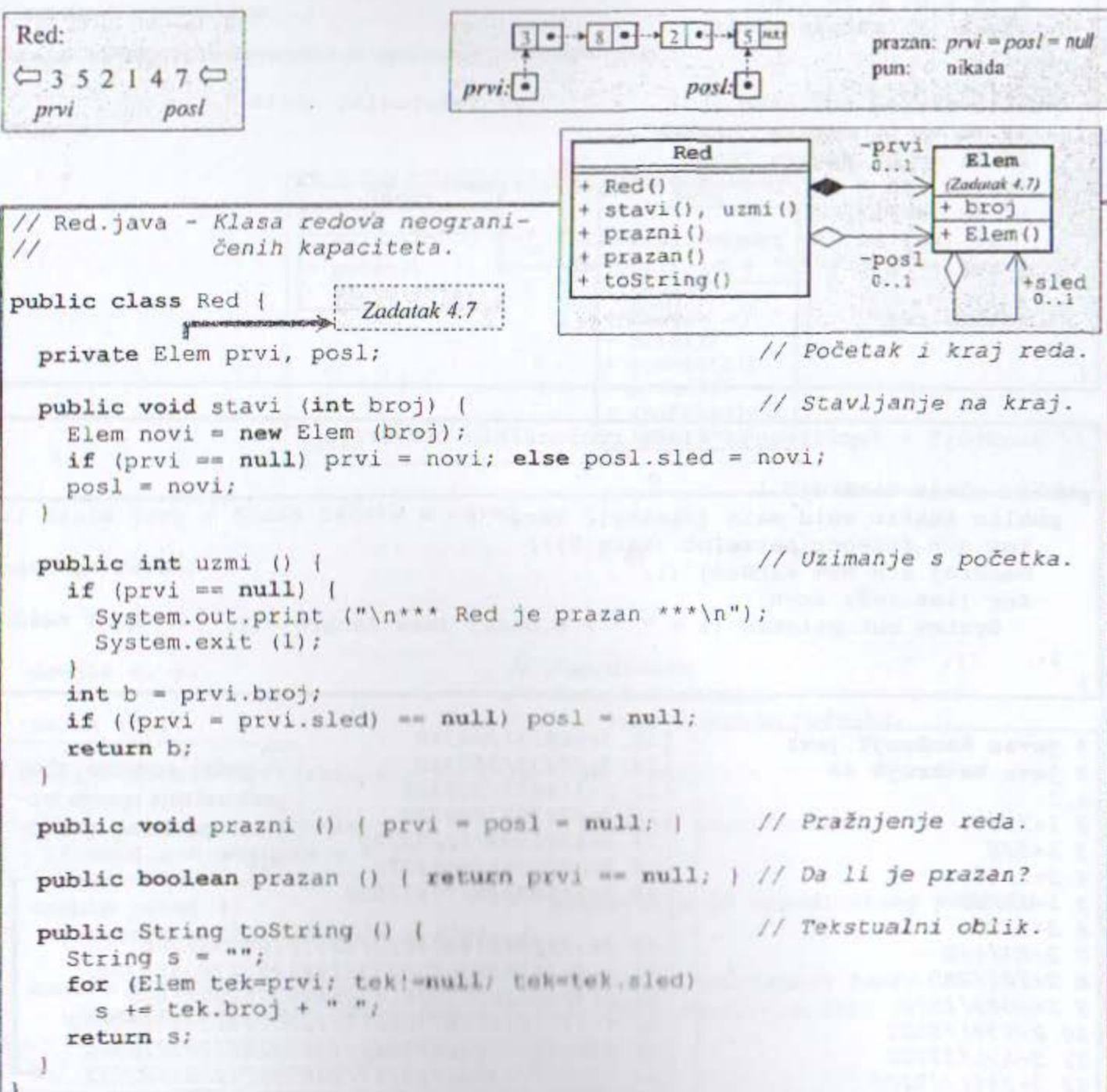
### Zadatak 4.8 Redovi brojeva neograničenih kapaciteta

Redovi su linearne strukture podataka kod kojih se podaci dodaju na jednom kraju i uzimaju na drugom kraju. Mogu imati ograničene ili neograničene kapacitete. Napisati na jeziku Java klasu redova celih brojeva neograničenih kapaciteta. Predviđeti:

- dodavanje jednog podatka i uzimanje jednog podatka,
- ispitivanje da li je red prazan,
- pražnjenje reda, i
- sastavljanje tekstualnog oblika reda.

Napisati na jeziku Java interaktivni program za ispitivanje prethodne klase.

Rešenje:



```

// RedT.java - Ispitivanje klase redova neograničenih kapaciteta.

public class RedT {
    public static void main (String[] varg) {
        Red r = new Red ();
        radi: while (true) {
            System.out.print (
                "\n1. Stavljanje podatka\n" +
                "2. Uzimanje podatka\n" +
                "3. Ispisivanje sadrzaja\n" +
                "4. Praznjenje reda\n" +
                "0. Zavrsetak rada\n\n" +
                "Vas izbor? "
            );
            switch (Citaj.Int ()) {
                case 1: System.out.print ("Broj? ");
                    r.stavi (Citaj.Int ());
                    break;
                case 2: if (! r.prazan ())
                    System.out.println ("Broj= " + r.uzmi ());
                    else System.out.println ("*** Red je prazan!");
                    break;
                case 3: System.out.println ("Red= " + r); break;
                case 4: r.prazni (); break;
                case 0: break radi;
                default: System.out.println ("*** Nedozvoljeni izbor!");
            }
        }
    }
}

```

```

$ javac RedT
$ java RedT

1. Stavljanje podatka
2. Uzimanje podatka
3. Ispisivanje sadrzaja
4. Praznjenje reda
0. Zavrsetak rada

Vas izbor? 1
Broj? 1
...
Vas izbor? 1
Broj? 2
...
Vas izbor? 1
Broj? 3
...

```

```

Vas izbor? 3
Red= 1 2 3
...
Vas izbor? 2
Broj= 1
...
Vas izbor? 2
Broj= 2
...
Vas izbor? 2
Broj= 3
...
Vas izbor? 2
*** Red je prazan!
...
Vas izbor? 7
*** Nedozvoljeni izbor!
...
Vas izbor? 0

```

### Zadatak 4.9 Racionalni brojevi

Napisati na jeziku Java klasu racionalnih brojeva koji se predstavljaju u obliku količnika dva cela broja. Predviđeti:

- stvaranje racionalnog broja,
- dodavanje i oduzimanje vrednosti drugog racionalnog broja,
- množenje i deljenje drugim racionalnim brojem, i
- sastavljanje tekstualnog oblika racionalnog broja.

Napisati na jeziku Java program za ispitivanje prethodne klase.

**Rešenje:**

```
// RacBroj.java - Klasa racionalnih brojeva.

public class RacBroj {
    // Zajednički niz prvih nekoliko prostih brojeva.
    static private final int maxProst = 10000;
    static private final long[] prosti = new long [maxProst];

    static {
        int n = 1;                                // Generisanje niza prostih brojeva.
        prosti[0] = 2;                            // (zajednički inicijalizator).
        for (int p=3; n<maxProst; p+=2) {
            boolean prost = true;
            for (int j=0; prosti[j]*prosti[j]<=p; j++)
                if (p % prosti[j] == 0) { prost = false; break; }
            if (prost) prosti[n++] = p;
        }
    }

    private long a, b;                          // Deljenik i delitelj.

    private void sredi () {                     // Sredivanje broja.
        if (a == 0) b = 1;
        else if (b != 0) {
            if (b < 0) { a = -a; b = -b; }
            for (int i=0; i < maxProst &&
                    prosti[i]*prosti[i]<=a &&
                    prosti[i]*prosti[i]<=b; i++) {
                while (a % prosti[i] == 0 && b % prosti[i] == 0) {
                    a /= prosti[i]; b /= prosti[i];
                }
            }
        }
    }

    public RacBroj () { b = 1; }                 // Inicijalizacija:
                                                // - nula (0/1),
    public RacBroj (long a) {                   // - ceo broj (a/1),
        this.a = a; b = 1; }

    public RacBroj (long a, long b) {           // - razlomljen broj (a/b),
        this.a = a; this.b = b;
        sredi ();
    }
}
```

**RacBroj**

- maxProst, prosti
- a, b
- + RacBroj()
- + dodaj(), oduzmi()
- + pomnozi(), podeli()
- + toString()

```

public RacBroj dodaj (RacBroj x) { // Dodavanje.
    a = a * x.b + x.a * b; b *= x.b;
    sredi (); return this;
}

public RacBroj oduzmi (RacBroj x) { // Oduzimanje.
    a = a * x.b - x.a * b; b *= x.b;
    sredi (); return this;
}

public RacBroj pomnozi (RacBroj x) { // Množenje.
    a *= x.a; b *= x.b;
    sredi (); return this;
}

public RacBroj podeli (RacBroj x) { // Deljenje.
    a *= x.b; b *= x.a;
    sredi (); return this;
}

public String toString () { // Tekstualni oblik.
    if (b == 0) return "GRESKA";
    if (a == 0) return "0";
    String rez = a / b != 0 ? Long.toString(a / b) : "";
    if (b != 1) {
        if (rez != "") rez += (a > 0) ? "+" : "-";
        rez += a % b + "/" + b;
    }
    return rez;
}
}

```

// RacBrojT - Ispitivanje klase racionalnih brojeva.

```

public class RacBrojT {
    public static void main (String[] varg) {
        int n = Integer.parseInt (varg[0]);
        RacBroj a = new RacBroj ();
        for (int i=1; i<=n;
            System.out.println (i + " " + a.dodaj (new RacBroj (1, i++)))
    }
}

```

```

8 javac RacBrojT.java
9 java RacBrojT 45
1 1
2 1+1/2
3 1+5/6
4 2+1/12
5 2+17/60
6 2+9/20
7 2+83/140
8 2+201/280
9 2+2089/2520
10 2+2341/2520
11 3+551/27720
12 3+2861/27720

```

```

13 3+64913/360360
14 3+90653/360360
15 3+114677/360360
16 3+274399/720720
17 3+5385503/12252240
18 3+2022061/4084080
19 3+42503239/77597520
...
40 4+135294214989013/485721041551200
41 4+6032783856100733/1991456270359920
42 4+929562872992619/2844937529085600
43 4+42816141067768217/1223323137506000
44 424165472261895961/4485518170858790
45 155607713594513747/24829824216468790

```

Pogrešni rezultati prekoračenja opsega jeva tipa long  
 $\pm 10^{21}$ )

**Zadatak 4.10 Tačke i krugovi koji ne smeju da se preklapaju u ravni**

Napisati na jeziku Java klasu tačaka u ravni. Predviđeti:

- stvaranje tačke zadatih koordinata (podrazumevano (0,0)), i
- izračunavanje rastojanja tačke od koordinatnog početka i od zadate tačke.

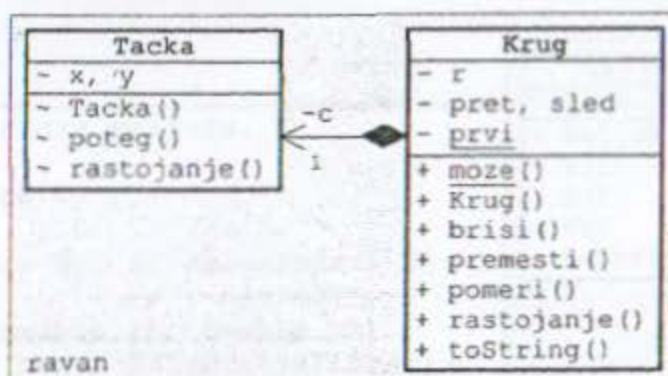
Napisati na jeziku Java klasu krugova u ravni koji ne smeju da se preklapaju. Predviđeti:

- proveru da li sme da se napravi krug datog poluprečnika i koordinata centra,
- stvaranje kruga zadatog poluprečnika i koordinata centra,
- uništavanje kruga,
- premeštanje centra kruga u određenu tačku i pomeranje kruga za zadati pomak,
- izračunavanje najmanjeg rastojanja do datog kruga, i
- sastavljanje tekstualnog oblika kruga.

Klase staviti u paket klasa.

Napisati na jeziku Java program za ispitivanje prethodnih klasa.

**Rešenje:**



```

// Tacka.java - Klasa tačaka u ravni.

package ravan;

class Tacka {

    double x, y; // Koordinate.
    // Inicijalizacija:
    // - u koordinatnom početku,
    Tacka () {}

    Tacka (double x) { this.x = x; } // - na x-osi,
    Tacka (double x, double y) // - zadatim koordinatama.
        ( this.x = x; this.y = y; )

    double poteg () // Rastojanje od koordinatnog početka.
        ( return Math.sqrt (x*x + y*y); )

    double rastojanje (Tacka t) // Rastojanje od zadate tačke.
        ( return Math.sqrt (Math.pow(x-t.x, 2)+Math.pow(y-t.y, 2)); )
}
  
```

```

// Krug.java - Klasa krugova u ravni koji ne smeju da se preklapaju.

package ravan;

public class Krug {

    private Tacka c; private double r; // Centar i poluprečnik.
    private Krug pret, sled;           // Prethodni i sledeći krug.
    private static Krug prvi = null;   // Zajednička lista svih krugova.

    private Krug () {}               // Samo za interne potrebe.
                                      // Može li krug da postoji?
    public static boolean moze (double r, double x, double y) {
        Krug k = new Krug (); k.r = r; k.c = new Tacka (x, y);
        for (Krug tek=prvi; tek!=null; tek=tek.sled)
            if (tek.rastojanje (k) < 0) return false;
        return true;
    }

    public Krug (double r, double x, double y) { // Stvaranje kruga.
        if (! moze (r, x, y)) System.exit (1);
        this.r = r;
        this.c = new Tacka (x, y);
        sled = prvi; pret = null;
        if (prvi != null) prvi.pret = this;
        prvi = this;
    }

    public void brisi () {             // Krug više nije potreban.
        if (pret != null) pret.sled = sled; else prvi = sled;
        if (sled != null) sled.pret = pret;
    }

    public boolean premesti (double x, double y) { // Premeštanje kruga.
        if (! moze (r, x, y)) return false;
        c.x = x; c.y = y; return true;
    }

    public boolean pomeri (double dx, double dy) { // Pomeranje kruga.
        if (! moze (r, c.x+dx, c.y+dy)) return false;
        c.x += dx; c.y += dy; return true;
    }

    double rastojanje (Krug k)          // Rastojanje do drugog kruga.
    { return c.rastojanje (k.c) - r - k.r; }

    public String toString ()           // Tekstualni oblik.
    { return "K[" + r + ", (" + c.x + "," + c.y + ")]"; }
}

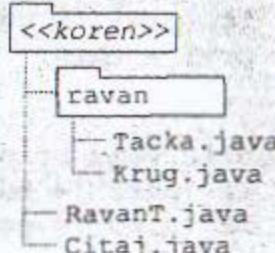
```

```
// RavanT.java - Ispitivanje klasa Krug i Tacka u paketu Ravan.

import ravan.Krug;

public class RavanT {
    public static void main (String[] varg) {
        Krug[] krugovi = new Krug [100];
        while (true) {
            int n = 0; System.out.println ();
            while (true) {
                System.out.print ("K[" + n + "] (r,x,y)? ");
                double r = Citaj.Double(), x = Citaj.Double(), y = Citaj.Double();
                if (r < 0) break;
                if (Krug.moze (r, x, y)) krugovi[n++] = new Krug (r, x, y);
                else System.out.println ("*** Ne moze da se smesti! ***");
            }
            if (n == 0) break;
            for (int i=0; i<n; System.out.println (krugovi[i++]));
            for (int i=0; i<n; krugovi[i++].brisi ());
        }
    }
}
```

Datoteke paketa treba stavljati u potkatalog (potfasciklu) čije je ime jednako imenu paketa. Može da se obrazuje hijerarhijska struktura paketa koja se stavlja u kataloge (fascikle) s istom strukturom.



```
% javac RavanT.java
% java RavanT

K[0] (r,x,y)? 1 1 1
K[1] (r,x,y)? 5 1 1
*** Ne moze da se smesti! ***
K[1] (r,x,y)? 1 5 5
K[2] (r,x,y)? 2 -1 1
*** Ne moze da se smesti! ***
K[2] (r,x,y)? 2 -2 2
K[3] (r,x,y)? -1 -1 -1
K[1.0, (1.0,1.0)]
K[1.0, (5.0,5.0)]
K[2.0, (-2.0,2.0)]

K[0] (r,x,y)? -1 -1 -1
```

## **5 Izvedene klase**

### Zadatak 5.1 Valjci i kante

*Valjak* se zadaje poluprečnikom i visinom. Napisati na jeziku Java klasu valjaka. Predviđeti:

- stvaranje valjka zadatog poluprečnika i visine,
- dohvatanje poluprečnika i visine,
- izračunavanje zapremine, i
- sastavljanje tekstualnog oblika valjka.

*Kanta* je valjak u koji može da se sipa tečnost. Napisati na jeziku Java klasu kanti kao izvedenu klasu iz klase valjaka. Pored mogućnosti osnovne klase predviđeti:

- stvaranje kante zadatog poluprečnika, visine i početne popunjenošći,
- dohvatanje količine tečnosti u kanti i određivanje koliko tečnosti može još da se dolije,
- ispitivanje da li je kanta skroz puna i da li je skroz prazna,
- dolivanje odredene količine tečnosti (ako se kanta prepuni višak tečnosti se prosipa),
- odlivanje odredene količine tečnosti,
- presipanje moguće najveće količine tečnosti iz jedne kante u drugu, i
- sastavljanje tekstualnog oblika kante.

Napisati na jeziku Java program za ispitivanje prethodnih klasa.

Rešenje:

```
// Valjak.java - Klasa valjaka.

public class Valjak {
    private double r, h; // Poluprečnik i
                        // visina.
    public Valjak (double rr, double hh)
        { r = rr; h = hh; } // Inicijalizacija.

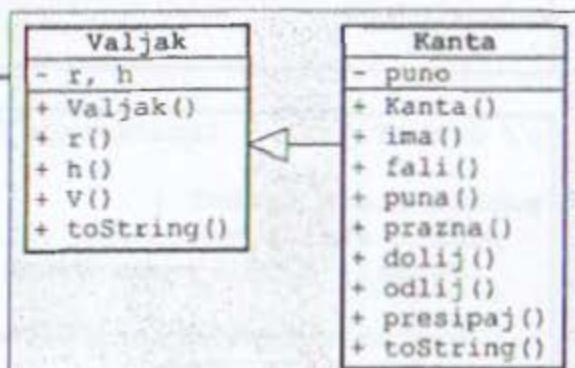
    public Valjak () { r = h = 1; }

    public double r () { return r; } // Poluprečnik.

    public double h () { return h; } // Visina.

    public double V () { return r * r * Math.PI * h; } // Zapremina.

    public String toString () // Tekstualni oblik.
        { return "[" + r + "," + h + "]"; }
}
```



```
// Kanta.java - Klasa kanti.

public class Kanta extends Valjak {
    private double puno; // Napunjeni deo.

    public Kanta (double rr, double hh, double pp) // Inicijalizacija
        { super (rr, hh); puno = pp <= V () ? pp : V (); }

    public Kanta (double rr, double hh) // (prazna kanta).
        { this (rr, hh, 0); }
```

```

public double ima () { return puno; }           // Koliko ima tečnosti?
public double fali () { return V() - puno; }     // Koliko tečnosti fali?
public boolean puna () { return puno == V(); }    // Da li je puna?
public boolean prazna () { return puno == 0; }   // Da li je prazna?
public Kanta dolij (double dopuna) {           // Dolivanje.
    puno = (puno+dopuna <= V()) ? puno+dopuna : V();
    return this;
}

public Kanta odlij (double odliv) {             // Odlivanje.
    if (puno-odliv > 0) puno -= odliv; else puno = 0;
    return this;
}

public Kanta presipaj (Kanta k) {               // Presipanje.
    if (this != k) {
        double prazno = V() - puno;
        if (prazno >= k.puno) { puno += k.puno; k.puno = 0; }
        else { puno = V(); k.puno -= prazno; }
    }
    return this;
}

public String toString ()                      // Tekstualni oblik.
{ return "(" + super.toString() + "," + puno + ")"; }
}

```

// ValjakT.java - Ispitivanje klase valjaka i kanti.

```

public class ValjakT {
    public static void main (String[] varg) {
        Valjak v1 = new Valjak (2, 3);
        System.out.println ("v1 : " + v1 + " " + v1.V());
        Kanta k1 = new Kanta (1, 3, 10);
        System.out.println ("k1(10) : " + k1 + " " + k1.V());
        System.out.println ("k1-=5 : " + k1.odlij(5));
        System.out.println ("k1+=4 : " + k1.dolij(4));
        Kanta k2 = new Kanta (0.6, 5);
        System.out.println ("k2(0) : " + k2 + " " + k2.V());
        System.out.println ("k2<-k1: " + k2.presipaj(k1));
        System.out.println ("k1 : " + k1 );
    }
}

```

```

$ javac ValjakT.java
$ java ValjakT
v1 : [2.0,3.0] 37.69911184307752
k1(10) : {[1.0,3.0],9.42477796076938} 9.42477796076938
k1-=5 : {[1.0,3.0],4.424777960769379}
k1+=4 : {[1.0,3.0],8.42477796076938}
k2(0) : {[0.6,5.0],0.0} 5.654866776461628
k2<-k1: {[0.6,5.0],5.654866776461628}
k1 : {[1.0,3.0],2.7699111843077517}

```

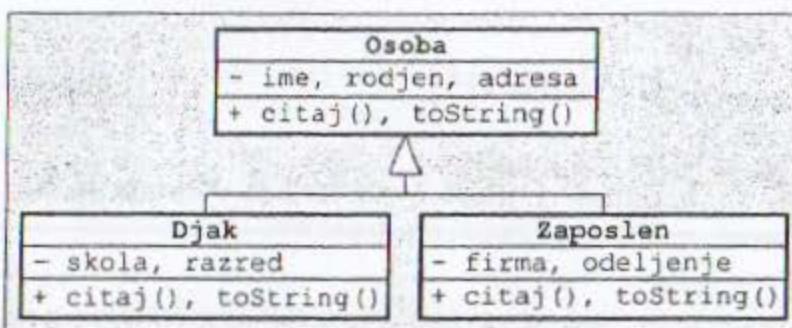
### Zadatak 5.2 Osobe, daci i zaposleni

Podatke o *osobi* čine ime, datum rođenja i adresa stanovanja. *Dak* je osoba za koju se dodatno znaju naziv škole i razred koji pohađa. *Zaposlen* je osoba za koju se dodatno zna naziv firme i naziv odeljenja u kome radi.

Napisati na jeziku Java klase koje omogućavaju unificiranu obradu podataka o nabrojanim kategorijama osoba. Svi atributi su niske znakova. Predviđeti unošenje podataka u objekte čitanjem s glavnog ulaza i sastavljanje tekstualnog oblika objekata.

Napisati na jeziku Java program za ispitivanje prethodnih klasa.

**Rešenje:**



```

// Osoba.java - Klasa osoba.

public class Osoba {

    private String ime, rodjen, adresa;

    public void citaj () {
        System.out.print ("Ime i prezime? "); Citaj.getNL (); ime = Citaj.Line ();
        System.out.print ("Datum rodjenja? "); rodjen = Citaj.Line ();
        System.out.print ("Adresa stanovanja? "); adresa = Citaj.Line ();
    }

    public String toString () {
        return "Ime i prezime: " + ime + "\n" +
            "Datum rodjenja: " + rodjen + "\n" +
            "Adresa stanovanja: " + adresa + "\n";
    }
}
  
```

```

// Djak.java - Klasa daka.

public class Djak extends Osoba {

    private String skola, razred;

    public void citaj () {
        super.citaj ();
        System.out.print ("Naziv skole? "); skola = Citaj.Line ();
        System.out.print ("Razred? "); razred = Citaj.Line ();
    }
}
  
```

```

public String toString () {
    return super.toString () +
        "Naziv skole:      " + skola      + "\n" +
        "Razred:          " + razred      + "\n" ;
}

// Zaposlen.java - Klasa zaposlenih.

public class Zaposlen extends Osoba {

    protected String firma, odeljenje;

    public void citaj () {
        super.citaj ();
        System.out.print ("Naziv firme?      "); firma = Citaj.readLine();
        System.out.print ("Naziv odeljenja?   "); odeljenje = Citaj.readLine();
    }

    public String toString () {
        return super.toString () +
            "Naziv firme:      " + firma      + "\n" +
            "Naziv odeljenja:  " + odeljenje + "\n" ;
    }
}

// OsobeT.java - Ispitivanje klasa osoba, daka i zaposlenih.

public class OsobeT {
    public static void main (String[] varg) {
        Osoba[] ljudi = new Osoba [20]; int n = 0;

        System.out.println ("Citanje podataka o ljudima");
        radi: while (true) {
            System.out.print ("\nIzbor (O=osoba, D=djak, Z=zaposlen, K=Kraj): ");
            switch (Citaj.Char ()) {
                case 'O': case 'o': ljudi[n] = new Osoba (); break;
                case 'D': case 'd': ljudi[n] = new Djak (); break;
                case 'Z': case 'z': ljudi[n] = new Zaposlen (); break;
                case 'K': case 'k': break radi;
            }
            if (ljudi[n] != null) ljudi[n++].citaj ();
        }

        System.out.println ("\nPrikaz procitanih podataka");
        for (int i=0; i<n; i++) System.out.print ("\n" + ljudi[i]);
    }
}

```

```
% javac OsobeT.java
```

```
% java OsobeT
```

Citanje podataka o ljudima

Izbor (O=osoba, D=djak, Z=zaposlen, K=kraj)? o

Ime i prezime? Marko Markovic

Datum rodjenja? 13. 5. 1984.

Adresa stanovanja? Avalska 33, Beograd

Izbor (O=osoba, D=djak, Z=zaposlen, K=kraj)? d

Ime i prezime? Zoran Zoranovic

Datum rodjenja? 1. 1. 1985.

Adresa stanovanja? Backa 132, Beograd

Naziv skole? O.S. "P. P. Njegos"

Razred? III-1

Izbor (O=osoba, D=djak, Z=zaposlen, K=kraj)? z

Ime i prezime? Petar Petrovic

Datum rodjenja? 23. 10. 1965.

Adresa stanovanja? Pancevacki put 37a, Beograd

Naziv firme? "IMT Rakovica", Beograd

Naziv odeljenja? Nabavna sluzba

Izbor (O=osoba, D=djak, Z=zaposlen, K=kraj)? k

Prikaz procitanih podataka

Ime i prezime: Marko Markovic

Datum rodjenja: 13. 5. 1984.

Adresa stanovanja: Avalska 33, Beograd

Ime i prezime: Zoran Zoranovic

Datum rodjenja: 1. 1. 1985.

Adresa stanovanja: Backa 132, Beograd

Naziv skole: O.S. "P. P. Njegos"

Razred: III-1

Ime i prezime: Petar Petrovic

Datum rodjenja: 23. 10. 1965.

Adresa stanovanja: Pancevacki put 37a, Beograd

Naziv firme: "IMT Rakovica", Beograd

Naziv odeljenja: Nabavna sluzba

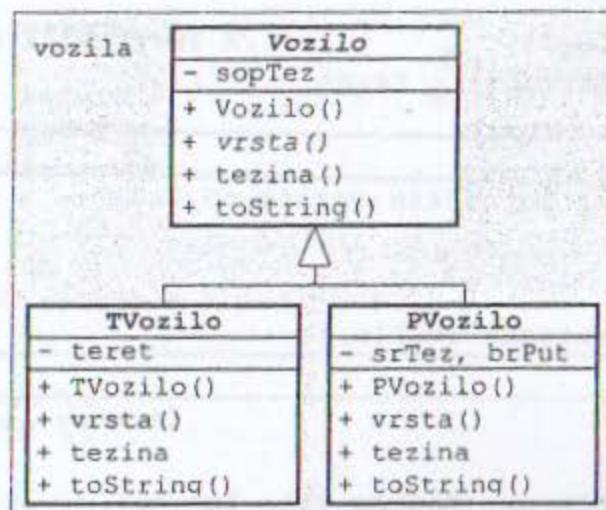
### Zadatak 5.3 Vozila, teretna vozila i putnička vozila

*Vozilo* ima sopstvenu težinu. *Teretno vozilo* je vozilo koje je natovareno teretom određene težine. *Putničko vozilo* je vozilo u kome se nalazi izvestan broj putnika zadate prosečne težine.

Napisati na jeziku Java klase koje omogućavaju unificiranu obradu nabrojanih vrsta vozila. Predvideti inicijalizaciju zadatim vrednostima parametara, dohvatanje jednoslovne oznake vrste vozila, izračunavanje ukupne težine vozila i sastavljanje tekstualnog oblika vozila.

Napisati na jeziku Java program koji s glavnog ulaza pročita podatke o određenom broju vozila i posle toga na glavnom izlazu ispiše podatke o vozilima koja mogu da pređu preko mosta zadate nosivosti.

**Rešenje:**



```

// Vozilo.java - Apstraktna klasa vozila.

package vozila;

public abstract class Vozilo {

    private double sopTez; // Sopstvena težina.

    public Vozilo (double st) { sopTez = st; } // Inicijalizacija.

    public abstract char vrsta (); // Vrsta vozila.

    public double tezina () { return sopTez; } // Ukupna težina.

    public String toString () // Tekstualni oblik.
        { return vrsta() + "(" + sopTez + ","; }
}
  
```

```
// TVozilo.java - Klasa teretnih vozila.

package vozila;

public class TVozilo extends Vozilo {

    private double teret;           // Težina tereta.

    public TVozilo (double st, double t) // Inicijalizacija.
        { super (st); teret = t; }

    public char vrsta () { return 'T'; } // Vrsta vozila.

    public double tezina ()          // Ukupna težina.
        { return super.tezina () + teret; }

    public String toString ()         // Tekstualni oblik.
        { return super.toString () + teret + ")"; }
}
```

```
// PVOzilo.java - Klasa putničkih vozila.

package vozila;

public class PVOzilo extends Vozilo {

    double srTez;                 // Srednja težina putnika.
    int brPut;                     // Broj putnika.

    public PVOzilo (double st, double srt, int bp) // Inicijalizacija.
        { super (st); srTez = srt; brPut = bp; }

    public char vrsta () { return 'P'; }           // Vrsta vozila.

    public double tezina ()                      // Ukupna težina.
        { return super.tezina () + srTez * brPut; }

    public String toString ()                    // Tekstualni oblik.
        { return super.toString () + srTez + "," + brPut + ")"; }
}
```

```
// VozilaT.java - Ispitivanje klase vozila.

import vozila.*;

public class VozilaT {
    public static void main (String[] varg) {
        Vozilo[] vozila = new Vozilo [100]; int n = 0;
        radi: while (true) {
            System.out.print ("\nVrsta vozila (T,P,*)? ");

```

```

switch (Citaj.Char ()) {
    case 't': case 'T':
        System.out.print ("Sopstvena tezina?      ");
        double sTez = Citaj.Double ();
        System.out.print ("Tezina tereta?      ");
        double ter = Citaj.Int ();
        vozila[n++] = new TVozilo (sTez, ter);
        break;
    case 'p': case 'P':
        System.out.print ("Sopstvena tezina?      ");
        sTez = Citaj.Double ();
        System.out.print ("Sr. tezina putnika?   ");
        double srTez = Citaj.Double ();
        System.out.print ("Broj putnika?       ");
        int brPut = Citaj.Int ();
        vozila[n++] = new PVozilo (sTez, srTez, brPut);
        break;
    case '*': break radi;
    default:
        System.out.println ("*** Nepoznata vrsta vozila!");
}
}
System.out.print ("\nNosivost mosta?      ");
double nosivost = Citaj.Double ();
System.out.print ("\nMogu da predju most:\n");
for (int i=0; i<n; i++)
    if (vozila[i].tezina() <= nosivost)
        System.out.println (vozila[i] + " - " + vozila[i].tezina());
}
}

```

```

% javac VozilaT
% java VozilaT

Vrsta vozila (T,P,*)? t
Sopstvena tezina?      1500
Tezina tereta?      5000

Vrsta vozila (T,P,*)? p
Sopstvena tezina?      1000
Sr. tezina putnika?    75
Broj putnika?      5

Vrsta vozila (T,P,*)? t
Sopstvena tezina?      2000
Tezina tereta?      10000

Vrsta vozila (T,P,*)? *
Nosivost mosta?      8000

Mogu da predju most:
T(1500.0,5000.0) - 6500.0
P(1000.0,75.0,5) - 1375.0

```

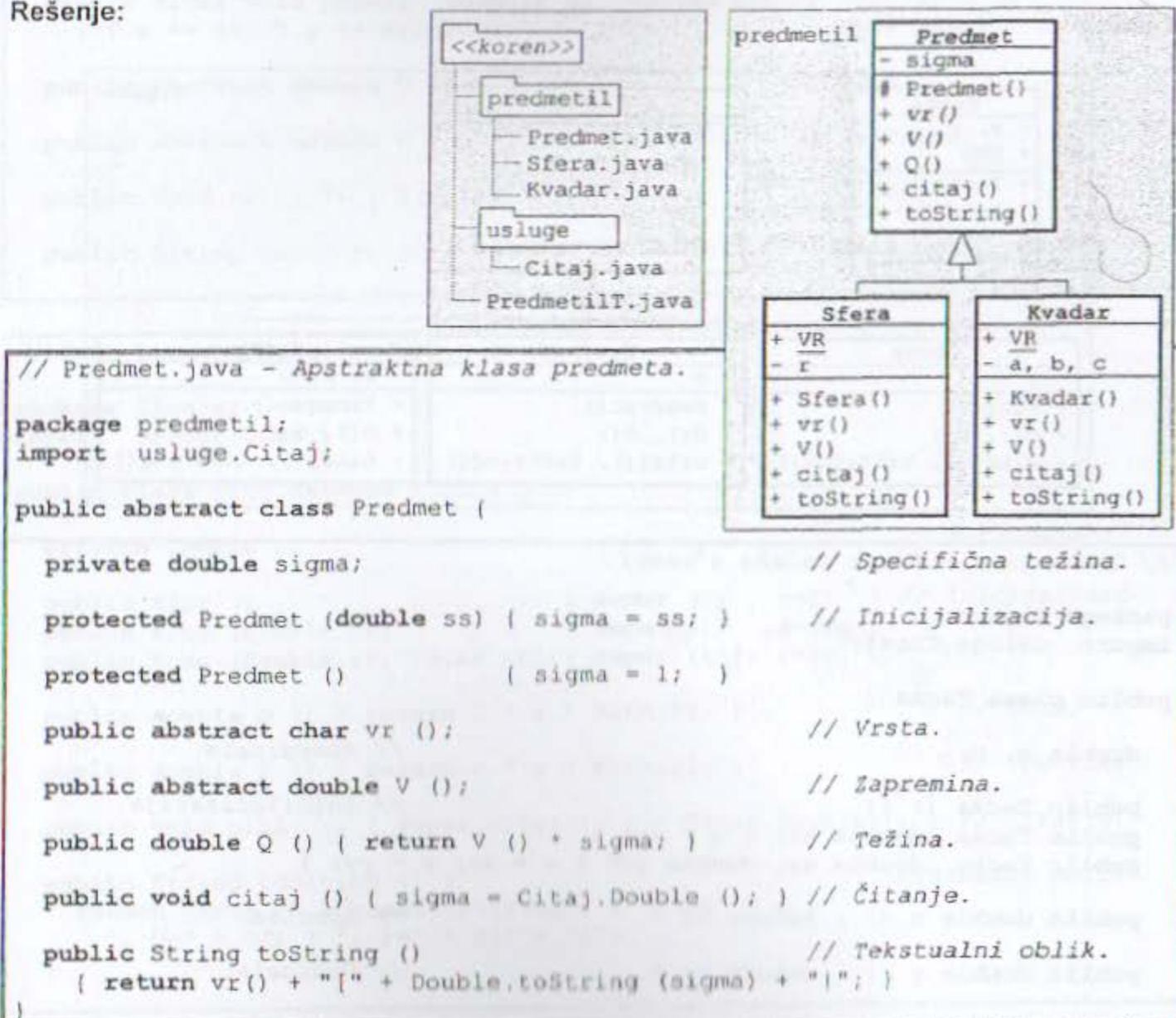
### Zadatak 5.4 Predmeti, sfere i kvadri

*Predmet* ima specifičnu težinu i jednoslovnu oznaku vrste predmeta. *Sfera* je predmet zadat poluprečnikom. *Kvadar* je predmet zadat dužinama ivica.

Napisati na jeziku Java klase koje omogućavaju unificiranu obradu naboranih vrsta predmeta. Predviđeti inicijalizaciju zadatim vrednostima parametara (podrazumevano 1), dohvatanje oznake vrste predmeta, izračunavanje zapremine, izračunavanje težine, čitanje predmeta s glavnog ulaza i sastavljanje tekstu-alnog oblika predmeta.

Napisati na jeziku Java program koji s glavnog ulaza pročita podatke o određenom broju predmeta i posle toga na glavnom izlazu ispiše podatke o predmetima čije su težine iznad prosečne.

Rešenje:



**Napomena:** U novijim verzijama jezika Java ne mogu da se uvezu klase iz bezimenog paketa (na primer: `import Citaj;` – što je ranije bilo moguće). Zbog toga postoje dva primerka klase `Citaj` – jedan u bezimenom paketu i jedan u paketu `usluge`. U ranijim zadacima, korišćen je primerak iz bezimenog paketa.

```
// Sfera.java - Klasa sfera.

package predmeti;
import usluge.Citaj;

public class Sfera extends Predmet {
    public static final char VR = 'S'; // Vrsta.

    private double r; // Poluprečnik.

    public Sfera (double ss, double rr) // Inicijalizacija.
        { super (ss); r = rr; }

    public Sfera () { this (1, 1); }

    public char vr () { return VR; } // Vrsta.

    public double V() { return 4./3 * r*r*r * Math.PI; } // Zajedno.

    public void citaj () // Čitanje.
        { super.citaj (); r = Citaj.Double (); }

    public String toString () // Tekstualni oblik.
        { return super.toString () + r + "]"; }
}
```

```
// Kvadar.java - Klasa kvadara.

package predmeti;
import usluge.Citaj;

public class Kvadar extends Predmet {
    public static final char VR = 'K'; // Vrsta.

    private double a, b, c; // Ivice.

    public Kvadar (double ss, double aa, double bb, double cc) // Inicijali-
        { super (ss); a = aa; b = bb; c = cc; } // zacija.

    public Kvadar () { this (1, 1, 1, 1); }

    public char vr () { return VR; } // Vrsta.

    public double V () { return a * b * c; } // Zajedno.

    public void citaj () // Čitanje.
        { super.citaj ();
          a = Citaj.Double (); b = Citaj.Double (); c = Citaj.Double ();
        }

    public String toString () // Tekstualni oblik.
        { return super.toString () + a + "," + b + "," + c + "]"; }
}
```

```
// PredmetiT.java - Ispitivanje klase predmeta.

import predmeti.*;
import usluge.Citaj;

public class PredmetiT {
    public static void main (String[] vpar) {
        Predmet[] p = new Predmet [100];
        double q = 0; int n = 0;
        radi: while (true) {
            switch (Citaj.Char ()) {
                case 's': case 'S': p[n] = new Sfera (); break;
                case 'k': case 'K': p[n] = new Kvadar (); break;
                case '.': break radi;
            }
            if (p[n] != null) {
                p[n].citaj ();
                System.out.println (p[n] + " (Q=" + p[n].Q () + ")");
                q += p[n++].Q ();
            }
        }
        if (n != 0) q /= n;
        System.out.print ("\nQsr= " + q + "\n\n");
        for (int i=0; i<n; i++)
            if (p[i].Q () > q)
                System.out.println (p[i] + " (Q=" + p[i].Q () + ")");
    }
}
```

**PredmetiT.pod**

```
k 0.5 1 2 3
s 2.8 1
s 1.8 0.75
k 1.5 2 2 2
.
```

```
% javac PredmetiT.java
% java PredmetiT <PredmetiT.pod
K{0.5|1.0,2.0,3.0} (Q=3.0)
S{2.8|1.0} (Q=11.728612573401893)
S{1.8|0.75} (Q=3.1808625617596658)
K{1.5|2.0,2.0,2.0} (Q=12.0)

Qsr= 7.477368783790389

S{2.8|1.0} (Q=11.728612573401893)
K{1.5|2.0,2.0,2.0} (Q=12.0)
```

### Zadatak 5.5 Geometrijske figure, krugovi, kvadrati i trouglovi u ravni

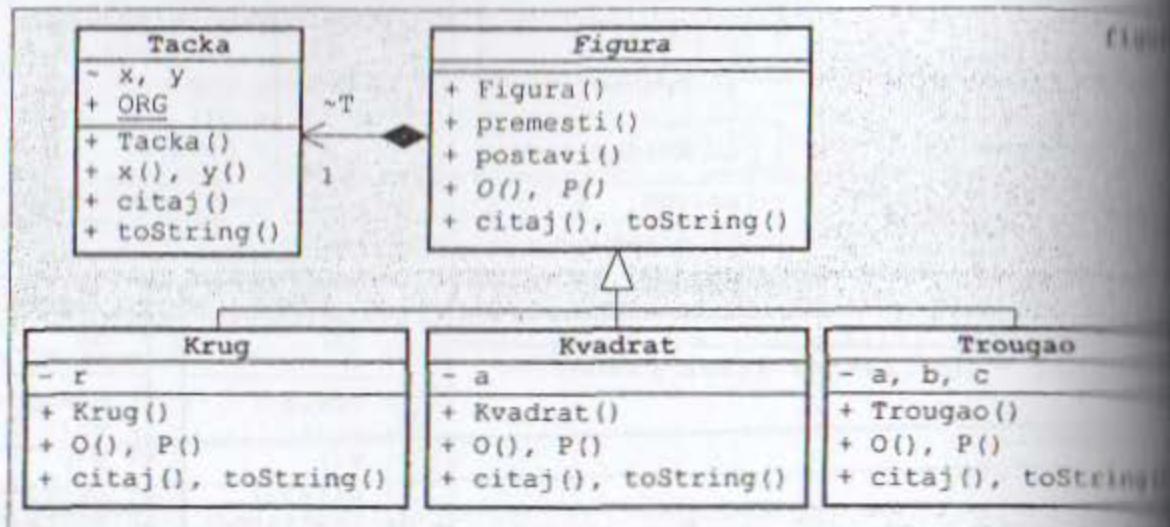
Napisati na jeziku Java klasu geometrijskih figura u ravni. Predvideti:

- stvaranje figure sa težištem u zadatoj tački,
- premeštanje figure u novu tačku i pomeranje figure za određeni pomak,
- izračunavanje obima i površine figure,
- čitanje figure s glavnog ulaza, i
- sastavljanje tekstualnog oblika figure.

Napisati na jeziku Java klase krugova, kvadrata i trouglova u ravni kao proširenja klase figura.

Napisati na jeziku Java program za ispitivanje prethodnih klasa.

**Rešenje:**



```
// Tacka.java - Klasa tacaka u ravni.

package figure;
import usluge.Citaj;

public class Tacka {

    double x, y; // Koordinate.

    public Tacka () {} // Inicijalizacija
    public Tacka (double xx) { x = xx; }
    public Tacka (double xx, double yy) { x = xx; y = yy; }

    public double x () { return x; } // Apscisa.

    public double y () { return y; } // Ordinata.

    public void citaj ()
        { x = Citaj.Double (); y = Citaj.Double (); } // Citanje.

    public String toString () // Tekstualni oblik
        { return "(" + x + "," + y + ")"; }

    public static final Tacka ORG = new Tacka (); // Koordinatni položaj
}
```

```
// Figura.java - Apstraktna klasa figure u ravni.

package figure;

public abstract class Figura {
    protected Tacka T;                                // Težiste figure.

    public Figura () { T = new Tacka (); }           // Inicijalizacija.
    public Figura (Tacka TT) { T = new Tacka (TT.x, TT.y); }

    public final void postavi (double xx, double yy) // Postavljanje u
    { T.x = xx; T.y = yy; }                         // novu tačku.

    public final void pomeri (double dx, double dy) // Pomeranje za
    { T.x += dx; T.y += dy; }                        // dati pomak.

    public abstract double O ();                      // Obim.

    public abstract double P ();                      // Površina.

    public void citaj () { T.citaj (); }             // Čitanje.

    public String toString () { return "T=" + T; }    // Tekstualni oblik.
}
```

```
// Krug.java - Klasa krugova u ravni.

package figure;
import usluge.Citaj;

public class Krug extends Figura {
    private double r;                                // Poluprečnik.

    public Krug () { super (); r=1; }                // Inicijalizac.
    public Krug (double rr) { super (); r=rr; }
    public Krug (double rr, Tacka tt) { super (tt); r=rr; }

    public double O () { return 2 * r * Math.PI; }    // Obim.

    public double P () { return r * r * Math.PI; }    // Površina.

    public void citaj () { super.citaj(); r = Citaj.Double(); } // Čitanje.

    public String toString () {                       // Tekstualni oblik.
        return "krug [" + super.toString() + ", r=" + r +
               ", O=" + O() + ", P=" + P() + "]";
    }
}
```

```
// Kvadrat.java - Klasa kvadrata u ravni.

package figure;
import usluge.Citaj;

public class Kvadrat extends Figura {
    private double a;                                // Stranica.

    public Kvadrat () { super (); a=1; } // Inicijalizacija.
    public Kvadrat (double aa) { super (); a=aa; } // zacijaja.
    public Kvadrat (double aa, Tacka tt) { super (tt); a=aa; }

    public double O () { return 4 * a; }                // Obim.

    public double P () { return a * a; }                 // Površina.

    public void citaj () { super.citaj (); a = Citaj.Double (); } // Čitanje.

    public String toString () {                         // Tekstualni oblik.
        return "kvadr[" + super.toString () + ", a=" + a +
               ", O=" + O () + ", P=" + P () + "]";
    }
}
```

```
// Trougao.java - Klasa trouglova u ravni.

package figure;
import usluge.Citaj;

public class Trougao extends Figura {
    private double a, b, c;                            // Stranice.

    public Trougao () { super (); a=b=c=1; } // Inicijalizacija.
    public Trougao (double aa) { super (); a=b=c=aa; }
    public Trougao (double aa, double bb)
        { super (); a = aa; b = c = bb; }
    public Trougao (double aa, double bb, double cc)
        { super (); a = aa; b = bb; c = cc; }
    public Trougao (double aa, double bb, double cc, Tacka tt)
        { super (tt); a = aa; b = bb; c = cc; }

    public double O () { return a + b + c; }            // Obim.

    public double P () {                               // Površina.
        double s = (a + b + c) / 2;
        return Math.sqrt (s * (s-a) * (s-b) * (s-c));
    }

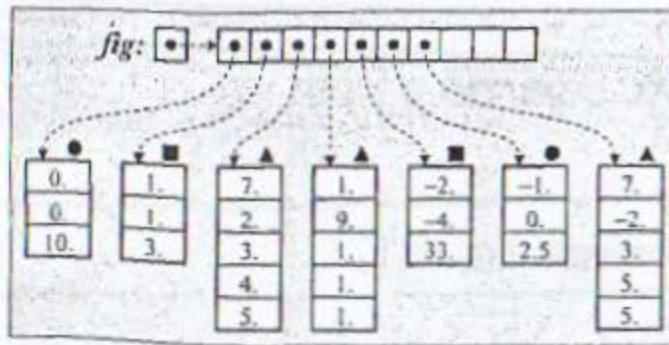
    public void citaj () { super.citaj ();             // Čitanje.
        a = Citaj.Double (); b = Citaj.Double (); c = Citaj.Double ();
    }

    public String toString () {                        // Tekstualni oblik.
        return "troug[" + super.toString () + ", a=" + a + ", b=" + b +
               ", c=" + c + ", O=" + O () + ", P=" + P () + "]";
    }
}
```

```
// FigureT.java - Ispitivanje klasa geometrijskih figura.

import figure.*;
import usluge.Citaj;

public class FigureT {
    public static void main (String[] varg) {
        Figura[] fig = new Figura [100];
        int n = 0;
        radi: while (true) {
            System.out.print ("Figura? ");
            switch (Citaj.Char ()) {
                case 'o': case 'O': fig[n] = new Krug (); break;
                case 'k': case 'K': fig[n] = new Kvadrat (); break;
                case 't': case 'T': fig[n] = new Trougao (); break;
                default : break radi;
            }
            fig[n++].citaj ();
        }
        for (int i=0; i<n; System.out.println (fig[i++]));
    }
}
```



```
* java FigureT
Figura? o 0 0 10
Figura? k 1 1 3
Figura? t 7 2 3 4 5
Figura? k -2 -4 33
Figura? o -1 0 2.5
Figura? t 7 -2 3 5 5
Figura? x
krug [T=(0.0,0.0), r=10.0, O=62.83185307179586, P=314.1592653589793]
kvadrat[T=(1.0,1.0), a=3.0, O=12.0, P=9.0]
troug[T=(7.0,2.0), a=3.0, b=4.0, c=5.0, O=12.0, P=6.0]
kvadrat[T=(-2.0,-4.0), a=33.0, O=132.0, P=1089.0]
krug [T=(-1.0,0.0), r=2.5, O=15.707963267948966, P=19.634954084936208]
troug[T=(7.0,-2.0), a=3.0, b=5.0, c=5.0, O=13.0, P=7.1545440106270926]
```

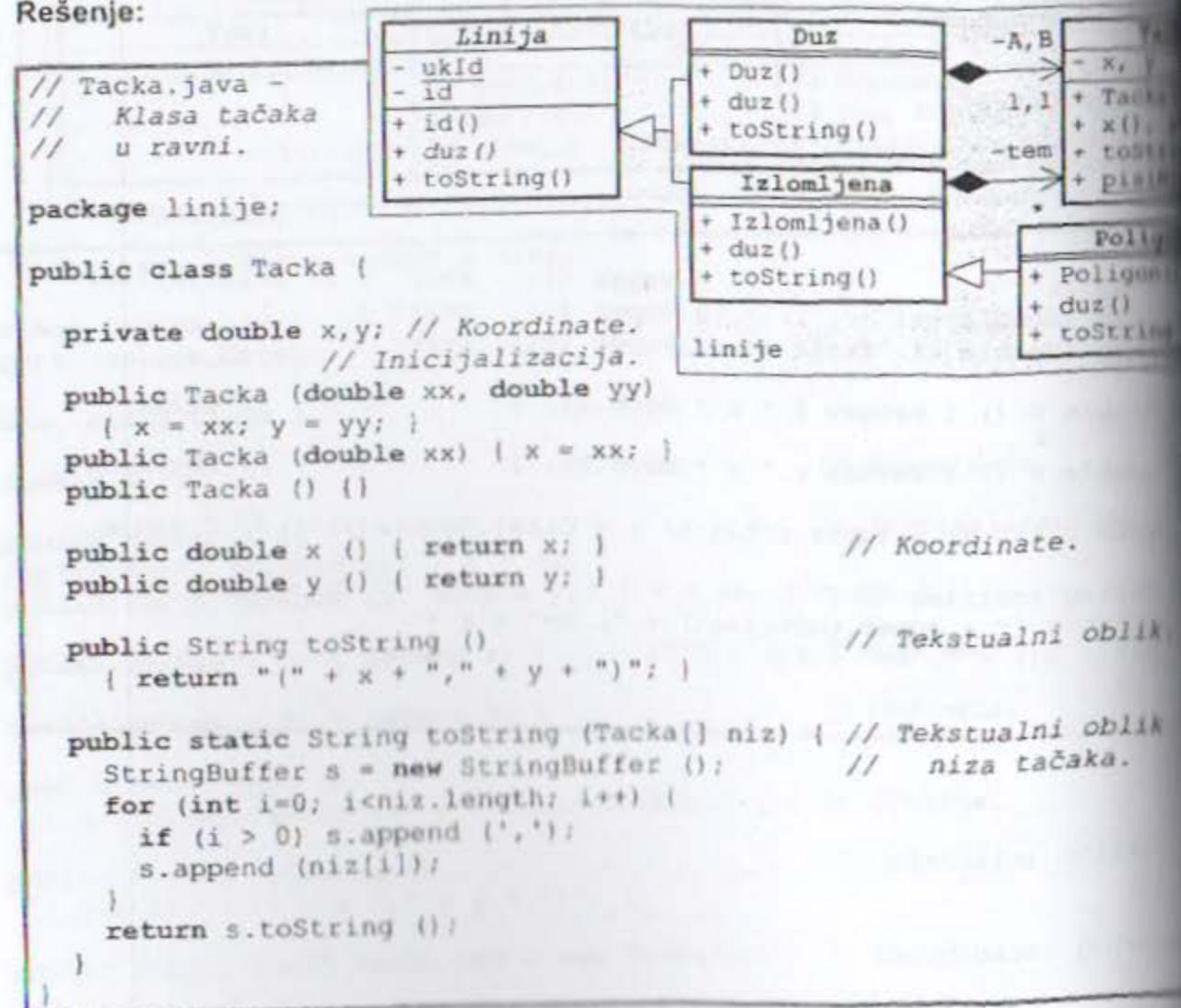
### Zadatak 5.6 Tačake, linije, duži, izlomljene linije i poligoni u ravni

Napisati na jeziku Java sledeće klase:

- **Tačka** u ravni se zadaje pomoću realnih koordinata (podrazumevano (0,0)), mogu da su vrednosti koordinata. Tekstualni oblik tačke je  $(x, y)$ , gde su:  $x$  i  $y$  – koordinate tačke. Postoji i uslužna metoda za sastavljanje tekstualnog oblika niza tačaka  $T, T_1, \dots, T_n$ . Tekstualni oblik jedne tačke u nizu.
- Apstraktna **linija** u ravni ima jedinstven, automatski generisan identifikacioni broj, dohvati identifikacioni broj, izračuna dužinu i sastavi tekstualni oblik koji sadrži identifikacioni broj linije.
- **Duž** je linija koja je zadata krajnjim tačkama (podrazumevano  $(-1, -1)$  i  $(1, 1)$ ). Tekstualni oblik, pored identifikacionog broja, sadrži i reč **duž** i koordinate krajnjih tačaka.
- **Izlomljena** linija je linija koja se sastoji od niza tačaka koje predstavljaju temena linije. Inicijalizuje nizom tačaka koje čine temena linije. Tekstualni oblik, pored identifikacionog broja, sadrži i reč **izlomljena** i niz koordinata temena.
- **Poligon** je zatvorena izlomljena linija koja se dobija spajanjem prvog i poslednjeg temena. Tekstualnom obliku koristi se reč **poligon**.

Napisati na jeziku Java klasu s funkcijom za čitanje jedne linije s glavnog ulaza i glavnu funkciju koja s glavnog ulaza pročitu niz raznovrsnih linija, ispiše sve pročitane podatke na glavnom izlazu (učujući i dužine pojedinih linija), pronađe najdužu liniju i ispiše je na glavnom izlazu. Dužina linije se zadaje se kao parametar glavne funkcije.

Rešenje:



```
// Linija.java - Apstraktna klasa linija u ravni.

package linije;

public abstract class Linija {

    private static int ukId = 0;           // Poslednje korisćeni identifikator.
    private int id = ++ukId;               // Identifikator linije.

    public int id() { return id; }         // Dohvatanje identifikatora linije.

    public abstract double duz();          // Dužina linije.

    public String toString()              // Tekstualni oblik.
        { return Integer.toString(id); }
}
```

```
// Duz.java - Klasa duži u ravni.

package linije;

public class Duz extends Linija {
    private Tacka A, B;                  // Krajnje tačke.

    public Duz (Tacka P, Tacka Q) { A = P; B = Q; } // Inicijalizacija.
    public Duz () { A = new Tacka (-1,-1); B = new Tacka (1,1); }

    public double duz () {                // Dužina duži.
        return Math.sqrt (Math.pow(A.x()-B.x(),2) + Math.pow(A.y()-B.y(),2));
    }

    public String toString()             // Tekstualni oblik.
        { return super.toString() + "[duz: " + A + ", " + B + "]"; }
}
```

```
// Izlomljena.java - Klasa izlomljenih linija u ravni.

package linije;

public class Izlomljena extends Linija {

    protected Tacka[] tem;               // Niz temena.

    public Izlomljena (Tacka[] niz)      // Inicijalizacija nizom tačaka.
        { tem = niz; }

    public double duz () {                // Dužina izlomljene linije.
        double d = 0;
        for (int i=0; i<tem.length-1; i++)
            d += new Duz(tem[i], tem[i+1]).duz();
        return d;
    }

    public String toString() {           // Tekstualni oblik.
        return super.toString() + "[izlomljena: " +
            Tacka.toString (tem) + "]";
    }
}
```

```
// Poligon.java - Klasa poligona u ravni.

package linije;

public class Poligon extends Izlomljena {
    public Poligon (Tacka[] niz)           // Inicijalizacija nizom tačaka.
    { super (niz); }

    public double duz ()                  // Dužina poligona.
    { return super.duz () + new Duz(tem[0], tem[tem.length-1]).duz (); }

    public String toString ()            // Tekstualni oblik.
    { return id() + "[poligon: " + Tacka.toString (tem) + "]"; }
}
```

```
// LinijeT.java - Ispitivanje klase linija u ravni.

import linije.*;

public class LinijeT {

    private static Linija citaj() {          // Čitanje jedne linije.
        System.out.print ("Vrsta (D, I, P)? ");
        char vrs = Citaj.Char ();
        switch (vrs) {
            case 'd': case 'D':
                System.out.print("A? ");
                double xA = Citaj.Double (), yA = Citaj.Double ();
                System.out.print("B? ");
                double xB = Citaj.Double (), yB = Citaj.Double ();
                return new Duz (new Tacka (xA, yA), new Tacka (xB, yB));
            }
            case 'i': case 'I':
            case 'p': case 'P':
                System.out.print("Broj temena? ");
                Tacka[] niz = new Tacka [Citaj.Int()];
                for (int i=0; i<niz.length; i++) {
                    System.out.print(i+1 + ". teme? ");
                    double x = Citaj.Double (), y = Citaj.Double ();
                    niz[i] = new Tacka (x, y);
                }
                return (vrs=='i' || vrs=='I') ? new Izlomljena (niz)
                                         : new Poligon (niz);
            }
            default: return null;
        }
}
```

```

public static void main (String[] varg) { // Glavna funkcija.
    Linija[] niz = new Linija [Integer.parseInt (varg[0])];
    for (int i=0; i<niz.length; ) {
        Linija lin = citaj ();
        if (lin != null) niz[i++] = lin;
        else System.out.println("*** Nepoznata vrsta linije!");
    }
    System.out.println ("\nProcitano:");
    double max = niz[0].duz (); int imax = 0;
    for (int i=0; i<niz.length; i++) {
        double d = niz[i].duz ();
        System.out.println (niz[i] + " d=" + d);
        if (d > max) { max = d; imax = i; }
    }
    System.out.println ("\nNajduza: " + niz[imax]);
}
}

```

```

% java LinijeT 3
Vrsta (D, I, P)? d
A? 1 1
B? 2 2
Vrsta (D, I, P)? i
Broj temena? 3
1. teme? 0 0
2. teme? 0 2
3. teme? 2 0
Vrsta (D, I, P)? x
*** Nepoznata vrsta linije!
Vrsta (D, I, P)? p
Broj temena? 3
1. teme? 0 0
2. teme? 1 0
3. teme? 1 1

Procitano:
1[duz: A(1.0,1.0), B(2.0,2.0)] d=1.4142135623730951
2[izlomljena: (0.0,0.0), (2.0,2.0), (2.0,0.0)] d=4.82842712474619
3[poligon: (0.0,0.0), (1.0,0.0), (1.0,1.0)] d=3.414213562373095

Najduza: 2[izlomljena: (0.0,0.0), (2.0,2.0), (2.0,0.0)]

```

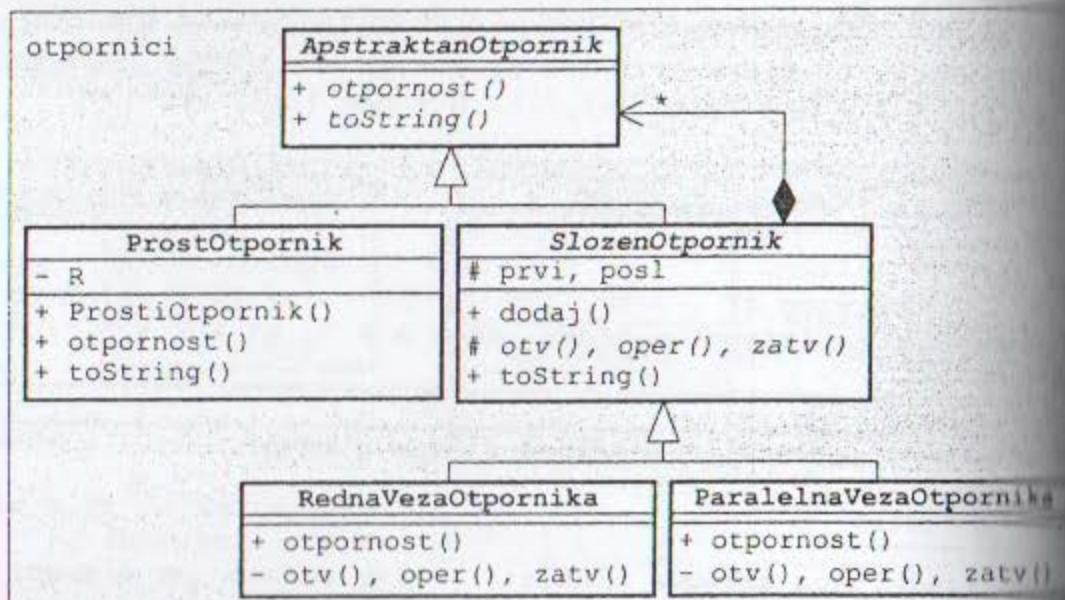
### Zadatak 5.7 Prosti i složeni otpornici; redne i paralelne veze otpornika

Napisati na jeziku Java sledeće klase:

- *Apstraktnom otporniku* može da se odredi otpornost.
- *Prostom otporniku* se zna vrednost njegove otpornosti. Tekstualni oblik sadrži vrednost.
- *Složen otpornik* je otpornik koji se sastoji od proizvoljnog broja otpornika. Stvara se prema njima mogu da mu se dodaju komponentni otpornici.
- *Redna veza otpornika* je složen otpornik. Otpornost je jednaka zbiru otpornosti sadržanih u njemu. Tekstualni oblik je  $(r+r+\dots+r)$ , gde je:  $r$  – tekstualni oblik jednog sadržanog otpornika.
- *Paralelna veza otpornika* je složen otpornik. Recipročna vrednost otpornosti je jednaka pročnih vrednosti otpornosti sadržanih otpornika. Tekstualni oblik je  $[r|r|\dots|r]$ , gde je  $r$  – oblik jednog sadržanog otpornika.

Napisati na jeziku Java klasu s funkcijom za čitanje jednog otpornika proizvoljne složenosti iz ulaza i glavnom funkcijom koja čita otpornike s glavnog ulaza, ispisuje ih na glavnom izlazu i vrati vrednostima njihove otpornosti, sve dok ne pročita "parazan" otpornik.

Rešenje:



```
// ApstraktanOtpornik.java - Klasa apstraktnih otpornika.

package otpornici;

public abstract class ApstraktanOtpornik {

    public abstract double otpornost(); // Otpornost otpornika.

    public abstract String toString(); // Tekstualni oblik.
}
```

```

// ProstOtpornik.java - Klasa prostih otpornika.

package otpornici;

public class ProstOtpornik extends ApstraktanOtpornik {

    private double R;                                // Vrednost otpornosti.

    public ProstOtpornik (double r) { R = r; } // Inicijalizacija.

    public double otpornost () { return R; } // Otpornost otpornika.

    public String toString ()                      // Tekstualni oblik.
        { return Double.toString (R); }
}

```

```

// SlozenOtpornik.java - Apstraktna klasa složenih otpornika.

package otpornici;

public abstract class SlozenOtpornik extends ApstraktanOtpornik {

    protected static class Elek {                         // Element liste otpornika:
        ApstraktniOtpornik R;                            // - sadržani otpornik,
        Elek sled;                                       // - sledeći element liste,
        Elek (ApstraktniOtpornik r) {                   // - inicijalizacija.
            R = r;
        }
    }

    protected Elek prvi, posl; // Početak i kraj liste sadržanih otpornika.

    public SlozenOtpornik dodaj (ApstraktanOtpornik r) { // Dodavanje
        Elek novi = new Elek (r);                         // otpornika.
        if (prvi == null) prvi = novi;
        else           posl.sled = novi;
        posl = novi;
        return this;
    }                                                       // Simboli za tekstualni oblik:
    protected abstract char otv ();                     // - otvorena zagrada,
    protected abstract char oper ();                    // - operator,
    protected abstract char zatv ();                   // - zatvorena zagrada.

    public final String toString () {                  // Tekstualni oblik.
        StringBuffer s = new StringBuffer ();
        s.append (otv ());
        for (Elek tek=prvi; tek!=null; tek=tek.sled) {
            s.append (tek.R);
            if (tek.sled != null) s.append (oper ());
        }
        return s.append (zatv()).toString ();
    }
}

```

```
// RednaVezaOtpornika.java - Klasa redno vezanih otpornika.

package otpornici;

public class RednaVezaOtpornika extends SlozenOtpornik {

    public double otpornost () { // Otpornost otpornika.
        double r = 0;
        for (Elem tek=prvi; tek!=null; tek=tek.sled)
            r += tek.R.otpornost ();
        return r;
    }

    protected final char otv () { return '('; } // Simboli za tekstualni
    protected final char oper () { return '+'; } // oblik.
    protected final char zatv () { return ')'; }
}
```

```
// ParalelnaVezaOtpornika.java - Klasa paralelno vezanih otpornika.

package otpornici;

public class ParalelnaVezaOtpornika extends SlozenOtpornik {

    public double otpornost () { // Otpornost otpornika.
        double s = 0;
        for (Elem tek=prvi; tek!=null; tek=tek.sled)
            s += 1 / tek.R.otpornost ();
        return 1 / s;
    }

    protected final char otv () { return '['; } // Simboli za tekstualni
    protected final char oper () { return '|'; } // oblik.
    protected final char zatv () { return ']'; }
}
```

```
// OtporniciT.java - Ispitivanje klasa otpornika.

import otpornici.*;

public class OtporniciT {

    private static ApstraktanOtpornik citaj () { // Čitanje otpornika.
        String vrs = Citaj.String ();
        if (vrs.equalsIgnoreCase ("prost"))
            return new ProstOtpornik (Citaj.Double ());
        else if (vrs.equalsIgnoreCase ("redna"))
            return citaj (new RednaVezaOtpornika ());
        else if (vrs.equalsIgnoreCase ("paralelna"))
            return citaj (new ParalelnaVezaOtpornika ());
        else
            return null;
    }
}
```

```

private static SlozenOtpornik citaj (SlozenOtpornik R) { //  

    while (true) {  

        ApstraktanOtpornik r = citaj ();  

        if (r == null) return R;  

        R.dodaj (r);  

    }  

}  
  

public static void main (String[] varg) { // Glavna funkcija  

    ApstraktanOtpornik R;  

    while ((R = citaj ()) != null)  

        System.out.println (R + " = " + R.otpornost());  

}
}

```

*OtporniciT.pod*

prost 5

redna prost 3 prost 2 prost 4 \*

paralelna prost 2 prost 4 prost 1 prost 4 \*

paralelna

redna

prost 2

paralelna prost 2 prost 2 \*

\*

redna prost 1 prost 1 prost 1 \*

redna

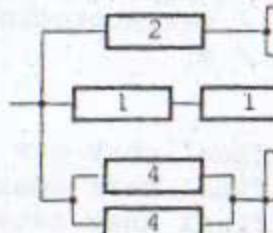
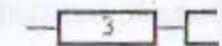
paralelna prost 4 prost 4 \*

paralelna prost 3 prost 3 prost 3 \*

\*

\*

\*



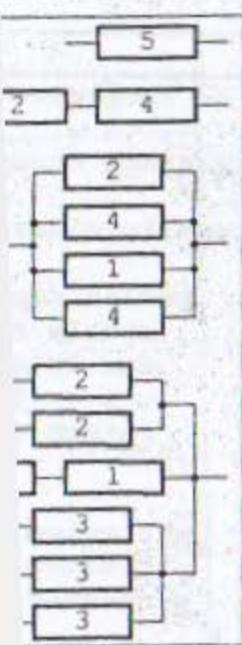
# java OtporniciT &lt;OtporniciT.pod

5.0 = 5.0

(3.0+2.0+4.0) = 9.0

[2.0|4.0|1.0|4.0] = 0.5

{(2.0+[2.0|2.0])|(1.0+1.0+1.0)|([4.0|4.0]+[3.0|3.0|3.0])} = 1.0



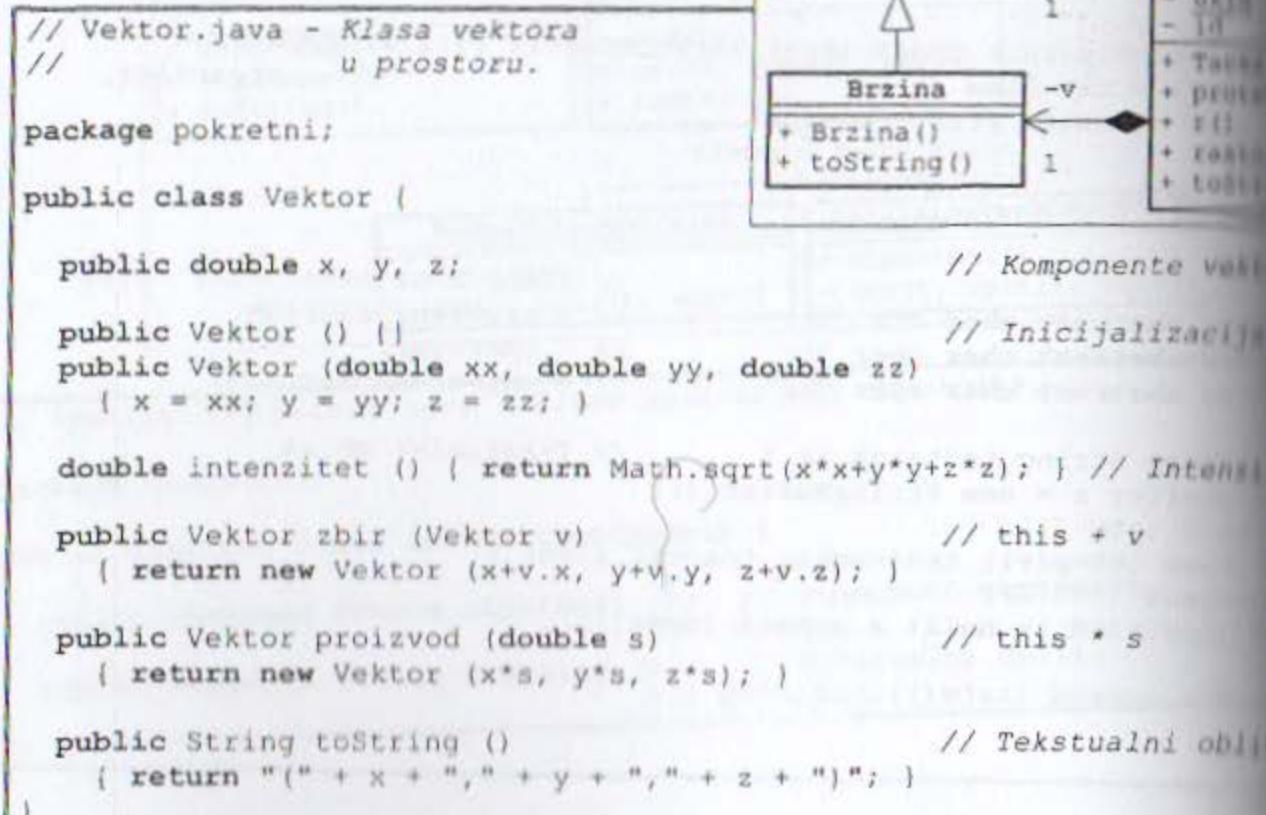
### Zadatak 5.8 Vektori, brzine, pokretni objekti i tačke u prostoru

Napisati na jeziku Java sledeće tipove:

- **Vektor** u trodimenzionalnom prostoru se predstavlja komponentama u pravcu  $x, y, z$ . Uzmevane vrednosti su  $(0,0,0)$ . Može da se izračuna intenzitet vektora, zbir dva vektora i skalarnog realnog broja. Tekstualni oblik vektora je  $(x,y,z)$ .
- **Brzina** je vektor čiji je tekstualni oblik  $v(x,y,z)$ .
- Apstraktni **pokretni** objekti mogu da promene svoj položaj u prostoru u zavisnosti od klog vremenskog intervala.
- **Tačka** je pokretan objekat koji ima jedinstven, automatski generisan, identifikacioni broj položaja i brzinu kretanja. Može da se dohvati trenutni vektor položaja tačke i da se izračuna razdalje između dve tačke. Tekstualni oblik tačke je  $T_i(x,y,z)$ , gde je:  $i$  – identifikacioni broj.

Napisati na jeziku Java program koji pročita niz tačaka s glavnog ulaza, a zatim u zadatom broju skih intervala sa zadatim korakom  $dt$  ispisuje na glavnom izlazu tačku koja je na kraju intervala najbliža koordinatnom početku i njeno rastojanje od koordinatnog početka.

Rešenje:



```
// Brzina.java - Klasa brzina.

package pokretni;

public class Brzina extends Vektor {

    public Brzina () { super (1, 0, 0); }           // Inicijalizacija.
    public Brzina (double x, double y, double z) { super (x, y, z); }

    public String toString ()                      // Tekstualni oblik.
        { return "v" + super.toString (); }
}
```

// Pokretan.java - Interfejs pokretnih objekata.

```
package pokretni;

public interface Pokretan {

    Pokretan proteklo (double dt); // Proteklo je vreme dt.
}
```

// Tacka.java - Klasa pokretnih tačaka.

```
package pokretni;

public class Tacka implements Pokretan {

    private static int ukId = 0; // Poslednje korišćeni identifikator.
    private final int id = ++ukId; // Identifikator tačke.
    private Vektor r;           // Vektor položaja.
    private Brzina v;           // Vektor brzine.

    public Tacka (Vektor rr, Brzina vv) { r = rr; v = vv; } // Inicijalizacija.
    public Tacka () { r = new Vektor (); v = new Brzina (); } // Inicijalizacija.

    public Pokretan proteklo (double dt)           // Promena položaja zbog
        { r = r.zbir(v.proizvod (dt)); return this; } // proteklog vremena.

    public Vektor r () { return r; }                  // Trenutni položaj.

    public double rastojanje (Tacka T)             // Rastojanje od druge
        { return r.zbir (T.r.proizvod(-1)).intenzitet(); } // tačke.

    public String toString() { return "T" + id + r; } // Tekstualni oblik.
}
```

```
// PokretniT.java - Ispitivanje klase pokretnih tačaka.

import pokretni.*;

public class PokretniT {
    public static void main (String[] varg) {
        System.out.print ("Broj tacaka? "); int n = Citaj.Int ();
        Tacka[] tacke = new Tacka [n];
        for (int i=0; i<n; i++) {
            System.out.print("Koordinate tmena " + i + "? ");
            double x = Citaj.Double(), y = Citaj.Double(), z = Citaj.Double();
            System.out.print ("Komponente brzine? ");
            double vx = Citaj.Double(), vy = Citaj.Double(), vz = Citaj.Double();
            tacke[i] = new Tacka (new Vektor(x,y,z), new Brzina(vx,vy,vz));
        }

        System.out.print ("\nBroj koraka? "); int k = Citaj.Int ();
        System.out.print ("Trajanje koraka? "); double dt = Citaj.Double();
        final Tacka org = new Tacka ();
        System.out.print ("ORG " + org + "\n\n");

        for (int i=0; i<k; i++) {
            for (int j=0; j<n; tacke[j++].proteklo(dt));
            double min = org.rastojanje (tacke[0]); int m = 0;
            for (int j=1; j<n; j++) {
                double d = org.rastojanje (tacke[j]);
                if (d < min) { min = d; m = j; }
            }
            System.out.println (i + " " + tacke[m] + " " + min);
        }
    }
}
```

```
$ java PokretniT
Broj tacaka? 3
Koordinate tmena 0? 2 2 2
Komponente brzine? -1 -1.2 -1.3
Koordinate tmena 1? -2 2 0
Komponente brzine? .5 -.6 .2
Koordinate tmena 2? 0 0 0
Komponente brzine? .3 .3 .3

Broj koraka? 10
Trajanje koraka? 1
ORG T4(0.0,0.0,0.0)

0 T3(0.3,0.3,0.3) 0.5196152422706632
1 T1(0.0,-0.3999999999999999,-0.6000000000000001) 0.7211102550927979
2 T2(-0.5,0.1999999999999996,0.6000000000000001) 0.806225774829855
3 T2(0.0,-0.4,0.8) 0.894427190999916
4 T2(0.5,-1.0,1.0) 1.5
5 T2(1.0,-1.6,1.2) 2.23606797749979
6 T2(1.5,-2.2,1.4) 3.0083217912982647
7 T2(2.0,-2.800000000000003,1.5999999999999999) 3.7947331922020555
8 T2(2.5,-3.400000000000004,1.799999999999998) 4.588027898781785
9 T3(2.999999999999996,2.999999999999996,2.999999999999996) 5.196152422706631
```

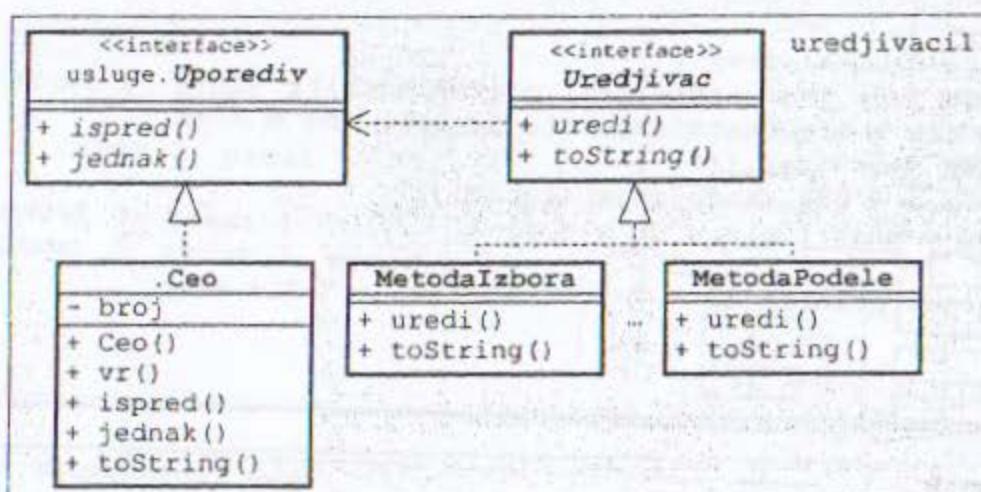
**Zadatak 5.9 Uporedivi objekti, razne vrste uređivača objekata i celi brojevi**

Napisati na jeziku Java sledeće tipove:

- Za apstraktan **uporediv** objekat može da se ispita da li se nalazi ispred drugog objekta i da li je jednak drugom objektu.
- Apstraktan **uredjivac** može da uredi niz uporedivih objekata. Tekstualni oblik sadrži naziv algoritma koji primenjuje.
- Konkretni uređivači ostvaruju sledeće algoritme uređivanja nizova: **metoda izbora, metoda umetanja, metoda zamene suseda i metoda podele**.
- **Ceo** broj je uporediv objekat koji sadrži celobrojnu vrednost koja može da se dohvati. Tekstualni oblik predstavlja sadržanu vrednost.

Napisati na jeziku Java program koji omogućava proveru ispravnosti prethodnih klasa i merenje vremena trajanja uređivanja.

**Rešenje:**



```

// Uporediv.java - Interfejs uporedivih objekata.

package usluge;

public interface Uporediv {

    boolean ispred (Uporediv u); // Da li je ispred drugog objekta?

    boolean jednak (Uporediv u); // Da li je jednak drugom objektu?
}
  
```

```

// Uredjivac.java - Interfejs algoritama uređivanja.

package uredjivacil;

public interface Uredjivac {

    void uredi (usluge.Uporediv[] niz); // Uredivanje niza.

    String toString (); // Tekstualni oblik.
}
  
```

```

// MetodaIzbora.java - Klasa za uređivanje metodom izbora
// (Selection Sort).

package uredjivacil;
import usluge.Uporediv;

public class MetodaIzbora implements Uredjivac {

    public void uredi (Uporediv[] niz) { // Uređivanje niza.
        int n = niz.length;
        for (int i=0; i<n-1; i++)
            for (int j=i+1; j<n; j++)
                if (niz[j].ispred (niz[i]))
                    { Uporediv p = niz[i]; niz[i] = niz[j]; niz[j] = p; }
    }

    public String toString () // Naziv algoritma.
    { return "Metoda izbora"; }
}

// MetodaIzbora2.java - Klasa za uređivanje poboljšanom metodom izbora.

package uredjivacil;
import usluge.Uporediv;

public class MetodaIzbora2 implements Uredjivac {

    public void uredi (Uporediv[] niz) { // Uređivanje niza.
        int n = niz.length;
        for (int i=0; i<n-1; i++) {
            int m = i;
            for (int j=i+1; j<n; j++)
                if (niz[j].ispred (niz[m])) m = j;
            if (m != i)
                { Uporediv p = niz[i]; niz[i] = niz[m]; niz[m] = p; }
        }
    }

    public String toString () // Naziv algoritma.
    { return "Poboljsana metoda izbora"; }
}

// MetodaUmetanja.java - Klasa za uređivanje metodom umetanja
// (Insertion Sort).

package uredjivacil;
import usluge.Uporediv;

public class MetodaUmetanja implements Uredjivac {

    public void uredi (Uporediv[] niz) { // Uređivanje niza.
        int n = niz.length;
        for (int i=1; i<n; i++) {
            int j = i - 1;
            while (j>=0 && niz[j+1].ispred (niz[j])) {
                Uporediv p = niz[j+1]; niz[j+1] = niz[j]; niz[j--] = p;
            }
        }
    }
}

```

```
public String toString () // Naziv algoritma.
    { return "Metoda umetanja"; }
```

```
// MetodaUmetanja2.java - Klasa za uređivanje poboljšanom metodom umetanja.

package uredjivacil;
import usluge.Uporediv;

public class MetodaUmetanja2 implements Uredjivac {

    public void uredi (Uporediv[] niz) { // Uredjivanje niza.
        int n = niz.length;
        for (int i=1; i<n; i++) {
            Uporediv p = niz[i];
            int j = i - 1;
            while (j>=0 && p.ispred (niz[j])) niz[j+1] = niz[j--];
            niz[j+1] = p;
        }
    }

    public String toString () // Naziv algoritma.
    { return "Poboljsana metoda umetanja"; }
```

```
// MetodaZameneSuseda.java - Klasa za uređivanje metodom zamene suseda
// (Bubble Sort).

package uredjivacil;
import usluge.Uporediv;

public class MetodaZameneSuseda implements Uredjivac {

    public void uredi (Uporediv[] niz) { // Uređivanje niza.
        int n = niz.length;
        boolean dalje = true;
        for (int i=0; i<n-1 && dalje; i++) {
            dalje = false;
            for (int j=n-2; j>=i; j--)
                if (niz[j+1].ispred (niz[j])) {
                    Uporediv p = niz[j+1]; niz[j+1] = niz[j]; niz[j] = p;
                    dalje = true;
                }
        }
    }

    public String toString () // Naziv algoritma.
    { return "Metoda zamene suseda"; }
```

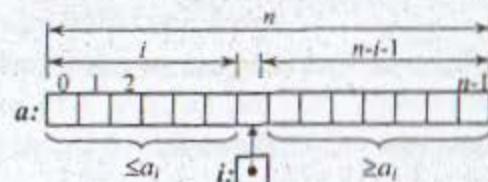
```
// MetodaPodele.java - Klasa za uređivanje metodom podele
// (Quick Sort).

package uredjivacil;
import usluge.Uporediv;

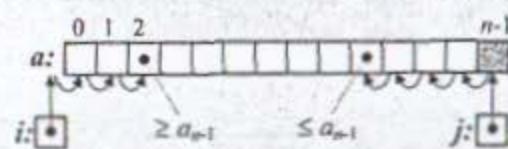
public class MetodaPodele implements Uredjivac {

    public void uredi (Uporediv[] niz)      // Uređivanje niza.
    { uredi (niz, 0, niz.length-1); }
```

Niz se podeli oko elementa  $i$  tako da levo svi brojevi budu  $\leq a_i$ , a desno svi budu  $\geq a_i$ . Levi i desni deo niza uređuju se nezavisno.



Podela se pravi premeštanjem elemenata većih od  $a_{n-1}$  s početka niza prema kraju, a elemenata manjih od  $a_{n-1}$  s kraja prema početku. Na kraju  $a_{n-1}$  se stavi negde u sredinu niza.



```
private void uredi (Uporediv[] a, int p, int q) {
    if (q > p) {
        int i = p-1, j = q;
        while (true) {
            do i++; while (a[i].ispred (a[q]));
            do j--; while (j>=0 && a[q].ispred (a[j]));
            if (i >= j) break;
            Uporediv b = a[i]; a[i] = a[j]; a[j] = b;
        }
        Uporediv b = a[i]; a[i] = a[q]; a[q] = b;
        uredi (a, p, i-1); uredi (a, i+1, q);
    }
}

public String toString ()                         // Naziv algoritma.
{ return "Metoda podele"; }
```

// Ceo.java - Klasa uporedivih celih brojeva.

```
import usluge.Uporediv;

public class Ceo implements Uporediv {

    private int broj;                           // Vrednost u objektu.

    public Ceo (int b) { broj = b; }           // Inicijalizacija.
```

```

public int vr () { return broj; } // Dohvatanje vrednosti.

public boolean ispred (Uporediv b) // Da li je ispred drugog objekta?
{ return broj < ((Ceo)b).broj; }

public boolean jednak (Uporediv b) // Da li je jednak drugom objektu?
{ return broj == ((Ceo)b).broj; }

public String toString () // Tekstualni oblik.
{ return Integer.toString (broj); }
}

```

// UredjvacilT.java - Ispitivanje klasa za uređivanje nizova.

```

import uredjivacil.*;
import usluge.Uporediv;
import java.util.Random;

public class UredjivacilT {

    // Niz za objekte algoritama uređivanja.
    private static Uredjivac[] metode = {
        new MetodaIzbora (), new MetodaIzbora2 (),
        new MetodaUmetanja (), new MetodaUmetanja2 (),
        new MetodaZameneSuseda (), new MetodaPodele ()
    };

    // Ispisivanje niza.
    private static void pisi (String nasl, Uporediv[] niz) {
        System.out.print ("\n" + nasl + "\n\n");
        for (int i=0; i<niz.length; i++) {
            System.out.print (niz[i] + "\t");
            if (i%8==7 || i==niz.length-1) System.out.println ();
        }
    }

    // Glavna funkcija.
    public static void main (String[] vpar) {
        Uredjivac metoda = metode[0];
        Uporediv[] niz = new Ceo [1000];
        Random sluc = new Random ();
        radi: while (true) { // Ispisivanje menija:
            System.out.print (
                "\n1. Zadavanje duzine niza\n"
                "2. Postavljanje generatora slucajnih brojeva\n"
                "3. Izbor algoritma\n"
                "4. Primena algoritma\n"
                "0. Kraj programa\n"
                "Vas izbor? "
            );
            switch (Citaj.Int ()) {

                case 1: // Zadavanje duzine niza:
                    System.out.print ("Duzina niza? ");
                    int d = Citaj.Int ();
                    if (d > 0) niz = new Ceo [d];
                    else System.out.println ("*** Nedozvoljena duzina!");
                    break;
            }
        }
    }
}

```

```

case 2: // Postavljanje generatora slučajnih brojeva:
System.out.print ("Pocetna vrednost generatora? ");
sluc = new Random (Citaj.Long ());
break;

case 3: // Izbor algoritma:
System.out.println ();
for (int i=0; i<metode.length; i++)
    System.out.println (i+1 + ". " + metode[i]);
System.out.print ("Vas izbor? ");
int izbor = Citaj.Int ();
if (izbor>0 && izbor<=metode.length)
    metoda = metode [izbor-1];
else System.out.println ("*** Nedozvoljen izbor!");
break;

case 4: // Primena algoritma:
for (int i=0; i<niz.length; i++)
    niz[i] = new Ceo ((int)(sluc.nextDouble() * 10000));
if (niz.length <= 100) pisi ("Pocetni niz:", niz);
long t1 = System.currentTimeMillis ();
metoda.uredi (niz);
long t2 = System.currentTimeMillis ();
if (niz.length <= 100)
    pisi ("Uredjeni niz:", niz);
else
    System.out.println (metoda + " (" + niz.length + "): " +
                        (t2-t1)/1000.);
break;

case 0: // Kraj programa:
break radi;

default: // Pogrešan izbor:
System.out.println ("*** Nedozvoljen izbor!");
break;

```

java UredjivacilT

- 1. Zadavanje duzine niza
  - 2. Postavljanje generatora slucajnih brojeva
  - 3. Izbor algoritma
  - 4. Primena algoritma
  - 0. Kraj programa

Vas izbor? 1

Duzina niza? 45

Macbeth 3

1. Metoda izbora
  2. Poboljsana metoda izbora
  3. Metoda umetanja
  4. Poboljsana metoda umetanja
  5. Metoda zamene suseda
  6. Metoda podele

Vas izbor? 3

Vas izbor? 4

Pocetni niz:

4658	9793	7595	9437	9338	5073	1757	4247
5770	7082	1419	1759	1467	696	8259	5162
8801	2453	4841	5049	2748	5983	8985	3119
8402	7997	4088	1870	4251	3662	6172	6634
2487	4036	6422	9015	796	2380	8752	8893
989	637	5565	5484	8605			

Uredjeni niz:

637	696	796	989	1419	1467	1757	1759
1870	2380	2453	2487	2748	3119	3662	4036
4086	4247	4251	4658	4841	5049	5073	5162
5484	5565	5770	5983	6172	6422	6634	7082
7595	7997	8259	8402	8605	8752	8801	8893
8985	9015	9338	9437	9793			

Vas izbor? 1

Duzina niza? 10000

Vas izbor? 4

Metoda umetanja (10000): 0.735

Vas izbor? 0

n = 10 000 (Pentium 4 / 2,4 GHz – JDK 1.5.0)

Metoda izbora: 1,324 s

Poboljšana metoda izbora: 0,630 s

Metoda umetanja: 0,728 s

Poboljšana metoda umetanja: 0,497 s

Metoda zamene suseda: 1,414 s

n = 100 000

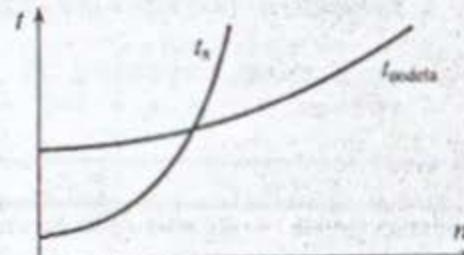
Poboljšana metoda umetanja: 105,0 s

Metoda podele: 0,063 s

n = 1 000 000

Metoda podele: 1,066 s

$$\begin{aligned} t_x &= a_2 n^2 + a_1 n + a_0 \\ t_{\text{podela}} &= b_2 n \log n + b_1 n + b_0 \\ (a_i &<< b_i) \end{aligned}$$



Metodu podele ne treba primenjivati na kratke nizove.

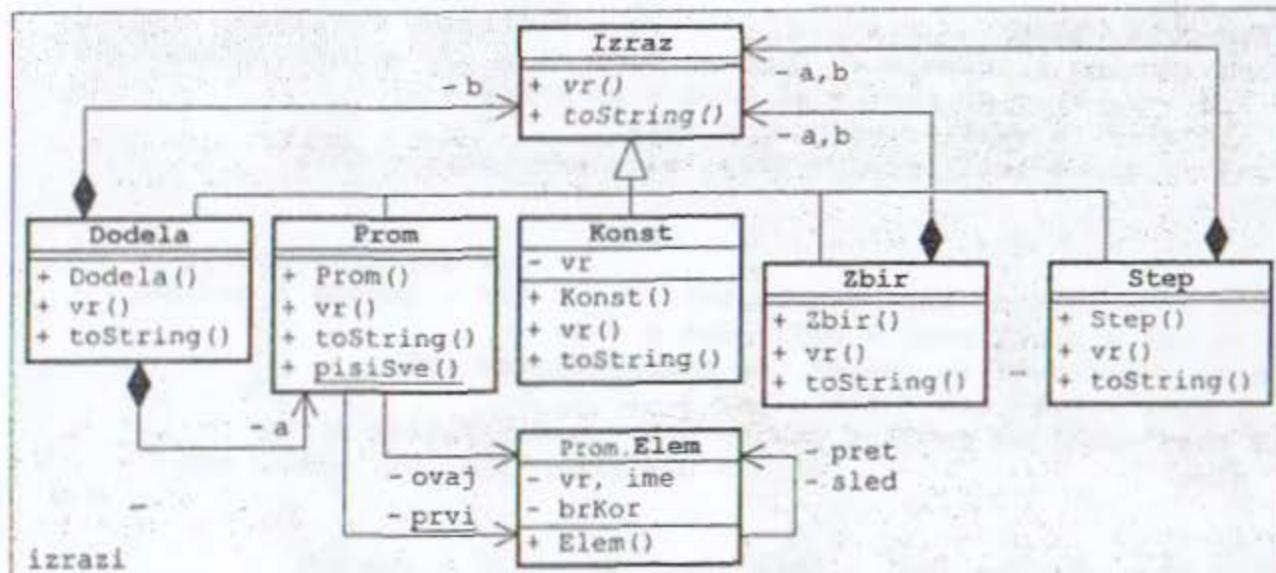
**Zadatak 5.10 Izrazi, konstante, promenljive, dodele vrednosti i aritmetičke operacije**

Napisati na jeziku Java sledeće tipove:

- Apstraktnom *izrazu* može da se izračuna vrednost realnog tipa i može da se sastavi tekstualni oblik izraza.
- *Konstanta* je izraz čija vrednost ne može da se promeni posle inicijalizacije. Tekstualni oblik sadrži vrednost konstante.
- *Promenljiva* je izraz koji ima ime i realnu vrednost (podrazumevano 0) koja može da se promeni i posle inicijalizacije. Tekstualni oblik sadrži ime promenljive. Više istoimenih promenljivih predstavljaju isti podatak. Može da se na glavnom izlazu ispiše niz imena svih postojećih promenljivih.
- *Dodela* je izraz koji se inicijalizuje jednom promenljivom (na primer: *a*) i jednim izrazom (na primer: *b*). Vrednost izraza *b* se, prilikom izračunavanja, dodeljuje promenljivoj *a*. Vrednost izraza dodele jednak je vrednosti koja je stavljen u promenljivu. Tekstualni oblik objekta dodele je (*a=b*), gde su: *a* i *b* – tekstualni oblici operanada.
- *Zbir*, *razlika*, *proizvod*, *količnik* i *stepen* su izrazi koji se inicijalizuju sa dva izraza (na primer: *a* i *b*) i čije su vrednosti jednake  $a+b$ ,  $a-b$ ,  $ab$ ,  $a/b$  i  $a^b$ . Njihovi tekstualni oblici su (*a+b*), (*a-b*), (*a\*b*), (*a/b*) i (*a^b*), gde su: *a* i *b* – tekstualni oblici operanada.

Napisati na jeziku Java program koji stvara objekat promenljive *x* i objekat za izraz  $x^3-2x$ , ispiše algebarski oblik tog izraza na glavnom izlazu i posle tabelira vrednosti tog izraza na glavnom izlazu za sve vrednosti  $x_{\min} \leq x \leq x_{\max}$  s korakom  $\Delta x$ . Parametri tabeliranja se dostavljaju kao parametri glavne funkcije.

Rešenje:



```

// Izraz.java - Apstraktna klasa izraza.

package izrazi;

public abstract class Izraz {
    public abstract double vr(); // Vrednost izraza.
    public abstract String toString(); // Tekstualni oblik izraza.
}
  
```

```
// Konst.java - Klasa konstanti.

package izrazi;

public class Konst extends Izraz {
    private final double vr; // Vrednost.

    public Konst (double v) { vr = v; } // Inicijalizacija brojem.

    public double vr () { return vr; } // Vrednost kostante.

    public String toString () { return "" + vr; } // Tekstualni oblik.
}
```

```
// Prom.java - Klasa promenljivih.

package izrazi;

public class Prom extends Izraz {

    private static Elektricnost prvi = null; // Početak liste svih promenljivih.
    private Elektricnost ovaj; // Element pridružen ovom objektu.

    private static class Elektricnost { // Element liste svih promenljivih:
        String ime; // - ime,
        double vr; // - vrednost,
        int brKor; // - broj korišćenja,
        Elektricnost pret, sled; // - prethodni i sledeći u listi.
        Elektricnost (String i, double v, Elektricnost p, Elektricnost s) { // - inicijalizacija.
            ime = i; vr = v; pret = p; sled = s; brKor = 1;
            if (sled != null) sled.pret = this;
            if (pret == null) prvi = this; else pret.sled = this;
        }
    }

    public Prom (String ime, double vr) { // - imenom i brojem,
        Elektricnost tek = prvi, pret = null; int p = 0;
        while (tek!=null && (p=tek.ime.compareTo(ime))<0)
            ( pret = tek; tek = tek.sled; )
        if (tek!=null && p==0) { ovaj = tek; ovaj.brKor++; }
        else
            ovaj = new Elektricnost (ime, vr, pret, tek);
    }

    public Prom (String ime) { this (ime, 0); } // - imenom,
    public Prom (String ime, Izraz i) // - imenom i izrazom,
    { this (ime, i.vr()); }

    public Prom (Prom p) // - kopijom promenljive.
    { ovaj = p.ovaj; ovaj.brKor++; }

    public void brisi () { // Uništavanje.
        if (--ovaj.brKor == 0) {
            if (ovaj.pret != null) ovaj.pret.sled = ovaj.sled;
            else prvi = ovaj.sled;
            if (ovaj.sled != null) ovaj.sled.pret = ovaj.pret;
        }
    }
}
```

```

        // Dodela vrednosti:
        // - realnog broja,
public Prom postavi (double v)
{ ovaj.vr = v; return this; }

        // - izraza.
public Prom postavi (Izraz i)
{ ovaj.vr = i.vr (); return this; }

        // Vrednost promenljive.
public double vr () { return ovaj.vr; }

        // Tekstualni oblik.
public String toString ()
{ return ovaj.ime; }

        // Spisak svih promenljivih.
public static void pisiSve () {
    for (Elem tek=prvi; tek!=null; tek=tek.sled)
        System.out.print (tek.ime + " ");
    System.out.println ();
}
}

```

// Dodela.java - Klasa dodela vrednosti.

```

package izrazi;

public class Dodela extends Izraz {

    private Prom a;                                // Operandi.
    private Izraz b;

    public Zbir (Izraz x, Izraz y) {
        a = x; b = y; }                           // Inicijalizacija promenljivom i izrazom.

    public double vr ()                            // Vrednost.
        { return a.postavi(b).vr(); }

    public String toString ()                     // Tekstualni oblik.
        { return "(" + a + "=" + b + ")"; }
}

```

// Zbir.java - Klasa zbirova.

```

package izrazi;

public class Zbir extends Izraz {

    private Izraz a, b;                          // Operandi.

    public Zbir (Izraz x, Izraz y)               // Inicijalizacija izrazima.
        { a = x; b = y; }

    public double vr()                           // Vrednost.
        { return a.vr() + b.vr(); }

    public String toString ()                  // Tekstualni oblik.
        { return "(" + a + "+" + b + ")"; }
}

```

```
// Razl.java - Klasa razlika.

package izrazi;

public class Razl extends Izraz {

    private Izraz a, b;                                // Operandi.

    public Razl (Izraz x, Izraz y)                      // Inicijalizacija izrazima.
        { a = x; b = y; }

    public double vr()                                  // Vrednost.
        { return a.vr() - b.vr(); }

    public String toString ()                          // Tekstualni oblik.
        { return "(" + a + "-" + b + ")"; }
}
```

```
// Proizv.java - Klasa proizvoda.

package izrazi;

public class Proizv extends Izraz {

    private Izraz a, b;                                // Operandi.

    public Proizv (Izraz x, Izraz y)                  // Inicijalizacija izrazima.
        { a = x; b = y; }

    public double vr()                                  // Vrednost proizvoda.
        { return a.vr() * b.vr(); }

    public String toString ()                          // Tekstualni oblik.
        { return "(" + a + "*" + b + ")"; }
}
```

```
// Kolic.java - Klasa količnika.

package izrazi;

public class Kolic extends Izraz {

    private Izraz a, b;                                // Operandi.

    public Kolic (Izraz x, Izraz y)                  // Inicijalizacija izrazima.
        { a = x; b = y; }

    public double vr()                                  // Vrednost.
        { return a.vr() / b.vr(); }

    public String toString ()                          // Tekstualni oblik.
        { return "(" + a + "/" + b + ")"; }
}
```

```
// Step.java - Klasa stepena.

package izrazi;

public class Step extends Izraz {

    private Izraz a, b; // Operandi.

    public Step (Izraz x, Izraz y) // Inicijalizacija izrazima.
        { a = x; b = y; }

    public double vr() // Vrednost.
        { return Math.pow (a.vr(), b.vr()); }

    public String toString () // Tekstualni oblik.
        { return "(" + a + "^" + b + ")"; }
}
```

```
// IzraziT.java - Ispitivanje klase za izraze.

import izrazi.*;

public class IzraziT {
    public static void main (String[] varg) {
        Prom x = new Prom ("x"),
            xmin = new Prom ("xmin", Double.parseDouble(varg[0])),
            xmax = new Prom ("xmax", Double.parseDouble(varg[1])),
            dx = new Prom ("dx", Double.parseDouble(varg[2]));
        Izraz izr = new Razl (
            new Step (x, new Konst(3)),
            new Proizv (new Konst(2), x)
        );
        Zbir korak = new Zbir (x, dx);
        System.out.println (xmin + "=" + xmin.vr() + ", " +
                            xmax + "=" + xmax.vr() + ", " + dx + "=" + dx.vr());
        System.out.println ("\n" + x + "\t" + izr + "\n=====");
        for (x.postavi(xmin); x.vr() <= xmax.vr(); x.postavi(korak))
            System.out.println (x.vr() + "\t" + izr.vr());
    }
}
```

```
% java IzraziT -5 5 1
xmin=-5.0, xmax=5.0, dx=1.0

x ((x^3.0)-(2.0*x))
=====
-5.0 -115.0
-4.0 -56.0
-3.0 -21.0
-2.0 -4.0
-1.0 1.0
0.0 0.0
1.0 -1.0
2.0 4.0
3.0 21.0
4.0 56.0
5.0 115.0
```

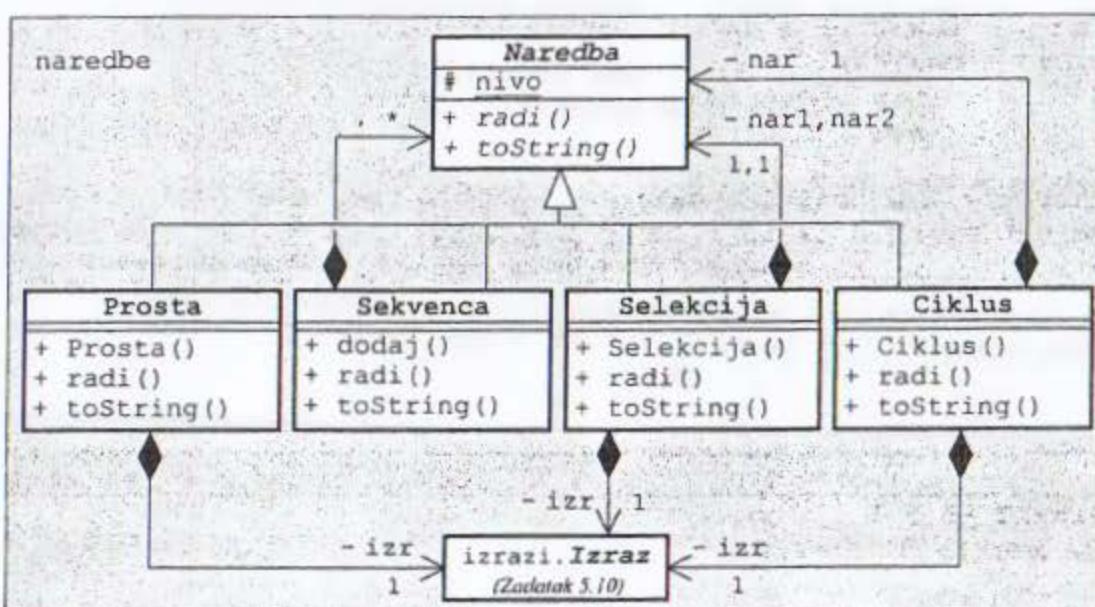
### Zadatak 5.11 Naredbe, proste naredbe, sekvence i ciklusi

Koristeći tipove za izraze iz prethodnog zadatka, napisati na jeziku Java sledeće tipove:

- Apstraktna **naredba** može da se izvrši i da se sastavi tekstualni oblik naredbe.
- **Prosta** naredba je naredba koja sadrži jedan izraz, može da bude i prazna. Izvršavanje se sastoji od izračunavanja izraza. Tekstualni oblik je *izr*;, gde je *izr* tekstualni oblik sadržanog izraza.
- **Sekvenca** je naredba koja sadrži proizvoljan broj naredbi. Stvara se prazna, a sadržane naredbe se dodaju jedna po jedna i izvršavaju po redosledu dodavanja. Tekstualni oblik sekvence je {*nar* ... *nar*}, gde je *nar* tekstualni oblik jedne sadržane naredbe.
- **Selekcija** je naredba koja sadrži jedan izraz i dve naredbe. Pri izvršavanju izračuna se vrednost izraza i ako je  $\neq 0$  izvršava se prva sadržana naredba, a ako je  $= 0$  izvršava se druga sadržana naredba. Tekstualni oblik je **if** (*izr*) *nar<sub>1</sub>*; **else** *nar<sub>2</sub>*, gde su: *izr* – tekstualni oblik sadržanog izraza, a *nar<sub>1</sub>* i *nar<sub>2</sub>* – tekstualni oblici sadržanih naredbi.
- **Ciklus** je naredba koja sadrži jedan izraz i jednu naredbu. Pri izvršavanju sadržana naredba se izvršava sve dok je vrednost sadržanog izraza  $\neq 0$ . Tekstualni oblik je **while** (*izr*) *nar*, gde su: *izr* – tekstualni oblik sadržanog izraza, a *nar* – tekstualni oblik sadržane naredbe.

Napisati na jeziku Java program koji stvara objekat naredbe koja predstavlja program za izračunavanje  $n!$ , ispiše naredbu na glavnom izlazu i posle toga čita vrednost za  $n$  s glavnog ulaza, izračunava  $n!$  i ispisuje dobijeni rezultat na glavnom izlazu, sve dok za  $n$  ne pročita negativnu vrednost.

**Rešenje:**



```

// Naredba.java - Apstraktna klasa naredbi.

package naredbe;

public abstract class Naredba {

    public abstract void radi(); // Izvršavanje.

    protected static String nivo = ""; // Nivo naredbe u tekstualnom obliku.

    public abstract String toString(); // Tekstualni oblik.
}
  
```

```
// Prosta.java - Klasa prostih naredbi.

package naredbe;
import izrazi.Izraz; → Zadatak 5.10

public class Prosta extends Naredba {
    Izraz izr;                                // Izraz koji se izračunava.

    public Prosta () {}                      // Prazna naredba.

    public Prosta (Izraz i) { izr = i; }      // Inicijalizacija izrazom.

    public void radi () {                     // Izvršavanje.
        if (izr != null) izr.vr();
    }

    public String toString () {               // Tekstualni oblik.
        return nivo + (izr != null ? izr : "") + "\n";
    }
}
```

```
// Sekvenca.java - Klasa sekvenci.

package naredbe;

public class Sekvenca extends Naredba {
    private Elem prva, posl;                  // Prva i poslednja naredba.

    private class Elem {                      // Element sekvence:
        Naredba nar;                         // - sadržana naredba,
        Elem sled;                           // - sledeći element liste,
        Elem (Naredba n) {                  // - inicijalizacija.
            nar = n;
            if (prva == null) prva = this;
            else posl.sled = this;
            posl = this;
        }
    }

    public Sekvenca dodaj (Naredba n) {       // Dodavanje naredbe.
        new Elem (n); return this;
    }

    public void radi () {                    // Izvršavanje.
        for (Elem tek=prva; tek!=null; tek=tek.sled) tek.nar.radi ();
    }

    public String toString () {             // Tekstualni oblik.
        String s = nivo + "{\n";
        nivo += " ";
        for (Elem tek=prva; tek!=null; tek=tek.sled) s += tek.nar;
        nivo = nivo.substring (2);
        return s + nivo + "}\n";
    }
}
```

```
// Selekcija.java - Klasa selekcija.

package naredbe;
import izrazi.Izraz; → Zadatak 5.10

public class Selekcija extends Naredba {

    private Izraz izr;           // Uslov selekcije.
    private Naredba nar1, nar2;  // Uslovno izvršavane naredbe.

    public Selekcija (Izraz i, Naredba nl, Naredba n2) // Inicijalizacija.
        { izr = i; nar1 = nl; nar2 = n2; }

    public void radi ()          // Izvršavanje.
        { if (izr.vr () != 0) nar1.radi (); else nar2.radi (); }

    public String toString ()    // Tekstualni oblik.
        {
            String s = nivo + "if(" + izr + ")\n";
            nivo += " "; s += nar1; nivo = nivo.substring (2);
            s += nivo + "else\n";
            nivo += " "; s += nar2; nivo = nivo.substring (2);
            return s;
        }
}
```

```
// Ciklus.java - Klasa ciklusa.

package naredbe;
import izrazi.Izraz; → Zadatak 5.10

public class Ciklus extends Naredba {

    private Izraz izr;           // Uslov ciklusa.
    private Naredba nar;         // Sadržaj ciklusa.

    public Ciklus (Izraz i, Naredba n) // Inicijalizacija.
        { izr = i; nar = n; }

    public void radi ()          // Izvršavanje.
        { while (izr.vr () != 0) nar.radi (); }

    public String toString ()    // Tekstualni oblik.
        {
            String s = nivo + "while(" + izr + ")\n";
            nivo += " "; s += nar; nivo = nivo.substring (2);
            return s;
        }
}
```

```
// NaredbeT.java - Ispitivanje klasa naredbi.

import izrazi.*; → Zadatak 5.10
import naredbe.*;

public class NaredbeT {
    public static void main (String[] vpar) {
        Prom n = new Prom ("n"), i = new Prom ("i"), f = new Prom ("f");
        Sekvenca telo = new Sekvenca ();
        telo.dodaj (new Prosta (new Dodela (i, new Zbir(i, new Konst (1)))));
        telo.dodaj (new Prosta (new Dodela (f, new Proizv (f, i))));
        Ciklus cikl = new Ciklus (new Razl (n, i), telo);
        Sekvenca prog = new Sekvenca ();
        prog.dodaj (new Prosta (new Dodela (i, new Konst (0))));
        prog.dodaj (new Prosta (new Dodela (f, new Konst (1))));
        prog.dodaj (cikl);
        System.out.println ("Program:\n" + prog);
        while (true) {
            System.out.print ("n? "); n.postavi (Citaj.Int ());
            if (n.vr () < 0) break;
            prog.radi ();
            System.out.println ("f= " + f.vr ());
        }
    }
}
```

```
% java NaredbeT
Program:
{
    (i=0.0);
    (f=1.0);
    while((n-i))
    {
        (i=(i+1.0));
        (f=(f*i));
    }
}

n? 0
f= 1.0
n? 4
f= 24.0
n? 8
f= 40320.0
n? 10
f= 3628800.0
n? -1
```

Uvod u programiranje u C++ je predstavljen u poglavljima 1 do 4. U ovom poglavju se detaljnije razgovara o konstruktorima i destruktorema. Konstruktor je funkcija koja se poziva kada se stvara objekat klase. Destruktor je funkcija koja se poziva kada se slobodi memorije koju je rezervisana za objekat klase. Osim toga, u ovom poglavju će se detaljnije razgovarati o operatorima, uključujući i operatorima prečitavanja i pisanja.

## 6 Izuzeci

**Zadatak 6.1 Vektor realnih brojeva sa zadatim opsezima indeksa**

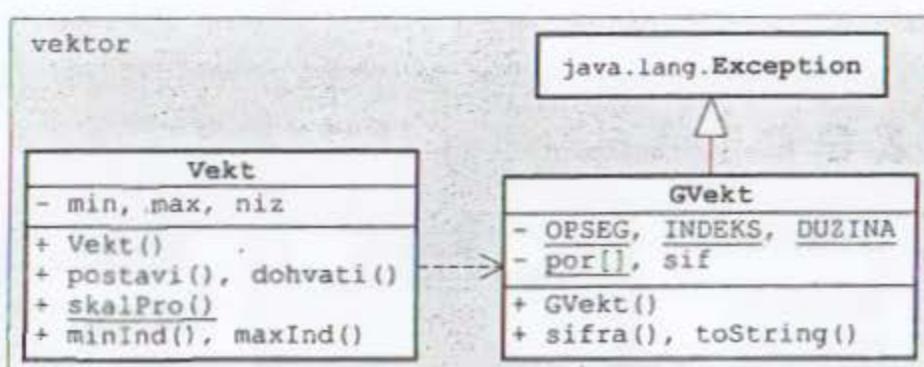
Napisati na jeziku Java klasu vektora realnih brojeva sa zadatim opsezima indeksa. Predviđeti:

- stvaranje vektora s opsegom indeksa od zadate donje do zadate gornje granice, od 1 do zadate gornje granice i od 1 do 10,
- postavljanje i dohvatanje vrednosti komponente sa zadatim indeksom,
- izračunavanje skalarnog proizvoda dva vektora, i
- dohvatanje graničnih vrednosti indeksa.

Za razrešavanje konfliktnih situacija koristiti mehanizam izuzetaka.

Napisati na jeziku Java program za ispitivanje prethodne klase.

**Rešenje:**



```

// GVekt.java - Klasa grešaka pri radu s vektorima.

package vektor;

public class GVekt extends Exception {
    // Kodovi grešaka:
    public static final int OPSEG = 0, // - neispravan opseg indeksa,
                               INDEKS = 1, // - indeks izvan opsega,
                               DUZINA = 2; // - neusaglašene dužine vektora.

    private static final String[] por = { // Tekstovi poruka o greškama.
        "Neispravan opseg indeksa!",
        "Indeks je izvan opsega!",
        "Neusaglasene duzine vektora!"
    };

    private int sif; // Šifra greške.

    public GVekt (int s) { sif = s; } // Inicijalizacija.

    public int sifra () { return sif; } // Dohvatanje šifre.
                                      // Tekstualni oblik.

    public String toString () { return "**** " + por[sif]; }
}
  
```

```

// Vekt.java - Klasa vektora realnih brojeva.

package vektor;

public class Vekt {

    int min, max;                                // Granice indeksa.
    double[] niz;                                 // Niz komponenata.
    // Inicijalizacija:
    public Vekt (int mi, int ma) throws GVekt { // - zadati opseg,
        if (mi > ma) throw new GVekt (GVekt.OPSEG);
        niz = new double [(max=ma) - (min=mi) +1];
    }

    public Vekt (int ma) throws GVekt {           // - zadata gornja granica,
        this (1, ma); }

    public Vekt () throws GVekt { this (1,10); } // - podrazumevani opseg.

    public Vekt postavi (int i, double b) throws GVekt { // Postavljanje
        if (i<min || i>max) throw new GVekt (GVekt.INDEKS); // komponente.
        niz[i-min] = b;
        return this;
    }

    public double dohvati (int i) throws GVekt      // Dohvatanje
        if (i<min || i>max) throw new GVekt (GVekt.INDEKS); // komponente.
        return niz[i-min];
    }

    public static double skalPro (Vekt v1, Vekt v2)      // Skalarni
        throws GVekt {                                     // proizvod.
        if (v1.niz.length != v2.niz.length) throw new GVekt (GVekt.DUZINA);
        double s = 0;
        for (int i=0; i<v1.niz.length; s+=v1.niz[i]*v2.niz[i++]);
        return s;
    }

    public int minInd () { return min; }                // Dohvatanje najmanjeg
    public int maxInd () { return max; }                // i najvecog indeksa.
}

```

```
// VektT.java - Ispitivanje klase realnih vektora.

import vektor.*;

public class VektT {
    public static void main (String[] varg) {
        while (true) {
            try {
                System.out.print ("Opseg indeksa prvog vektora? ");
                int min = Citaj.Int (), max = Citaj.Int ();
                if (min==0 && max==0) break;
                Vekt v1 = new Vekt (min, max);
                System.out.print ("Komponente prvog vektora? ");
                for (int i=min; i<=max; v1.postavi(i++, Citaj.Double()));
                System.out.print ("Opseg indeksa drugog vektora? ");
                min = Citaj.Int (); max = Citaj.Int ();
                if (min==0 && max==0) break;
                Vekt v2 = new Vekt (min, max);
                System.out.print ("Komponente prvog vektora? ");
                for (int i=min; i<=max; v2.postavi(i++, Citaj.Double()));
                System.out.println ("Skalarni proizvod = " +
                    Vekt.skalPro (v1, v2) + "\n");
            } catch (GVekt g) {
                System.out.println (g + "\n");
            } catch (OutOfMemoryError g) {
                System.out.println ("*** Dodela memorije nije uspela!\n");
            }
        }
    }
}
```

```
% java VektT
Opseg indeksa prvog vektora? 1 5
Komponente prvog vektora? 1 2 3 4 5
Opseg indeksa drugog vektora? -5 -1
Komponente prvog vektora? 5 4 3 2 1
Skalarni proizvod = 35.0

Opseg indeksa prvog vektora? 5 1
*** Neispravan opseg indeksa!

Opseg indeksa prvog vektora? 10 12
Komponente prvog vektora? 11 22 33
Opseg indeksa drugog vektora? -2 2
Komponente prvog vektora? 10 20 30 40 50
*** Neusaglasene duzine vektora!

Opseg indeksa prvog vektora? 0 2000000000
*** Dodela memorije nije uspela!

Opseg indeksa prvog vektora? 0 0
```

### Zadatak 6.2 Verižni razlomci

Napisati na jeziku Java klasu verižnih razlomaka prema priloženom izrazu. Predviđeti:

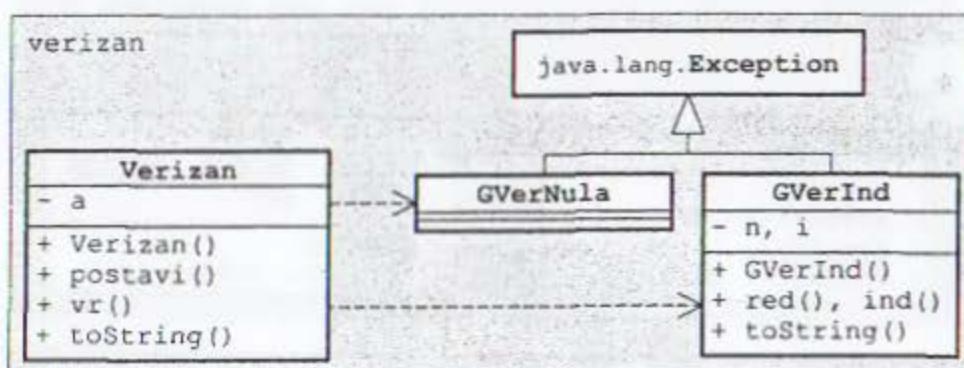
- stvaranje verižnog razlomka zadatog reda s koeficijentima jednakim nuli,
- postavljanje vrednosti zadatog koeficijenta na zadatu vrednost,
- izračunavanje vrednosti verižnog razlomka za zadatu vrednost nezavisno promenljive, i
- sastavljanje tekstualnog oblika verižnog razlomka koji sadrži koeficijente razlomka.

$$v(x) = \frac{a_0}{x + \frac{a_1}{x + \frac{\dots}{x + \frac{a_{n-2}}{x + \frac{a_{n-1}}{x}}}}}$$

Konfliktne situacije (nedozvoljen indeks koeficijenta, pokušaj deljenja nulom) prijavljivati izuzecima.

Napisati na jeziku Java program za ispitivanje prethodne klase.

Rešenje:



// GVerNula.java - Klasa za greške: Deljenje nulom u verižnom razlomku.

```

package verižan;

public class GVerNula extends Exception {} 
```

// GVerInd.java - Klasa za greške: Nedozvoljen indeks u verižnom razlomku.

```

package verižan;

public class GVerInd extends Exception {

    private int n, i; // Red razlomka i nedozvoljeni indeks.

    public GVerInd (int n, int i) // Inicijalizacija.
        { this.n = n; this.i = i; }

    public int red () { return n; } // Dohvatanje atributa.
    public int ind () { return i; }

    public String toString () { // Tekstualni oblik.
        return "GRESKA: Nedozvoljeni indeks " + i +
               " u verižnom razlomku reda " + n;
    }
} 
```

```

// Verizan.java - Klasa verižnih razlomaka.

package verizan;

public class Verizan {
    double[] a;                                // Koeficijenti razlomka.

    public Verizan (int n) { a = new double [n]; } // Inicijalizacija.

    public Verizan postavi (int i, double b) throws GVerInd { // Postavljanje
        if (i<0 || i>=a.length) throw new GVerInd(a.length, i); // koefici-
        a[i] = b;                                         // jenta.
        return this;
    }

    public double vr (double x) throws GVerNula { // Vrednost razlomka.
        double s = 0;
        for (int i=a.length-1; i>=0; i--) {
            if (x + s == 0) throw new GVerNula ();
            s = a[i] / (x + s);
        }
        return s;
    }

    public String toString () {                   // Tekstualni oblik.
        String s = "Veriz[";
        for (int i=0; i<a.length; i++) {
            if (i > 0) s += ",";
            s += a[i];
        }
        return s + "]";
    }
}

```

```

// VerizanT.java - Ispitivanje klase verižnih razlomaka.

import verizan;

public class VerizanT {
    public static void main (String[] varg) {
        while (true) {
            System.out.print ("\nn? "); int n = Citaj.Int ();
            if (n <= 0) break;
            Verizan v = new Verizan (n);
            while (true) {
                System.out.print ("i, a[i]? ");
                try {
                    v.postavi (Citaj.Int(), Citaj.Double());
                } catch (GVerInd g) {
                    if (g.ind () == -1) break;
                    System.out.println (g);
                }
            }
            System.out.println ("\n" + v);

            System.out.print ("\nxmin, xmax, dx? ");
            double xmin = Citaj.Double(), xmax = Citaj.Double(),
                   dx = Citaj.Double();
        }
    }
}

```

```

        System.out.println ("\nx\tv(x)\n=====");
        for (double x=xmin; x<=xmax; x+=dx) {
            System.out.print (x + "\t");
            try {
                System.out.println (v.vr (x));
            } catch (GVerNula g) {
                System.out.println ("Ne moze!");
            }
        }
    }
}

```

```

$ java VerizanT

n? 5
i, a[i]? 3 2
i, a[i]? 5 0
GRESKA: Nedozvoljeni indeks 5 u veriznom razlomku reda 5
i, a[i]? 1 4
i, a[i]? 2 3
i, a[i]? 4 1
i, a[i]? 0 5
i, a[i]? -1 0

Veriz[5.0,4.0,3.0,2.0,1.0]

xmin, xmax, dx? -1 1 .25

x  v(x)
=====
-1.0 -1.923076923076923
-0.75 -2.130199589628801
-0.5 -2.597864768683274
-0.25 -4.323919020244939
0.0 Ne moze!
0.25 4.323919020244939
0.5 2.597864768683274
0.75 2.130199589628801
1.0 1.923076923076923

n? 0

```

### Zadatak 6.3 Matrice realnih brojeva

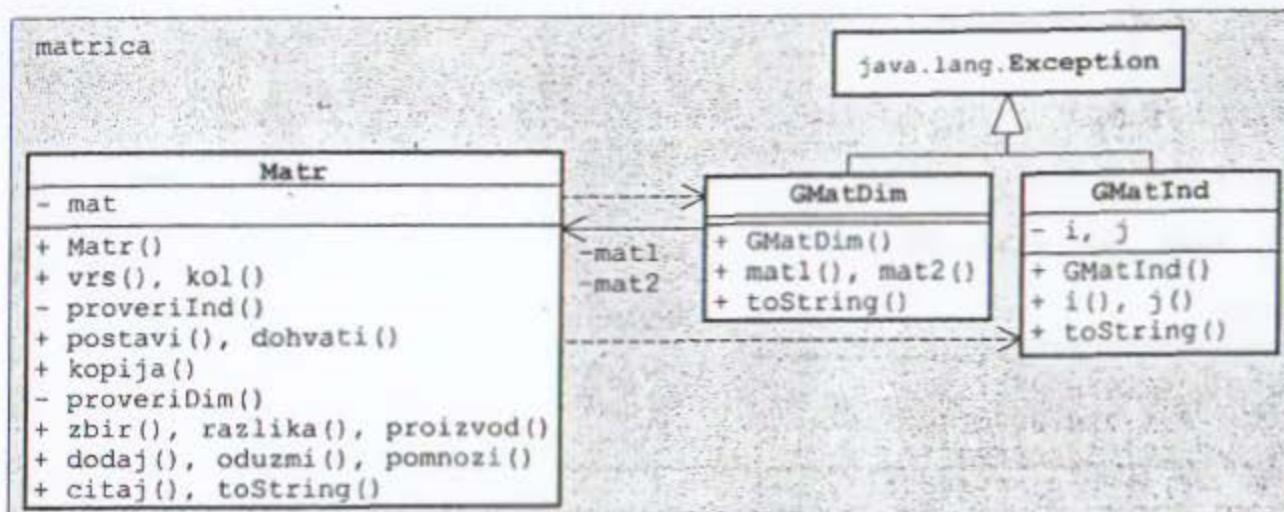
Napisati na jeziku Java klasu matrica realnih brojeva. Predvideti:

- stvaranje matrica zadatih dimenzija (podrazumevano 10x10),
- postavljanje i dohvatanje vrednosti zadatog elementa,
- stvaranje kopije matrice,
- izračunavanje zbiru, razlike i proizvoda dve matrice,
- dodavanje i oduzimanje druge matrice i množenje drugom matricom,
- čitanje matrice s glavnog ulaza, i
- sastavljanje tekstualnog oblika matrice.

Konfliktne situacije (nedozvoljeni indeks elementa, neusaglašene dimenzije matrica) prijavljivati izuzecima.

Napisati na jeziku Java interaktivni program za ispitivanje prethodne klase.

**Rešenje:**



```

// GMatInd.java - Klasa za greške: Nedozvoljeni indeks.

package matrica;

public class GMatInd extends Exception {

    private int i, j; // Nedozvoljeni indeksi.

    public GMatInd (int ii, int jj) { i = ii; j = jj; } // Inicijalizacija.

    public int i () { return i; } // Dohvatanje nedozvoljenih indeksa.
    public int j () { return j; }

    public String toString () // Stvaranje tekstualnog oblika.
        { return "**** Nedozvoljeni indeksi [" + i + "][" + j + "]"; }
}

```

```

// GMatDim.java - Klasa za greške: Neusaglašene dimenzijs.
package matrica;

public class GMatDim extends Exception {
    private Matr mat1, mat2;           // Problematične matrice.

    public GMatDim (Matr m1, Matr m2)   // Inicijalizacija.
        { mat1 = m1; mat2 = m2; }

    public Matr mat1 () { return mat1; } // Dohvatanje problematičnih
    public Matr mat2 () { return mat2; } // matrica.

    public String toString () {          // Stvaranje tekstualnog oblika.
        return "**** Neusaglasene dimenzijs matrica [" +
            mat1.mat.length + "][" + mat1.mat[0].length + "] i [" +
            mat2.mat.length + "][" + mat2.mat[0].length + "]";
    }
}

```

```

// Matr.java - Klasa matrica realnih brojeva.

package matrica;
import usluge.Citaj;

public class Matr {
    double[][] mat;                  // Elementi matrice.

    public Matr (int m, int n) { mat = new double[m][n]; } // Inicijaliza-
                                                // cija.

    public Matr () { this (10, 10); }

    public int vrs () { return mat.length; }                   // Broj vrsta.

    public int kol () { return mat[0].length; }                // Broj kolona.

    private void proveriInd (int i, int j) throws GMatInd { // Provera
        if (i<0 || i>=mat.length || j<0 || j>mat[0].length) // indeksa.
            throw new GMatInd (i, j);
    }                                                       // Postavljanje elementa.

    public Matr postavi (int i, int j, double b) throws GMatInd
        { proveriInd (i, j); mat[i][j] = b; return this; }

    public double dohvati (int i, int j) throws GMatInd      // Dohvatanje
        { proveriInd (i, j); return mat[i][j]; }               // elementa.

    public Matr kopija () {                                     // Stvaranje kopije.
        Matr m = new Matr (mat.length, mat[0].length);
        for (int i=0; i<mat.length; i++)
            for (int j=0; j<mat[0].length; j++)
                m.mat[i][j] = mat[i][j];
        return m;
    }
}

```

```

private void proveriDim (Matr m2) throws GMatDim {           // Provera
    if (mat.length != m2.mat.length ||                      // dimenzija.
        mat[0].length != m2.mat[0].length)
        throw new GMatDim (this, m2);
}

public Matr dodaj (Matr m2) throws GMatDim { // Dodavanje matrice.
    proveriDim (m2);
    for (int i=0; i<mat.length; i++)
        for (int j=0; j<mat[0].length; j++)
            mat[i][j] += m2.mat[i][j];
    return this;
}                                                       // Zbir matrica.

public static Matr zbir (Matr m1, Matr m2) throws GMatDim {
    return m1.kopija ().dodaj (m2);
}

public Matr oduzmi (Matr m2) throws GMatDim { // Oduzimanje matrice.
    proveriDim (m2);
    for (int i=0; i<mat.length; i++)
        for (int j=0; j<mat[0].length; j++)
            mat[i][j] -= m2.mat[i][j];
    return this;
}                                                       // Razlika matrica.

public static Matr razlika (Matr m1, Matr m2) throws GMatDim {
    return m1.kopija ().oduzmi (m2);
}                                                       // Proizvod matrica.

public static Matr proizvod (Matr m1, Matr m2) throws GMatDim {
    if (m1.mat[0].length != m2.mat.length)
        throw new GMatDim (m1, m2);
    Matr m = new Matr (m1.mat.length, m2.mat[0].length);
    for (int i=0; i<m1.mat.length; i++)
        for (int k=0; k<m2.mat[0].length; k++)
            for (int j=0; j<m2.mat.length; j++)
                m.mat[i][k] += m1.mat[i][j] * m2.mat[j][k];
    return m;
}

public Matr pomnozi (Matr m2) throws GMatDim{ // Množenje matricom.
    Matr m = proizvod (this, m2); mat = m.mat; return this;
}

public Matr citaj () {                                     // Čitanje matrice.
    System.out.print ("Vrs, kol? ");
    int vrs = Citaj.Int (), kol = Citaj.Int ();
    mat = new double [vrs][kol];
    for (int i=0; i<vrs; i++) {
        System.out.print ("Vrsta " + i + "? ");
        for (int j=0; j<kol; mat[i][j++]=Citaj.Double ());
    }
    return this;
}

```

```

public String toString () {           // Sastavljanje tekstualnog oblika.
    StringBuffer s = new StringBuffer ();
    for (int i=0; i<mat.length; i++){
        for (int j=0; j<mat[0].length; s.append(mat[i][j++]).append(' '));
        s.append('\n');
    }
    return s.toString ();
}
}

```

// MatricaT.java - Ispitivanje klase matrica realnih brojeva.

```

import matrica.*;
import usluge.Citaj;

class MatricaT {
    public static void main (String[] varg) {
        Matr[] mat = new Matr [Integer.parseInt (varg[0])];
        for (int i=0; i<mat.length; mat[i++]=new Matr ());
        int ml = 0;
        radi: while (true) {
            System.out.print (
                "\n1. Izbor matrice      5. Zbir matrica\n" +
                "2. Unos matrice       6. Razlika matrica\n" +
                "3. Prikaz matrice     7. Proizvod matrica\n" +
                "4. Postavljanje elementa 8. Prikaz elementa\n" +
                "0. Kraj rada\n\n" +
                "Vas izbor? "
            );
            try {
                int izb = Citaj.Int ();
                switch (izb) {
                    case 1: // Izbor matrice:
                        System.out.print ("Indeks? ");
                        ml = Citaj.Int ();
                        break;
                    case 2: // Unos matrice:
                        mat[ml].citaj (); break;
                    case 3: // Prikaz matrice:
                        System.out.print (mat[ml]); break;
                    case 4: // Postavljanje elementa:
                        System.out.print ("Indeksi? ");
                        int i = Citaj.Int (), j = Citaj.Int ();
                        System.out.print ("Vrednost? ");
                        mat[ml].postavi (i, j, Citaj.Double ());
                        break;
                    case 5: case 6: case 7: // Zbir, razlika i proizvod matrica:
                        System.out.print ("Ind. druge matr.? ");
                        int m2 = Citaj.Int (); Matr m = null;
                        switch (izb) {
                            case 5: m = Matr.zbir (mat[ml], mat[m2]); - break;
                            case 6: m = Matr.razlika (mat[ml], mat[m2]); break;
                            case 7: m = Matr.proizvod (mat[ml], mat[m2]); break;
                        }
                        System.out.print (m);
                        break;
                }
            }
        }
    }
}

```

```
        case 8: // Prikaz elementa:  
            System.out.print ("Indeksi? ");  
            System.out.println ("Vrednost= " +  
                               mat[m1].dohvati(Citaj.Int(),Citaj.Int()));  
            break;  
        case 0: // Kraj rada.  
            break radi;  
        default: // Nedozvoljeni izbor.  
            throw new Exception ("Nedozvoljeni izbor \\" + izb +"\\"");  
    }  
} catch (Exception g) { System.out.println (g); }  
}
```

java MatricaT 4

1. Izbor matrice	5. Zbir matrica
2. Unos matrice	6. Razlika matrica
3. Prikaz matrice	7. Proizvod matrica
4. Postavljanje elemenata	8. Prikaz elementa
0. Kraj rada	

Vas izbor? 2  
Vrs, kol? 3 4  
Vrsta 0? 1 2 3 4  
Vrsta 1? 2 3 4 5  
Vrsta 2? 3 4 5 6

Vas izbor? 3  
1.0 2.0 3.0 4.0  
2.0 3.0 4.0 5.0  
3.0 4.0 5.0 6.0

Vas izbor? 1  
Indeks? 2

Vas izbor? 2  
Vrs, kol? 2 3  
Vrsta 0? 1 2 3  
Vrsta 12 2 3 4

Was ist das? 5

Ind. druge matr.? 0  
\*\*\* Neusaglasene dimenzije matrica [2][3] i [3][4]

Vas izbor? 7  
Ind. druge matr.? 0  
14.0 20.0 26.0 32.0  
20.0 29.0 38.0 47.0

Vas izbor? 2  
Vrs, kol? 3 4  
Vrstva 0? 11 12 13 14  
Vrstva 1? 21 22 23 24  
Vrstva 2? 31 32 33 34

```

Vas izbor? 5
Ind. druge matr.? 0
12.0 14.0 16.0 18.0
23.0 25.0 27.0 29.0
34.0 36.0 38.0 40.0
...
Vas izbor? 6
Ind. druge matr.? 0
12.0 14.0 16.0 18.0
23.0 25.0 27.0 29.0
34.0 36.0 38.0 40.0
...
Vas izbor? 7
Ind. druge matr.? 0
*** Neusaglasene dimenzije matrica [3][4] i [3][4]
...
Vas izbor? 4
Indeksi? 0 0
Vrednost? -1
...
Vas izbor? 3
-1.0 12.0 13.0 14.0
21.0 22.0 23.0 24.0
31.0 32.0 33.0 34.0
...
Vas izbor? 8
Indeksi? 2 2
Vrednost= 33.0
...
Vas izbor? 8
Indeksi? 3 3
*** Nedozvoljeni indeksi [3][3]
...
Vas izbor? 1
Indeks? 8
...
Vas izbor? 2
java.lang.ArrayIndexOutOfBoundsException: 8
...
Vas izbor? 99
java.lang.Exception: Nedozvoljeni izbor "99"
...
Vas izbor? a
java.lang.NumberFormatException: For input string: "a"
...
Vas izbor? 0

```

#### Zadatak 6.4 Police za predmete

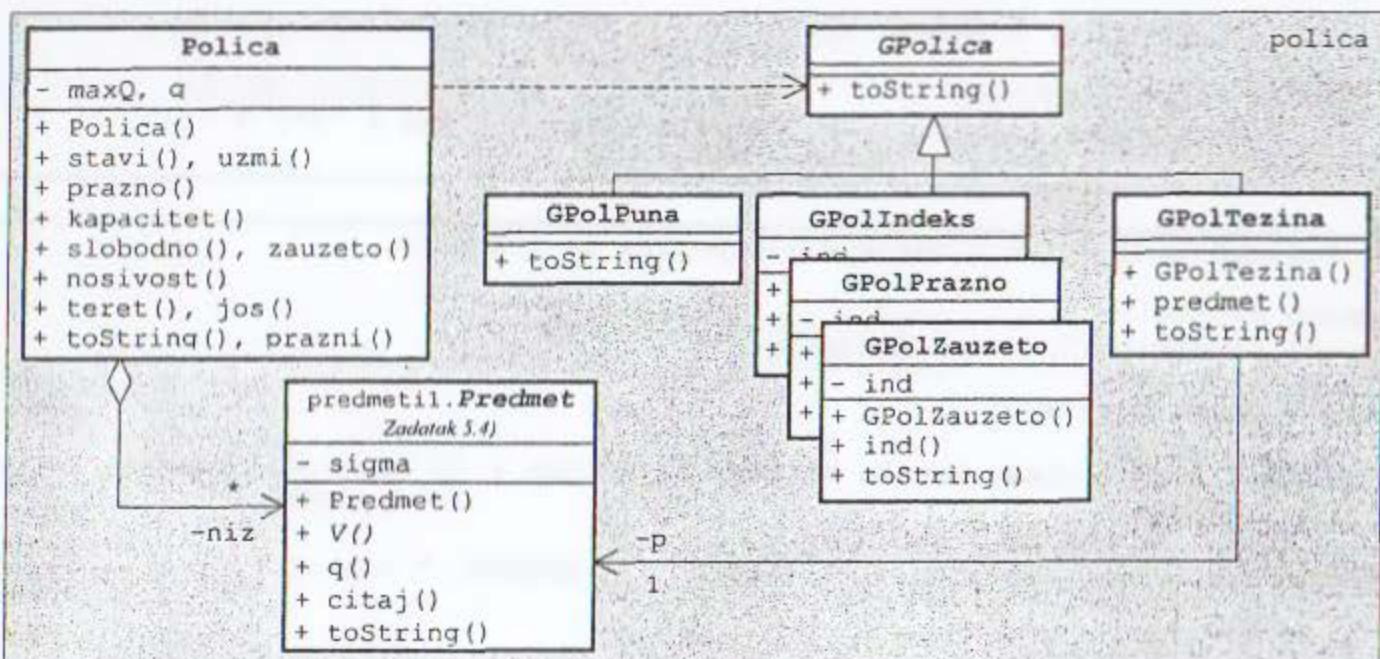
Napisati na jeziku Java klasu polica za apstraktne predmete iz zadatka 5.4. Predvideti:

- stvaranje police sa zadatim brojem mesta za stavljanje predmeta i zadatom dozvoljenom najvećom težinom svih predmeta na polici,
- stavljanje predmeta na zadato mesto i na prvo slobodno mesto na polici,
- uzimanje predmeta sa zadatog mesta na polici (predmet se uklanja sa police),
- pristup predmetu na datom mestu na polici (predmet ostaje na polici),
- ispitivanje da li je neko mesto na polici prazno,
- dohvatanje podataka o stanju police (kapacitet, broj popunjениh i praznih mesta, nosivost, ukupan teret na polici i koliko tereta može još da se doda),
- sastavljanje tekstualnog oblika police, i
- pražnjenje police.

Konfliktne situacije (nedozvoljeni indeks mesta, stavljanje na popunjeno mesto, uzimanje sa praznog mesta itd.) prijavljivati izuzecima.

Napisati na jeziku Java interaktivni program za ispitivanje klase polica.

**Rešenje:**



```

// Polica.java - Klasa polica za predmete.

package polica;
import predmeti.Predmet;

public class Polica {

    private Predmet[] niz; // Niz za smeštanje predmeta.
    private double maxQ; // Nosivost police.
    private double q = 0; // Trenutna opterećenost police.

    public Polica (int kap, double Qmax) { // Inicijalizacija.
        niz = new Predmet [kap];
        maxQ = Qmax;
    }
}
  
```

```

public Polica stavi (Predmet p, int i) // Stavljanje na dano mesto.
throws GPolIndeks, GPolZauzeto, GPolTezina {
    if (i<0 || i>=niz.length) throw new GPolIndeks (i);
    if (niz[i] != null) throw new GPolZauzeto (i);
    if (q + p.Q() > maxQ) throw new GPolTezina (p);
    niz[i] = p; q += p.Q ();
    return this;
}

public Polica stavi (Predmet p) // Stavljanje na prvo slobodno mesto.
throws GPolTezina, GPolPuna {
    if (q + p.Q() > maxQ) throw new GPolTezina (p);
    int i = 0; while (i<niz.length && niz[i]!=null) i++;
    if (i == niz.length) throw new GPolPuna ();
    niz[i] = p; q += p.Q ();
    return this;
}

public Predmet uzmi (int i) // Uzimanje sa datog mesta.
throws GPolIndeks, GPolPrazno {
    if (i<0 || i>=niz.length) throw new GPolIndeks (i);
    if (niz[i] == null) throw new GPolPrazno (i);
    Predmet p = niz[i]; niz[i] = null; q -= p.Q ();
    return p;
}

public Predmet dohvati (int i) // Dohvatanje na datom mestu.
throws GPolIndeks, GPolPrazno {
    if (i<0 || i>=niz.length) throw new GPolIndeks (i);
    if (niz[i] == null) throw new GPolPrazno (i);
    return niz[i];
}

public boolean prazno (int i) throws GPolIndeks { // Da li je mesto
    if (i<0 || i>=niz.length) throw new GPolIndeks (i); // prazno?
    return niz[i] == null;
}

public int kapacitet() { return niz.length; } // Kapacitet police.

public int slobodno () // Broj slobodnih mesta.
{
    int k = 0;
    for (int i=0; i<niz.length; i++) if (niz[i] == null) k++;
    return k;
}

public int zauzeto () // Broj zauzetih mesta.
{
    return kapacitet () - slobodno ();
}

public double nosivost () { return maxQ; } // Nosivost police.

public double teret () { return q; } // Ukupan teret na polici.

public double jos() { return maxQ - q; } // Dozvoljeni dodatni teret.

public String toString () // Tekstualni oblik.
{
    StringBuffer s = new StringBuffer ();
    for (int i=0; i<niz.length; i++)
        if (niz[i] != null)
            s.append (i).append (".").append (niz[i]).append ("\n");
    return s.toString ();
}

```

```

public Polica prazni () {                                // Pražnjenje police.
    for (int i=0; i<niz.length; niz[i++]=null);
    q = 0;
    return this;
}
}

```

// GPolica.java - Apstraktna klasa za greške pri radu s policama.

```

package polica;

public abstract class GPolica extends Exception {
    public String toString ()                      // Tekstualni oblik.
        ( return "**** GRESKA s policom: "; )
}

```

// GPolPuna.java - Klasa za greške: Polica je puna.

```

package polica;

public class GPolPuna extends GPolica {
    public String toString ()                      // Tekstualni oblik.
        ( return super.toString() + "Nema mesta na polici!"; )
}

```

// GPolIndeks.java - Klasa za greške: Nedozvoljen indeks.

```

package polica;

public class GPolIndeks extends GPolica {
    private int ind;                                // Nedozvoljeni indeks.
    public GPolIndeks (int i) { ind = i; }          // Inicijalizacija.
    public int ind () { return ind; }                // Dohvatanje indeksa.
    public String toString ()                      // Tekstualni oblik.
        ( return super.toString() + "Nedozvoljen indeks " + ind +"!"; )
}

```

// GPolTezina.java - Klasa za greške: Prekoračena je nosivost police.

```

package polica;
import predmeti.Predmet; → Zadatak 5.4

public class GPolTezina extends GPolica {
    private Predmet p;                            // Pretežak predmet.
    public GPolTezina (Predmet pp) { p = pp; }    // Inicijalizacija.
    public Predmet predmet () { return p; }        // Dohvatanje predmeta.
    public String toString ()                      // Tekstualni oblik.
        ( return super.toString() + "Prevelika tezina " + p.Q() +"!"; )
}

```

```
// GPolPrazno.java - Klasa za greške: Mesto je prazno.

package polica;

public class GPolPrazno extends GPolica {
    private int ind;                                // Indeks praznog mesta.
    public GPolPrazno (int i) { ind = i; }           // Inicijalizacija.
    public int ind () { return ind; }                // Dohvatanje indeksa.
    public String toString ()                      // Tekstualni oblik.
        { return super.toString() + "Prazno mesto " + ind + "!"; }
}
```

```
// GPolZauzeto.java - Klasa za greške: Mesto je zauzeto.

package polica;

public class GPolZauzeto extends GPolica {
    private int ind;                                // Indeks popunjeno mesta.
    public GPolZauzeto (int i) { ind = i; }           // Inicijalizacija.
    public int ind () { return ind; }                // Dohvatanje indeksa.
    public String toString ()                      // Tekstualni oblik.
        { return super.toString() + "Zauzeto mesto " + ind + "!"; }
}
```

```
// PolicaT.java - Ispitivanje klase polica za predmete.

import polica.*;
import predmeti.*; → Zadatak 5.4

public class PolicaT {

    private static class Greska extends Exception { // Klasa za lokalne
        String por;                                // greške.
        Greska (String p) { por = p; }
        public String toString () { return "**** GRESKA: " + por; }
    }

    private static Predmet citaj () throws Greska { // Čitanje predmeta.
        System.out.print ("Vrsta (S-sfera, K-kvadar)? ");
        Predmet p = null;
        switch (Citaj.Char ()) {
            case 's': case 'S': p = new Sfera (); break;
            case 'k': case 'K': p = new Kvadar (); break;
            default: throw new Greska ("Nepoznata vrsta predmeta!");
        }
        System.out.print ("Spec. tez. i dimenzije? ");
        p.citaj (); return p;
    }

    public static void main (String[] varg) {      // Glavna funkcija.
        Polica p = new Polica (Integer.parseInt (varg[0]),
                               Double.parseDouble (varg[1]));
    }
}
```

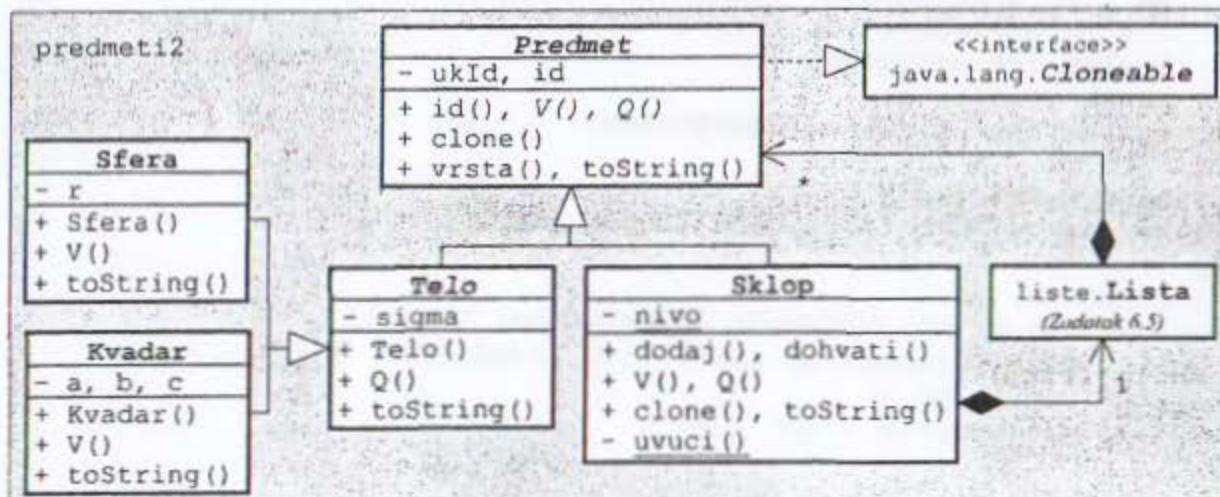
**Zadatak 6.6 Predmeti koji mogu da se kopiraju, tela, sfere, kvadri i sklopovi**

Napisati na jeziku Java sledeće tipove:

- Apstraktan *predmet* ima svoj jedinstven, automatski generisan, identifikacioni broj. Može da se dohvati identifikacioni broj predmeta, da se odredi zapremina i težina predmeta, da se stvori kopija predmeta, da se dohvati ime vrste predmeta i da se sastavi tekstualni oblik predmeta.
- Apstraktno *telo* je homogeni predmet određene specifične težine.
- *Sfera* i *kvadar* su tela zadata poluprečnikom, odnosno dužinama ivica.
- *Sklop* je predmet koji može da sadrži proizvoljan broj predmeta proizvoljne vrste. Stvara se prazan, posle čega predmeti se dodaju jedan po jedan. Može da se dohvati predmet u sklopu sa određenim rednim brojem. Za skladištenje predmeta koristiti klasu listi iz zadatka 6.5.

Napisati na jeziku Java program za ispitivanje prethodnih tipova.

**Rešenje:**



```

// Predmet.java - Apstraktna klasa predmeta.

package predmeti2;

public abstract class Predmet implements Cloneable {

    private static int ukId;      // Poslednje korišćeni identifikator.
    private int id = ++ukId;      // Identifikator predmeta.

    public final int id () { return id; } // Dohvatanje identifikatora.

    public abstract double V ();           // Zapremina.

    public abstract double Q ();           // Težina.

    public Object clone () {               // Stvaranje kopije.
        try {
            Predmet p = (Predmet)super.clone ();
            p.id = ++ukId;
            return p;
        } catch (CloneNotSupportedException g) { return null; }
    }
}
  
```

```

public final String vrsta () {           // Dohvatanje vrste predmeta.
    String s = getClass ().getName ();
    return s.substring (s.lastIndexOf ('.') + 1);
}

public String toString ()               // Tekstualni oblik.
{ return id + " " + vrsta (); }
}

```

// Telo.java - Apstraktna klasa tela.

```

package predmeti2;

public abstract class Telo extends Predmet {

    protected double sigma;                  // Specifična težina.

    public Telo (double s) { sigma = s; }      // Inicijalizacija.

    public final double Q () { return sigma * V (); } // Težina.
                                                    // Tekstualni oblik.
    public String toString () { return super.toString () + " " + sigma; }
}

```

// Sfera.java - Klasa sfera.

```

package predmeti2;

public class Sfera extends Telo {

    private double r;                      // Poluprečnik.

    public Sfera (double sigma, double r)     // Inicijalizacija.
        { super (sigma); this.r = r; }

    public final double V () { return 4./3 * r*r*r * Math.PI; } // Zapremina.

    public String toString ()                // Tekstualni oblik.
        { return super.toString () + " " + r; }
}

```

// Kvadar.java - Klasa kvadara.

```

package predmeti2;

public class Kvadar extends Telo {

    private double a, b, c;                 // Ivice.
                                                // Inicijalizacija.

    public Kvadar (double sigma, double a, double b, double c)
        { super (sigma); this.a = a; this.b = b; this.c = c; }

    public final double V () { return a * b * c; } // Zapremina.

    public String toString ()                // Tekstualni oblik.
        { return super.toString () + " " + a + " " + b + " " + c; }
}

```

```

// Sklop.java - Klasa sklopova predmeta.

package predmeti2;
import liste.*; → Zadatak 6.5

public class Sklop extends Predmet {

    private Lista lst = new Lista ();           // Lista sadržanih predmeta.

    public Sklop dodaj (Predmet p)             // Dodavanje predmeta.
        { lst.dodaj (p); return this; }

    public Predmet dohvati (int ind) {          // Dohvatanje predmeta.
        lst.naPrvi ();
        for (int i=1; i<ind; i++) lst.naSledeci ();
        try { return (Predmet)lst.dohvatiTek (); }
        catch (GLstNemaTek g) { return null; }
    }

    public final double V () {                  // Zapremina.
        double V = 0;
        for (lst.naPrvi(); lst.imaTek(); lst.naSledeci())
            try { V += ((Predmet)lst.dohvatiTek ()).V (); }
            catch (GLstNemaTek g) {}
        return V;
    }

    public final double Q () {                  // Težina.
        double Q = 0;
        for (lst.naPrvi(); lst.imaTek(); lst.naSledeci())
            try { Q += ((Predmet)lst.dohvatiTek ()).Q (); }
            catch (GLstNemaTek g) {}
        return Q;
    }

    public Object clone () {                   // Stvaranje kopije.
        Sklop s = (Sklop)super.clone ();
        s.lst = new Lista ();
        for (lst.naPrvi(); lst.imaTek(); lst.naSledeci())
            try {
                Predmet p = (Predmet)lst.dohvatiTek ();
                s.dodaj ((Predmet)p.clone ());
            } catch (GLstNemaTek g) {}
        return s;
    }

    private static int nivo = 0;               // Nivo uklapanja predmeta.

    private static void uvuci (StringBuffer s) { // Uvlačenje uklopljenih
        for (int i=0; i<nivo; i++) s.append ("  "); // predmeta u tekstu-
    }                                         // alnom obliku.
}

```

```

public String toString () { // Tekstualni oblik.
    StringBuffer s = new StringBuffer ();
    s.append (super.toString ()).append ("\n");
    nivo++;
    for (lst.naPrvi (); lst.imaTek (); lst.naSledeci ()) {
        try {
            Predmet p = (Predmet)lst.dohvatiTek ();
            uvuci (s); s.append (p).append ('\n');
        } catch (GLstNemaTek g) {}
    }
    nivo--;
    uvuci (s); return s.append (')').toString ();
}
}

```

// Predmeti2T.java - Ispitivanje klase predmeta.

```

import predmeti2.*;

public class Predmeti2T {
    public static void main (String[] vpar) {
        Sklop s = new Sklop ();
        s.dodaj (new Sfera (1, 2))
        .dodaj (new Kvadar (3, 4, 5, 6))
        .dodaj (new Sklop ().dodaj (new Kvadar (2, 5, 3, 4))
                 .dodaj (new Sfera (3, 1)))
        .dodaj (new Sfera (4, 2));
        System.out.println (s + "\nV=" + s.V() + " Q=" + s.Q());
        System.out.println ();
        s = (Sklop)s.clone ();
        System.out.println (s + "\nV=" + s.V() + " Q=" + s.Q());
    }
}

```

```

% java Predmeti2T
1 Sklop {
2 Sfera 1.0 2.0
3 Kvadar 3.0 4.0 5.0 6.0
4 Sklop {
5 Kvadar 2.0 5.0 3.0 4.0
6 Sfera 3.0 1.0
}
7 Sfera 4.0 2.0
}
V=251.20943348136862 Q=660.1179788058148

8 Sklop {
9 Sfera 1.0 2.0
10 Kvadar 3.0 4.0 5.0 6.0
11 Sklop {
12 Kvadar 2.0 5.0 3.0 4.0
13 Sfera 3.0 1.0
}
14 Sfera 4.0 2.0
}
V=251.20943348136862 Q=660.1179788058148

```

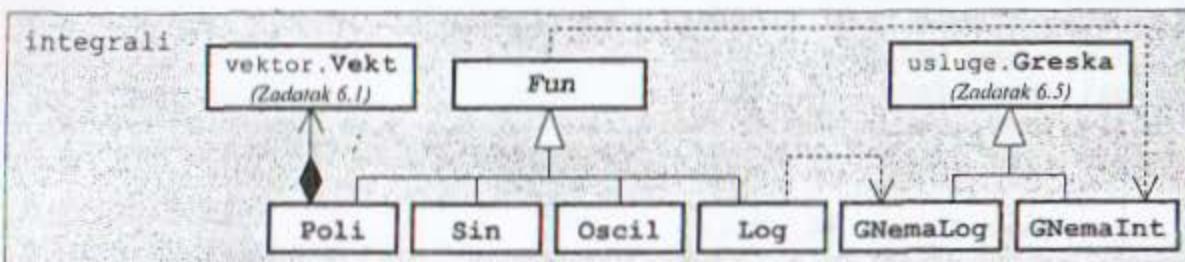
### Zadatak 6.7 Funkcije sa izračunavanjem određenog integrala

Napisati na jeziku Java sledeće tipove:

- Za apstraktnu *funkciju* s jednim realnim argumentom  $x$  može da se izračuna vrednost za dato  $x$ , vrednost neodređenog integrala za dato  $x$  (za integracionu konstantu se uzima 0) i vrednost određenog integrala za dati interval  $a \leq x \leq b$ . Ako ne postoji algebarski izraz za neodređeni integral, pri pokušaju njegovog izračunavanja, prijavljuje se izuzetak. Određeni integral se računa na osnovu neodređenog integrala, ako postoji algebarski izraz za to, odnosno po približnoj formuli, ukoliko ne postoji takav izraz.
- $\sin x$ ,  $e^{-0.1x}$ ,  $\sin x$ ,  $\log x$ , i polinom  $p(x)$  su funkcije.
- Konfliktne situacije prijavljivati objektima tipa specijalnih klasa kao potklase klase Greska iz zadatka 6.5.

Napisati na jeziku Java program za ispitivanje prethodnih tipova.

Rešenje:



```

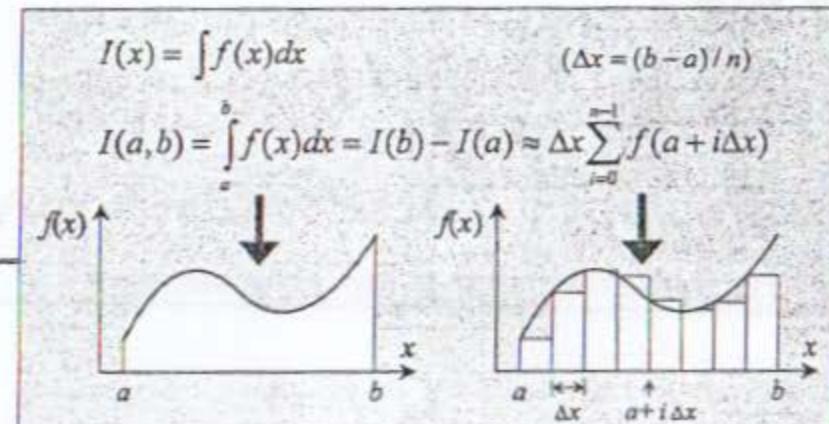
// Fun.java - Apstraktna klasa funkcija jednog realnog argumenta.

package integrali;
import usluge.Greska;

public abstract class Fun {
    // Vrednost funkcije.
    public abstract double f (double x) throws Greska;

    public double I (double x) throws GNemaInt // Neodređeni integral.
    { throw new GNemaInt(); }

    public final double I (double a, double b) throws Greska {
        final int N = 30;
        try {
            return I (b) - I (a);
        } catch (GNemaInt g) {
            double dx = (b-a)/N,
            s = 0;
            for (int i=0; i<N; i++)
                s += f(a+i*dx);
            return s * dx;
        }
    }
}
  
```



```
// GNemaInt.java - Klasa za greške: Funkcija nema integral.

package integrali;
public class GNemaInt extends usluge.Greska {
    public GNemaInt () { super ("Funkcija nema integral!"); }
}
```

Zadatak 6.5

```
// Sin.java - Klasa za funkciju sinus.

package integrali;
public class Sin extends Fun {
    public double f (double x) { return Math.sin (x); } // Funkcija.
    public double I (double x) { return - Math.cos (x); } // Integral.
}
```

$$\int \sin x dx = -\cos x$$

```
// Oscil.h - Klasa za prigušene oscilacije.

package integrali;
public class Oscil extends Fun {
    public double f (double x) { return Math.exp (-0.1 * x) * Math.sin (x); } // Funkcija.
}
```

$$\int e^{-0.1x} \sin x dx = ???$$

```
// GNemaLog.java - Klasa za greške: Nedozvoljeni argument za logaritam.

package integrali;
public class GNemaLog extends usluge.Greska {
    private double x; // Nedozvoljena vrednost.
    public GNemaLog () { super ("Logaritam negativnog broja"); } // Inicijalizacija:
    public GNemaLog (double x) { this.x = x; } // - bez navodenja
                                                // nedozvoljene vrednosti,
    public double x () { return imaPoruke() ? 1 : x; } // - navodenjem
                                                       // nedozvoljene vrednosti.
    public String toString () { // Uzimanje nedozvoljene
        return imaPoruke() ? super.toString()
            : "!!! Ne postoji logaritam od " + x + "!";
    }
}
```

Zadatak 6.5

```
// Log.java - Klasa za funkciju logaritam.

package integrali;

public class Log extends Fun {

    public double f (double x) throws GNemaLog { // Funkcija.
        if (x <= 0) throw new GNemaLog (x);
        return Math.log (x);
    }
}
```

$$\int \log x dx = ???$$

```
// Poli.java - Klasa polinoma.

package integrali;
import vektor.*; → Zadatak 6.1

public class Poli extends Fun {

    private Vekt a; // Vektor koeficijenata.

    public Poli (int n) throws GVekt // Inicijalizacija.
        ( a = new Vekt (0, n); )

    public Poli postavi (int i, double b) throws GVekt // Postavljanje
        ( a.postavi (i, b); return this; ) // koeficijenta.

    public double dohvati (int i) throws GVekt // Dohvatanje koeficijenta.
        ( return a.dohvati (i); )

    public int red () { return a.maxInd (); } // Red polinoma.

    public double f (double x) { // Vrednost polinoma.
        double s = 0;
        try {
            for (int i=a.maxInd(); i>=0; s = s*x + a.dohvati(i--));
        } catch (GVekt g) {}
        return s;
    }

    public double I (double x) { // Vrednost integrala.
        double s = 0; // integrala.
        try {
            for (int i=a.maxInd(); i>=0; s = s*x + a.dohvati(i) / (i-- + 1));
        } catch (GVekt g) {}
        return s * x;
    }
}
```

$$p(x) = \sum_{i=0}^n a_i x^i$$

$$\int p(x) dx = \sum_{i=0}^n a_i \frac{x^{i+1}}{i+1} = x \sum_{i=0}^n \frac{a_i}{i+1} x^i$$

```

// IntegraliT.java - Ispitivanje računanja integrala.

import integrali.*;

public class IntegraliT {
    public static void main (String[] varg) {
        radi: while (true) {
            Fun fun = null;
            try {
                System.out.print ("\nFunkcija (S, O, P, L, .)? ");
                switch (Citaj.Char ()) {
                    case 's': case 'S': fun = new Sin (); break;
                    case 'o': case 'O': fun = new Oscil (); break;
                    case 'p': case 'P': {
                        System.out.print ("Red polinoma? "); int n = Citaj.Int ();
                        Poli p = new Poli (n);
                        while (true) {
                            System.out.print ("Indeks i vrednost koeficijenta? ");
                            int i = Citaj.Int ();
                            if (i<0) break;
                            p.postavi(i, Citaj.Double ());
                        }
                        fun = p;
                        break; → Zadatak 6.1
                    }
                    case 'l': case 'L': fun = new Log (); break;
                    case '.': break radi;
                    default: throw new usluge.Greska ("Nedozvoljen izbor!");
                }
                System.out.print ("Interval? "); → Zadatak 6.5
                System.out.println ("Integral= " +
                    fun.I (Citaj.Double(), Citaj.Double()));
            } catch (usluge.Greska g) { System.out.println (g);
            } catch (vektor.GVekt g) { System.out.println (g);
            }
        }
    }
}

```

% java IntegraliT

Funkcija (S, O, P, L, .)? s  
Interval? 0 3.14159  
Integral= 1.9999999999964793

Funkcija (S, O, P, L, .)? l  
Interval? 1 10  
Integral= 13.673735099805986

Funkcija (S, O, P, L, .)? p  
Red polinoma? 3  
Indeks i vrednost koeficijenta? 3 -1  
Indeks i vrednost koeficijenta? 2 2  
Indeks i vrednost koeficijenta? 1 -3  
Indeks i vrednost koeficijenta? 0 4  
Indeks i vrednost koeficijenta? -1

Interval? -5 5  
Integral= 206.66666666666657

Funkcija (S, O, P, L, .)? k  
\*\*\* Nedozvoljen izbor!

Funkcija (S, O, P, L, .)? l  
Interval? -1 1  
\*\*\* Ne postoji logaritam od -1.0!

Funkcija (S, O, P, L, .)? p  
Red polinoma? 3  
Indeks i vrednost koeficijenta? 6 1  
\*\*\* Indeks je izvan opsega!

Funkcija (S, O, P, L, .)? .

**Zadatak 6.8 Funkcije za koje mogu da se stvaraju izvodi, monomi, eksponencijalne funkcije i zbirovi funkcija**

apisati na jeziku Java sledeće klase:

- Apstraktnoj *funkciji* može da se izračuna vrednost za datu vrednost realne nezavisno promenljive i može da se stvari funkcija koja predstavlja njen prvi izvod. Tekstualni oblik predstavlja algebarski oblik funkcije.
- *Monom* je funkcija oblika  $ax^b$  (podrazumevano  $x$ ), a njen izvod je  $abx^{b-1}$ . Tekstualni oblik je  $a*x^b$ , gde su  $a$  i  $b$  vrednosti parametara monoma.
- *Eksponencijalna funkcija* je funkcija oblika  $a e^{bx}$  (podrazumevano  $e^x$ ), a njen izvod je  $abe^{bx}$ . Tekstualni oblik je  $a*\exp(b*x)$ , gde su  $a$  i  $b$  vrednosti parametara eksponencijalne funkcije.
- *Zbir* funkcija je funkcija koja može da sadrži zadati broj funkcija (podrazumevano 2). Stvara se prazan, posle čega funkcije mogu da se dodaju jedna po jedna. Pokušaj stavljanja funkcije u pun zbir funkcija se prijavljuje izuzetkom odgovarajućeg tipa. Vrednost zbiru funkcija je zbir vrednosti sadržanih funkcija. Izvod zbiru funkcija je zbir izvoda sadržanih funkcija. Tekstualni oblik zbiru funkcija je  $(fun) + \dots + (fun)$ , gde je *fun* tekstualni oblik jedine sadržane funkcije.

apisati na jeziku Java program koji napravi jedan zbir funkcija kapaciteta koji se zadaje kao parametar glavne funkcije, dodaje nekoliko prostih funkcija (monoma ili eksponencijalne funkcije) čitajući potrebne podatke s glavnog ulaza, ispiše algebarski oblik dobijenog zbiru funkcija i njegovog izvoda na glavnom izlazu i posle tabelira vrednosti zbiru funkcija i njegovog izvoda za svako  $x_{\min} \leq x \leq x_{\max}$  sa korakom  $\Delta x$ . Parametri tabeliranja zadaju se kao parametri glavne funkcije.

Rešenje:

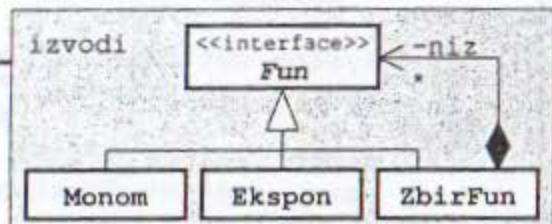
```
// Fun.java - Interfejs za funkcije.

package izvodi;

public interface Fun {
    double f (double x);                                // Vrednost funkcije.

    Fun izv ();                                         // Izvod funkcije.

    String toString ();                                 // Tekstualni oblik.
}
```



```
// Monom.java - Klasa monoma.

package izvodi;

public class Monom implements Fun {
    private double a, b;                                // Parametri.

    public Monom (double aa, double bb)                // Inicijalizacija.
    { a = aa; b = bb; }

    public Monom () { a = b = 1; }

    public double f (double x)                         // Računanje vrednosti.
    { return a * Math.pow (x, b); }
```

```

public Fun izv ()                                // Stvaranje izvoda.
{ return new Monom (a*b, b-1); }

public String toString ()                         // Tekstualni oblik.
{ return a + "*x^" + b; }
}

```

```

// Ekspon.java - Klasa eksponencijalnih funkcija.

package izvodi;

public class Ekspon implements Fun {

    private double a, b;                          // Parametri.

    public Ekspon (double aa, double bb)          // Inicijalizacija.
    { a = aa; b = bb; }

    public Ekspon () { a = b = 1; }

    public double f (double x)                    // Računanje vrednosti.
    { return a * Math.exp (b * x); }

    public Fun izv ()                            // Stvaranje izvoda.
    { return new Ekspon (a*b, b); }

    public String toString ()                   // Tekstualni oblik.
    { return a + "*exp(" + b + "*x)"; }
}

```

```

// GZbirFunPun.java - Klasa za greške: Zbir funkcija je pun.

package izvodi;

public class GZbirFunPun extends Exception {

    public String toString ()                  // Tekstualni oblik.
    { return "*** Zbir funkcija je pun!"; }
}

```

```

// ZbirFun.java - Klasa zbirova funkcija.

package izvodi;

public class ZbirFun implements Fun {

    private Fun[] niz;                           // Niz funkcija.
    private int duz;                            // Broj popunjениh mesta.

    public ZbirFun (int n) { niz = new Fun [n]; } // Inicijalizacija.

    public ZbirFun () { this (2); }
}

```

```

public ZbirFun dodaj (Fun f) throws GZbirFunPun { // Dodavanje funkcije.
    if (duz == niz.length) throw new GZbirFunPun ();
    niz[duz++] = f;
    return this;
}

public double f (double x) {                                // Računanje vrednosti.
    double s = 0;
    for (int i=0; i<duz; s+=niz[i++].f(x));
    return s;
}

public Fun izv () {                                       // Stvaranje izvoda.
    ZbirFun z = new ZbirFun (niz.length);
    try {
        for (int i=0; i<duz; z.dodaj (niz[i++].izv()));
    } catch (GZbirFunPun g) {}
    return z;
}

public String toString () {                             // Tekstualni oblik.
    StringBuffer s = new StringBuffer();
    for (int i=0; i<duz; i++) {
        if (i > 0) s.append ('+');
        s.append ('(').append (niz[i]).append ());
    }
    return s.toString ();
}
}

```

// IzvodiT.java - Ispitivanje stvaranja izvoda.

```

import izvodi.*;
import usluge.Greska; → Zadatak 6.5

public class IzvodiT {
    public static void main (String[] varg) {
        ZbirFun zbir = new ZbirFun (Integer.parseInt (varg[0]));
        try {
            radi: while (true) {
                double a, b; char vrs;
                System.out.print ("Vrsta (M, E, *)? ");
                switch (Citaj.Char ()) {
                    case 'm': case 'M':
                        System.out.print ("a,b? ");
                        zbir.dodaj (new Monom (Citaj.Double(), Citaj.Double()));
                        break;
                    case 'e': case 'E':
                        System.out.print ("a,b? ");
                        zbir.dodaj (new Ekspon (Citaj.Double(), Citaj.Double()));
                        break;
                    case '*': break radi;
                    default:
                        throw new Greska ("Nedozvoljeni izbor!");
                }
            }
            Fun izvod = zbir.izv ();
            System.out.println ("Fun= " + zbir );
            System.out.println ("Izv= " + izvod);
        }
    }
}

```

```

        double xmin = Double.parseDouble (varg[1]),
               xmax = Double.parseDouble (varg[2]),
               dx   = Double.parseDouble (varg[3]);
    for (double x=xmin; x<=xmax; x+=dx)
        System.out.println (x + "\t" + zbir.f(x) + "\t" + izvod.f(x));
    } catch (Exception g) { System.out.println (g); }
}
}

```

```

% java IzvodiT 5 -2 2 .5
Vrsta (M, E, *)? m
a,b? 4 2
Vrsta (M, E, *)? e
a,b? .5 -2
Vrsta (M, E, *)? m
a,b? 3 4
Vrsta (M, E, *)? *
Fun= (4.0*x^2.0)+(0.5*exp(-2.0*x))+(3.0*x^4.0)
Izv= (8.0*x^1.0)+(-1.0*exp(-2.0*x))+(12.0*x^3.0)
-2.0 91.29907501657212 -166.59815003314424
-1.5 34.23026846159384 -72.58553692318768
-1.0 10.694528049465326 -27.389056098930652
-0.5 2.5466409142295228 -8.218281828459045
0.0 0.5 -1.0
0.5 1.3714397205857212 5.132120558828557
1.0 7.067667641618306 19.864664716763386
1.5 24.21239353418393 52.45021293163214
2.0 64.00915781944437 111.98168436111126

```

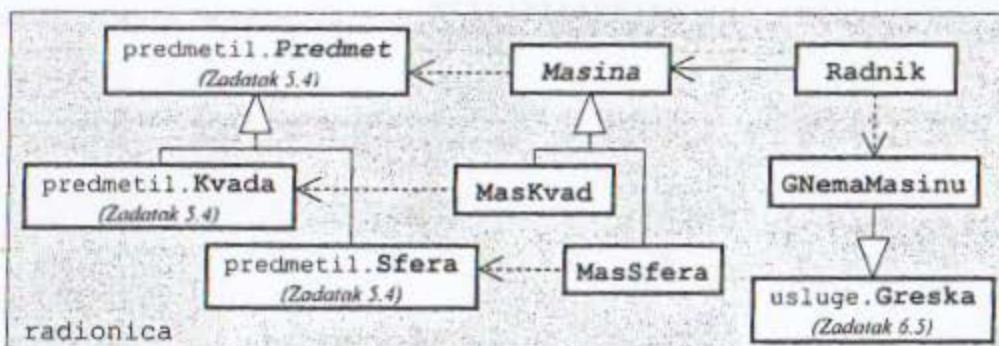
### Zadatak 6.9 Mašine, mašine za sfere, mašine za kvadre i radnici

Koristeći tipove predmeta iz zadatka 5.4 napisati na jeziku Java sledeće tipove:

- Apstraktna **mašina** može da proizvodi apstraktni predmet od materijala zadate specifične težine i zna se koliko je predmeta od datog materijala proizvela ta mašina. Može da se dohvati oznaka vrste proizvoda koje data mašina proizvodi i broj proizvedenih predmeta. Može da se promeni materijal od čega se prave predmeti, i da se dohvati specifična težina korišćenog materijala.
- Mašina za kvadre** je mašina koja može da proizvodi kvadre zadatih dimenzija. Parametri maštine (dimenzije kvadra) ne mogu da se promene, ali mogu da se dohvate.
- Mašina za sfere** je mašina koja može da proizvodi sfere zadatog poluprečnika. Parametar maštine (poluprečnik sfere) ne može da se promeni, ali može da se dohvati.
- Radnik** ima ime i jedinstven, automatski generisan identifikacioni broj. Izrađuje predmete određene vrste pomoću odgovarajuće maštine. Mašina na kojoj radnik radi može da se promeni. Zna se koliko je predmeta radnik izradio na mašini na kojoj trenutno radi. Radnik može da ne bude raspoređen ni na jednu mašinu. U tom slučaju pokušaj izrade predmeta ili dohvatanja broja izrađenih predmeta se prijavljuje izuzetkom odgovarajućeg tipa. Tekstualni oblik radnika sadrži ime i identifikacioni broj radnika. U slučaju da je radnik raspoređen na neku mašinu, dodaje se i vrsta predmeta koje trenutno izraduje i broj proizvoda koje je izradio na toj mašini od početka rada na njoj.

Napisati na jeziku Java program za ispitivanje prethodnih tipova.

Rešenje:



```

// Masina.java - Apstraktna klasa mašina.

package radionica;
import predmetil.Predmet;

public abstract class Masina {

    protected double spTez;           // Specifična težina pravljenih predmeta.
    protected int br;                 // Broj napravljenih predmeta.

    protected Masina (double s) { spTez = s; } // Inicijalizacija.

    public abstract char vrsta ();      // Vrsta pravljenih predmeta.

    public int broj () { return br; }       // Broj napravljenih predmeta.

    public double uzmiSpTez () { return spTez; } // Dohvatanje spec. težine.
  
```

```

public void postaviSpTez (double s)           // Postavljanje spec. težine.
{ spTez = s; br = 0; }

public abstract Predmet napravi ();          // Pravljenje predmeta.
}

```

// MasSfera.java - Klasa mašina za sfere.

```

package radionica;
import predmetil.*; ← Iz zadatka 5.4

public class MasSfera extends Masina {

    private double r;                         // Poluprečnik pravljениh sfera.

    public MasSfera (double ss, double rr) // Inicijalizacija.
    { super (ss); r = rr; }

    double dohvatiR () { return r; }          // Dohvatanje poluprečnika.

    public char vrsta () { return Sfera.VR; } // Vrsta pravljениh proizv.

    public Predmet napravi ()                // Pravljenje sfere.
    { br++; return new Sfera (spTez, r); }
}

```

// MasKvad.java - Klasa mašina za kvadre.

```

package radionica;
import predmetil.*; → Zadatak 5.4

public class MasKvad extends Masina {

    private double a, b, c;                  // Dimenzije pravljениh kvadara.

    public MasKvad (double ss, double aa, double bb, double cc) // Inicijali-
    { super (ss); a = aa; b = bb; c = cc; }           // zacija.

    double dohvatiA () { return a; }             // Dohvatanje dimenzija
    double dohvatiB () { return b; }             // pravljениh kvadara.
    double dohvatiC () { return c; }

    public char vrsta () { return Kvadar.VR; } // Vrsta pravljениh predmeta.

    public Predmet napravi ()                // Pravljenje kvadra.
    { br++; return new Kvadar (spTez, a, b, c); }
}

```

```
// GNemaMas.java - Klasa za greške: Radnik nema mašinu.

package radionica;
import usluge.Greska; → Zadatak 6.5

public class GNemaMas extends Greska {

    String ime; int id; // Ime i identifikator problematičnog radnika.

    public GNemaMas (String iime, int iid) // Inicijalizacija.
        { ime = iime; id = iid; }

    public String ime () { return ime; } // Dohvatanje atributa.
    public int id () { return id; }

    public String toString () { // Tekstualni oblik.
        return "**** Radniku " + ime + ", id " + id +
            ", nije dodeljena masina!";
    }
}
```

```
// Radnik.java - Klasa radnika.

package radionica;
import predmeti.Predmet; → Zadatak 5.4

public class Radnik {

    private static int ukId; // Poslednje korišćeni identifikator.
    private int id = ++ukId; // Identifikator radnika.
    private String ime; // Ime radnika.
    private Masina mas; // Mašina koju radnik koristi.
    private int br; // Broj napravljenih proizvoda na mašini.

    public Radnik (String iime) { ime = iime; } // Inicijalizacija.

    public int dohvatiId () { return id; } // Dohvatanje identifikatora
    public String dohvatiIme () { return ime; } // i imena.

    public Radnik rasporedi (Masina m) // Raspoređivanje na mašinu.
        { mas = m; br = 0; return this; }

    public Masina dohvatiMas () { return mas; } // Dohvatanje masine.

    public int broj () throws GNemaMas { // Broj napravljenih
        if (mas == null) throw new GNemaMas (ime, id); // predmeta.
        return br;
    }

    public Predmet napravi () throws GNemaMas { // Pravljenje predmeta.
        if (mas == null) throw new GNemaMas (ime, id);
        br++;
        return mas.napravi ();
    }
}
```

```

public String toString () {                                // Tekstualni oblik.
    String s = ime + "/" + id;
    if (mas != null) s += "(" + mas.vrsta() + "," + br + ")";
    return s;
}
}

```

// RadionicaT.java - Ispitivanje klase mašina i radnika.

```

import radionica.*;

public class RadionicaT {
    public static void main (String[] varg) {
        try {
            MasKvad m1 = new MasKvad (0.5, 1, 2, 3);
            MasSfera m2 = new MasSfera (0.7, 2);
            Radnik marko = new Radnik ("Marko");
            System.out.println (marko);
            marko.rasporedi (m1);
            for (int i=0; i<10; i++) marko.napravi ();
            System.out.println (marko);
            marko.rasporedi (m2);
            for (int i=0; i<5; i++) marko.napravi ();
            System.out.println (marko);
            marko.rasporedi (null);
            for (int i=0; i<12; i++) marko.napravi ();
            System.out.println (marko);
        } catch (GNemaMas g) {
            System.out.println (g);
        }
    }
}

```

```

% java RadionicaT
Marko/1
Marko/1(K,10)
Marko/1(S,5)
*** Radniku Marko, id 1, nije dodeljena masina!

```

**Zadatak 6.10 Časovnici, pokretni radnici, kupci, prodavci i radnje**

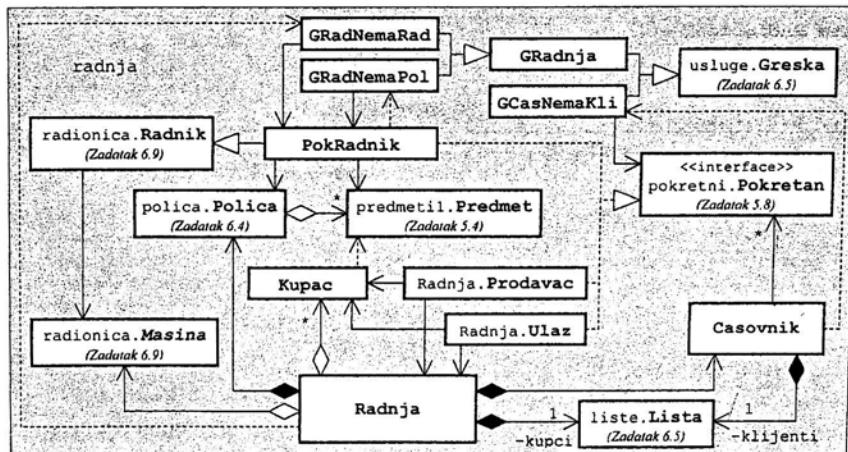
Napisati na jeziku Java sledeće tipove za simuliranje rada radnje za proizvodnju i prodaju predmeta:

- **Časovnik** služi za merenje vremena. Pokretni objekti (iz zadatka 5.8) mogu da se registruju kao klijenti kod časovnika radi obaveštavanja o protoku vremena. Kada se časovnik pokrene svoje klijente obaveštava o protoku vremena zadatim korakom do zadatog ukupnog vremena. Klijenti koji prestaju sa radom mogu da se odjave od časovnika.
  - **Pokretan radnik** je radnik (iz zadatka 6.9) koji za proizvodnju predmeta (iz zadatka 5.4) troši promenljivo vreme do zadate gornje granice. Napravljene predmete stavlja na određenu policu (iz zadatka 6.4).
  - **Kupac** ima jedinstven, automatski generisan, identifikacioni broj i kupuje predmet zadate vrste. Tekstualni oblik kupca je *Kid(vr)*, gde su: *id* – identifikacioni broj kupca, a *vr* – oznaka vrste predmeta koji kupuje.
  - **Ulas** u radnju je pokretan objekat kroz koji kupci ulaze u promenljivim vremenskim intervalima sa zadatom najvećom dužinom intervala. Kupci staju u red da bi sačekali da budu posluženi.
  - **Prodavac** je pokretan objekat koji poslužuje kupce. Posluživanje traje promenljivo vreme sa zadatim najdužim trajanjem. Kupcu se daje prvi predmet željene vrste sa police. Ako takvog predmeta nema kupac odlazi praznih ruku.
  - **Radnja** sadrži jednu policu zadate nosivosti i broja mesta za odlaganje predmeta, može da sadrži najviše 10 mašina (iz zadatka 6.9) za proizvodnju predmeta. Radnja ima jedan ulaz i jednog prodavca. Stvara se prazna posle čega mogu da se unose mašine u radnju i da se zaposle radnici koji se odmah raspoređuju na neku od mašina. Radnici mogu i da se otpuste iz radnje.

Konfliktne situacije prijavljivati odgovarajućim izuzecima.

Napisati na jeziku *Java* program za ispitivanje prethodnih klasa.

**Rešenje:**



```
// Casovnik.java - Klasa časovnika.

package radnja;
import pokretni.Pokretan; → Zadatak 5.8
import liste.*; → Zadatak 6.5

public class Casovnik {

    private Lista kiljenti = new Lista () // Lista klijenata.

    public Casovnik dodaj (Pokretan k) // Dodavanje klijenta listi.
        { kiljenti.dodaj (k); return this; } // Izbacivanje klijenta iz liste.

    public Casovnik izbaci (Pokretan k) throws GCasNemaKli {
        try { kiljenti.izbaci (k); }
        catch (GLstNemaObj g) { throw new GCasNemaKli (k); }
        return this;
    } // Parametri časovnika:
    private double dt; // - korak časovnika,
    private double t; // - trenutno vreme,
    private double tMax; // - trajanje rada.

    public double dt () { return dt; } // Dohvatanje parametara časovnika.
    public double t () { return t; }
    public double tMax () { return tMax; }

    public void radi (double dt, double tMax) { // Rad časovnika.
        this.dt = dt; this.tMax = tMax;
        for (t=0; t<=tMax; t+=dt) {
            System.out.println ("Vreme: " + t);
            for (kiljenti.naPrvi(); kiljenti.imaTek(); kiljenti.naSledeci())
                try { ((Pokretan)kiljenti.dohvatiTek()).proteklo (dt); }
                catch (GLstNemaTek g) {}
        }
    }
}
```

```
// GCasNemaKli.java - Klasa za greške: Nema traženog klijenta časovnika.

package radnja;
import pokretni.Pokretan;

public class GCasNemaKli extends usluge.Greska {

    private Pokretan klijent; // Problematični klijent.

    public GCasNemaKli (Pokretan k) // Inicijalizacija.
        { super ("Ne postoji klijent [" + k + "]"); klijent = k; }

    public Pokretan klijent () { return klijent; } // Dohvatanje klijenta.
}
```

```
// PokRadnik.java - Klasa pokretnih radnika.

package radnja;
import radionica.*;
import pokretni.Pokretan;
import polica.*;
import predmetil.*;

public class PokRadnik extends Radnik implements Pokretan {

    private Polica pol;           // Odredišna polica.
    private double maxT;          // Najduže vreme proizvodnje predmeta.
    private double t;              // Preostalo vreme do narednog događaja.
    private Predmet pred;         // Sledeći proizvedeni predmet.

    public PokRadnik (String ime, double tMax, Polica p) // Inicijalizacija.
        { super (ime); pol = p; maxT = tMax; }
    public PokRadnik (String ime, double tMax) { this (ime, tMax, null); }

    public PokRadnik polica (Polica p)                  // Pridruživanje police.
        { pol = p; return this; }

    public Pokretan proteklo (double dt) {                // Sledeći korak rada.
        if ((t == dt) <= 0) {
            try {
                if (pred != null) {
                    if (pol == null) throw new GRadNemaPol (this);
                    pol.stavi (pred);
                    System.out.println ("Radnik " + this + " je stavio " + pred);
                }
                pred = napravi ();
                t = Math.random () * maxT;
            } catch (GPolica g) {
                System.out.println ("Radnik " + this + " je zatekao punu policu");
            } catch (usluge.Greska g) {
                System.out.println (g);
            }
        }
        return this;
    }
}
```

```
// Kupac.java - Klasa kupaca.

package radnja;
import predmetil.Predmet; → Zadatak 5.4

public class Kupac {

    private static int ukId;      // Poslednje korišćeni identifikator.
    private int id = ++ukId;       // Identifikator kupca.
    private char vrsta;           // Vrsta predmeta koji se kupuje.

    public Kupac (char v) { vrsta = v; } // Inicijalizacija.

    public int dohvatiId () { return id; } // Dohvatanje identifikatora.

    public char dohvatiVrs () { return vrsta; } // Dohvatanje vrste predmeta.
```

```

public void usluzen (Predmet p) {           // Preuzimanje predmeta.
    System.out.println (
        "Kupac " + this + " " + (p!=null ? "je dobio " + p
                                      : "nije dobio nista")
    );
}                                               // Tekstualni oblik.
public String toString () { return "K" + id + "(" + vrsta + ")"; }
}

```

// Radnja.java - Klasa radnji.

```

package radnja;
import polica.Polica;      → Zadatak 6.4
import pokretni.Pokretan;   → Zadatak 5.8
import radionica.Masina;    → Zadatak 6.9
import predmetil.Predmet;   → Zadatak 5.4
import liste.Lista;         → Zadatak 6.5

public class Radnja {

    private Masina[] masine = new Masina [10];      // Mašine za proizvodnju.
    private int brMas;                                // Broj mašina.
    private Polica polica;                            // Polica sa artiklima.
    private Casovnik casovnik = new Casovnik ();     // Generator takta.
    private Lista kupci = new Lista ();               // Kupci u radnji.

    public Radnja (                                     // Inicijalizacija:
        double maxQ,                                // - nosivost police,
        int kap,                                    // - broj mesta na polici,
        double maxUzaz,                             // - najduže vreme do ulaska kupca,
        double maxKup                                // - najduže vreme kupovine.
    ) {
        polica = new Polica (kap, maxQ);
        casovnik.dodaj (new Ulaz (maxUzaz));
        casovnik.dodaj (new Prodavac (maxKup));
    }

    public Radnja dodajMas (Masina mas)             // Nabavka mašine.
    { masine[brMas++] = mas; return this; }

    public Radnja zaposli (PokRadnik radnik) {      // Zaposljavanje radnika.
        casovnik.dodaj (radnik);
        radnik.polica (polica);
        radnik.rasporedi (masine[(int)(Math.random()*brMas)]);
        System.out.println ("Zaposlen je radnik " + radnik);
        return this;
    }                                                 // Otpuštanje radnika.

    public Radnja otpusti (PokRadnik radnik) throws GRadNemaRad {
        try { casovnik.izbaci (radnik); }
        catch (GCasNemaKli g) { throw new GRadNemaRad (radnik); }
        return this;
    }

    public void radi (double dt, double maxT)        // Radno vreme radnje.
    { casovnik.radi (dt, maxT); }
}

```

```

private class Ulaz implements Pokretan { // KLASA ULAZA U RADNUJU.

    private double maxT; // Najduže vreme do ulaska kupca.
    private double t; // Preostalo vreme do narednog dogadaja.
    private Kupac kup; // Sledеји kupac koji ulazi.

    private Ulaz (double tMax) { maxT = tMax; } // Inicijalizacija.

    public Pokretan proteklo (double dt) { // Sledеји korak rada.
        if ((t - dt) <= 0) {
            try {
                if (kup != null) {
                    kupci.dodaj_(kup);
                    System.out.println ("Kupac " + kup + " je usao");
                }
                int i = (int)(Math.random()*brMas);
                kup = new Kupac (masine[i].vrsta());
                t = Math.random () * maxT;
            } catch (Exception g) {}
        }
        return this;
    }
} // class Ulaz

private class Prodavac implements Pokretan { // KLASA PRODAVCA.

    private double maxT; // Najduže vreme kupovine.
    private double t; // Preostalo vreme do narednog dogadaja.
    private Kupac kup; // Kupac koji kupuje.

    private Prodavac (double tMax) { maxT = tMax; } // Inicijalizacija.

    public Pokretan proteklo (double dt) { // Sledеји korak rada.
        if ((t - dt) <= 0) {
            if (kup != null) {
                int i; Predmet p = null;
                for (i=0; i<polica.kapacitet(); i++)
                    try {
                        if ((p = polica.dohvati (i)).vr() == kup.dohvatiVrs())
                            break;
                    } catch (Exception g) {}
                if (p != null) try { polica.uzmi (i); } catch (Exception g) {}
                kup.usluzen (p);
                kup = null;
            }
            try {
                kup = (Kupac)kupci.naPrvi().dohvatiTek ();
                kupci.izbaciti(kup);
                t = Math.random () * maxT;
            } catch (Usluge.Greska g) {}
        }
        return this;
    }
} // class Prodavac
} // class Radnja

```

```
// GRadnja.java - Apstraktna klasa grešaka za radnje.

package radnja;

public abstract class GRadnja extends usluge.Greska {

    public GRadnja (String por) { super (por); } // Inicijalizacija.
    public GRadnja () {}
}
```

Zadatak 6.5

```
// GRadNemaPol.java - Klasa za greške: Radnik nema policu.

package radnja;

public class GRadNemaPol extends GRadnja {

    private PokRadnik radnik; // Problematični radnik.

    public GRadNemaPol (PokRadnik r) // Inicijalizacija.
        { super ("Nije odrednjena polica radniku [" + r + "]"); radnik = r; }

    public PokRadnik radnik () { return radnik; } // Dohvatanje radnika.
}
```

// GRadNemaRad.java - Klasa za greške: Radnik ne postoji u radnji.

```
package radnja;

public class GRadNemaRad extends GRadnja {

    private PokRadnik radnik; // Problematični radnik.

    public GRadNemaRad (PokRadnik r) // Inicijalizacija.
        { super ("U radnji ne postoji radnik [" + r + "]"); radnik = r; }

    public PokRadnik radnik () { return radnik; } // Dohvatanje radnika.
}
```

// RadnjaT.java - Ispitivanje klase radnja.

```
import radnja.*;
import polica.Polica;
import radionica.*;
```

Zadatak 6.4

Zadatak 6.9

```
public class RadnjaT {
    public static void main (String[] varg) {
        Radnja rad = new Radnja (50, 5, 2, 2);
        rad.dodajMas (new MasSfera (1,1));
        rad.dodajMas (new MasKvad (0.5, 1, 2, 3));
        rad.zaposli (new PokRadnik ("Marko", 4));
        rad.zaposli (new PokRadnik ("Stevo", 5));
        rad.zaposli (new PokRadnik ("Ljubo", 6));
        rad.radi (1.25, 20);
    }
}
```

### • RadnjaT

```
len je radnik Marko/1(S,0)
len je radnik Stevo/2(S,0)
len je radnik Ljubo/3(K,0)
: 0.0
: 1.25
k Marko/1(S,1) je stavio S[1.0|1.0]
: 2.5
K1(S) je usao
k Stevo/2(S,1) je stavio S[1.0|1.0]
: 3.75
K1(S) je dobio S[1.0|1.0]
k Stevo/2(S,2) je stavio S[1.0|1.0]
k Ljubo/3(K,1) je stavio K[0.5|1.0,2.0,3.0]
: 5.0
K2(K) je usao
k Marko/1(S,2) je stavio S[1.0|1.0]
k Stevo/2(S,3) je stavio S[1.0|1.0]
: 6.25
K3(S) je usao
: 7.5
K4(K) je usao
K2(K) je dobio K[0.5|1.0,2.0,3.0]
k Ljubo/3(K,2) je stavio K[0.5|1.0,2.0,3.0]
: 8.75
K5(S) je usao
K3(S) je dobio S[1.0|1.0]
k Marko/1(S,3) je stavio S[1.0|1.0]
ik Stevo/2(S,4) je zatekao punu policu
: 10.0
: K6(K) je usao
: K4(K) je dobio K[0.5|1.0,2.0,3.0]
ik Marko/1(S,4) je stavio S[1.0|1.0]
ik Stevo/2(S,4) je zatekao punu policu
: 11.25
c K7(S) je usao
ik Stevo/2(S,4) je zatekao punu policu
: 12.5
c K8(S) je usao
c K5(S) je dobio S[1.0|1.0]
ik Stevo/2(S,4) je stavio S[1.0|1.0]
e: 13.75
c K6(K) je dobio S[1.0|1.0]
ik Marko/1(S,5) je zatekao punu policu
ik Ljubo/3(K,3) je zatekao punu policu
e: 15.0
c K9(S) je usao
ik Marko/1(S,5) je zatekao punu policu
ik Ljubo/3(K,3) je zatekao punu policu
e: 16.25
c K7(S) je dobio S[1.0|1.0]
ik Marko/1(S,5) je stavio S[1.0|1.0]
ik Ljubo/3(K,3) je zatekao punu policu
e: 17.5
ic K10(K) je usao
ic K8(S) je dobio S[1.0|1.0]
ik Stevo/2(S,5) je stavio S[1.0|1.0]
ik Ljubo/3(K,3) je zatekao punu policu
e: 18.75
ic K9(S) je dobio S[1.0|1.0]
ik Ljubo/3(K,3) je stavio K[0.5|1.0,2.0,3.0]
e: 20.0
ic K11(K) je usao
ic K10(K) je dobio K[0.5|1.0,2.0,3.0]
ik Marko/1(S,6) je stavio S[1.0|1.0]
ik Ljubo/3(K,4) je zatekao punu policu
```

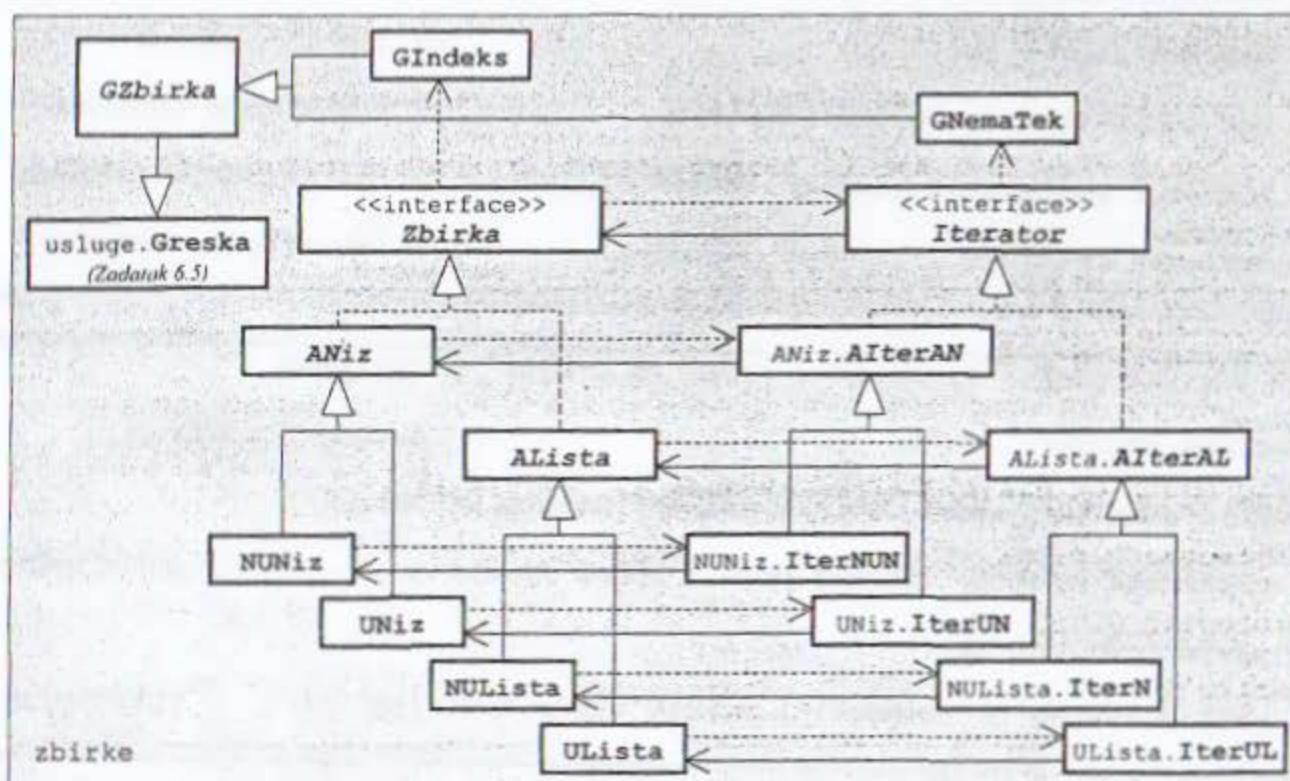
### Zadatak 6.11 Zbirke, iteratori, neuređeni i uređeni nizovi i liste

Napisati na jeziku Java sledeće tipove:

- Apstraktna *zbirka* celih brojeva predviđa:
  - 1 dohvatanje broja elemenata zbirke,
  - 2 dodavanje novog elementa zbirki,
  - 3 postavljanje i dohvatanje vrednosti elementa sa zadatim rednim brojem,
  - 4 izbacivanje elementa sa zadatim rednim brojem iz zbirke,
  - 5 stvaranje iteratora za obilazak zbirke, i
  - 6 sastavljanje tekstualnog oblika sadržaja zbirke.
- Apstraktan *iterator* predviđa:
  - 1 pomeranje na početak i na sledeći element zbirke,
  - 2 ispitivanje da li postoji tekući element zbirke,
  - 3 dohvatanje i postavljanje vrednosti tekućeg elementa zbirke, i
  - 4 izbacivanje tekućeg elementa iz zbirke.
- *Apstraktan niz* je zbirka koja elemente uskladištava u niz. Kapacitet niza se, po potrebi, dinamički povećava ili smanjuje.
- *Neuredjen niz* je niz kod kojeg se novi elementi stavljuju na kraj niza.
- *Uređeni niz* je niz kod kojeg su elementi uređeni po neopadajućem redosledu.
- *Apstraktna lista* je zbirka koja elemente uskladištava u jednostruko povezanu listu.
- *Neuređena lista* je lista kod koje se novi elementi stavljuju na kraj liste.
- *Uređena lista* je lista kod koje su elementi uređeni po neopadajućem redosledu.
- Za prijavljivanje izuzetaka koriste se klase izvedene iz zajedničke osnovne klase.

Napisati na jeziku Java program za ispitivanje prethodnih tipova.

Rešenje:



```
// Zbirka.java - Interfejs sekvencijalnih zbirki.

package zbirke;

public interface Zbirka {

    int vel();                                // Veličina zbirke.

    Zbirka dodaj (int b);                     // Dodavanje elementa.

    Zbirka postavi (int i, int b) throws GIndeks; // Postavljanje elementa.

    int dohvati (int i) throws GIndeks;        // Dohvatanje elementa.

    Zbirka brisi (int i) throws GIndeks;        // Izbacivanje elementa.

    Zbirka isprazni();                         // Pražnjenje zbirke.

    Iterator iterator();                       // Stvaranje iteratora.

    String toString();                         // Tekstualni oblik zbirke.
}
```

```
// Iterator.java - Interfejs iteratora sekvencijalnih zbirki.

package zbirke;

public interface Iterator {

    Iterator naPocetak();                    // Pomeranje na početak zbirke.

    Iterator naSledeci();                   // Pomeranje na sledeći element.

    boolean imaTek();                      // Ima li tekućeg?

    int dohvatiTek() throws GNemaTek;      // Dohvatanje tekućeg.

    Iterator postaviTek (int b) throws GNemaTek; // Postavljanje tekućeg.

    Iterator brisiTek () throws GNemaTek;   // Izbacivanje tekućeg.
}
```

*// GZbirka.java - Apstraktna klasa za greške sa zbirkama.*

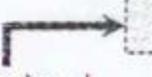
```
package zbirke;

public abstract class GZbirka extends usluge.Greska {

    protected GZbirka (String por) { super (por); }           // Inicijalizac.

    protected GZbirka () {}

    public String toString () { return super.toString (); } // Tekstualni
                                                          // oblik.
}
```

 Zadatak 6.5

```
// GNemaTek.java - Klasa za greške: Nema tekućeg elementa.

package zbirke;

public class GNemaTek extends GZbirka {                                // Inicijalizac.
    public GNemaTek () { super ("Nema tekuceg elementa u zbirci!"); }
}
```

```
// GInderks.java - Klasa za grešku: Nedozvoljeni indeks.

package zbirke;

public class GInderks extends GZbirka {

    private int ind;                                         // Nedozvoljeni indeks.

    public GInderks (int i) { ind = i; }                      // Inicijalizacija.

    public int ind () { return ind; }                          // Dohvatanje indeksa.

    public String toString ()                               // Tekstualni oblik.
        { return super.toString() + "Nedozvoljen indeks " + ind + "!"; }
}
```

```
// ANiz.java - Apstraktna klasa nizova.

package zbirke;

public abstract class ANiz implements Zbirka {

    protected int[] niz;                                     // Elementi niza.
    protected int n;                                         // Broj elemenata niza.

    public ANiz () { niz = new int [5]; }                    // Inicijalizacija.

    public int vel () { return n; }                           // Veličina zbirke.

    public int dohvati (int i) throws GInderks {           // Dohvatanje elementa.
        if (i<0 || i>=n) throw new GInderks (i);
        return niz[i];
    }

    public abstract Zbirka dodaj (int b);                  // Dodavanje elementa.

    protected void povecaj () {                             // Povećavanje kapaciteta
        int k = (n>50 ? n/10 : 5);                         // (10%, ali bar za 5).
        int[] pom = new int[n+k];
        for (int i=0; i<n; pom[i]=niz[i++]);
        niz = pom;
    }                                                       // Postavljanje elementa.

    public abstract Zbirka postavi (int i, int b) throws GInderks;
```

```

public Zbirka brisi (int i) throws GIndeks { // Izbacivanje elementa.
    if (i<0 || i>=n) throw new GIndeks (i);
    int b = niz[i], m = niz.length - n;
    if (m > niz.length/5 && m > 5) {           // - ako je bar 20% slobodno,
        m = (niz.length>10 ? niz.length/10 : 5); //   skrati niz za 10%,
        int[] pom = new int [niz.length - m];     //   ali bar za 5.
        for (int j=0, k=0; j<n; j++)
            if (j != i) pom[k++] = niz[j];
        niz = pom;
    } else
        while (i < n-1) niz[i] = niz[++i];
    n--;
    return this;
}

public Zbirka isprazni ()                      // Pražnjenje zbirke.
{ niz = new int [5]; n = 0; return this; }

public abstract Iterator iterator ();           // Stvaranje iteratora.

public String toString () {                     // Tekstualni oblik zbirke.
    StringBuffer s = new StringBuffer ("[");
    for (int i=0; i<n; i++) {
        if (i > 0) s.append (',');
        s.append (niz[i]);
    }
    return s.append (']').toString();
}

// UNUTRAŠNJA KLASA ITERATORA ZA NIZOVE.
protected abstract class AIterAN implements Iterator {
    protected int tek = 0;                         // Indeks tekućeg elementa.

    public Iterator naPocetak ()                  // Pomeranje na početak zbirke.
    { tek = 0; return this; }

    public Iterator naSledeci ()                   // Pomeranje na sledeći element.
    { tek++; return this; }

    public boolean imaTek ()                      // Ima li tekućeg elementa?
    { return tek < n; }

    public int dohvatiTek () throws GNemaTek { // Dohvatanje tekućeg
        if (!imaTek ()) throw new GNemaTek (); // elementa.
        return niz[tek];
    }

    public void postaviTek (int b) throws GNemaTek; // Postavljanje tekućeg elementa.

    public Iterator brisiTek () throws GNemaTek { // Brisanje tekućeg
        if (!imaTek ()) throw new GNemaTek (); // elementa.
        try { brisi (tek); } catch (GIndeks g) {}
        return this;
    }
}

```

```

// NUNiz.java - Klasa neuređenih nizova.

package zbirke;

public class NUNiz extends ANiz {

    public Zbirka dodaj (int b) {                                // Dodavanje
        if (n == niz.length) povecaj ();                          // elementa.
        niz[n++] = b;
        return this;
    }

    public Zbirka postavi (int i, int b) throws GIndeks { // Postavljanje
        if (i<0 || i>=n) throw new GIndeks (i);           // elementa.
        niz[i] = b;
        return this;
    }                                                       // Stvaranje iteratora.

    public Iterator iterator () { return new IterNUNiz (); }

    // UNUTRAŠNJA KLASA ITERATORA ZA NEUREĐENE NIZOVE.
    private class IterNUN extends AIterAN {

        public Iterator postaviTek (int b) throws GNematek { // Postavljanje
            if (!imaTek ()) throw new GNematek ();           // tekućeg
            niz[tek] = b;                                     // elementa.
            return this;
        }
    }
}

```

```

// UNiz.java - Klasa uređenih nizova.

package zbirke;

public class UNiz extends ANiz {

    public Zbirka dodaj (int b) {                                // Dodavanje
        if (n == niz.length) povecaj ();                          // elementa.
        int i = n; while (i>0 && niz[i-1]>b) niz[i] = niz[--i];
        niz[i] = b; n++;
        return this;
    }

    public Zbirka postavi (int i, int b) throws GIndeks { // Postavljanje
        if (i<0 || i>=n) throw new GIndeks (i);           // elementa.
        if (b > niz[i])      while (i< n && niz[i]<b) niz[i]=niz[++i];
        else if (b < niz[i]) while (i>=0 && niz[i]>b) niz[i]=niz[--i];
        niz[i] = b; novoMesto = i;
        return this;
    }

    private int novoMesto;          // Mesto elementa posle promene vrednosti.
                                    // Stvaranje iteratora.

    public Iterator iterator () { return new IterUNiz (); }
}

```

```
// UNUTRAŠNJA KLASA ITERATORA ZA UREDENE NIZOVE.
private class IterUN extends AIterAN {

    public Iterator postaviTek (int b) throws GNemaTek { // Postavljanje
        if (! imaTek()) throw new GNemaTek (); // tekućeg
        try { postavi (tek, b); } catch (GIndeks g) {} // elementa.
        tek = novoMesto;
        return this;
    }
}
```

```
// ALista.java - Apstraktna klasa listi.

package zbirke;

public abstract class ALista implements Zbirka {

    protected Elem prvi, posl; // Početak i kraj liste.
    protected int n; // Dužina liste.

    protected class Elem { // Element liste:
        int broj; // - sadržani broj,
        Elem sled; // - sledeći element,
        Elem (int b, Elem pret) { // - inicijalizacija.
            broj = b;
            if (pret == null) { sled = prvi; prvi = this; }
            else { sled = pret.sled; pret.sled = this; }
            posl = this;
        }
    }

    protected Elem nadji (int i) throws GIndeks { // Nalaženje elementa.
        if (i<0 || i>=n) throw new GIndeks (i);
        Elem tek = prvi; while (i-- > 0) tek = tek.sled;
        return tek;
    }

    public int vel () { return n; } // Veličina zbirke.

    public int dohvati (int i) throws GIndeks // Dohvatanje elementa.
    { return nadji (i).broj; }

    public abstract Zbirka dodaj (int b); // Dodavanje elementa.
                                         // Postavljanje elementa.

    public abstract Zbirka postavi (int i, int b) throws GIndeks;

    public Zbirka brisi (int i) throws GIndeks { // Izbacivanje elementa.
        Elem pret = nadji (i-1);
        if (pret.sled == null) throw new GIndeks (i);
        pret.sled = pret.sled.sled; n--;
        return this;
    }

    public Zbirka isprazni () // Pražnjenje zbirke.
    { prvi = posl = null; n = 0; return this; }

    public abstract Iterator iterator (); // Stvaranje iteratora.
```

```

public String toString () { // Tekstualni oblik zbirke.
    StringBuffer s = new StringBuffer ("[");
    for (Elem tek=prvi; tek!=null; tek=tek.sled) {
        if (tek != prvi) s.append (',');
        s.append (tek.broj);
    }
    return s.append (']').toString();
}

// UNUTRAŠNJA KLASA ITERATORA ZA LISTE.
protected abstract class AIterAL implements Iterator {

    protected Elem tek=prvi, pret=null; // Tekući i prethodni element.

    public Iterator naPocetak () // Pomeranje na početak zbirke.
        { tek = prvi; pret = null; return this; }

    public Iterator naSledeci () { // Pomeranje na sledeći element.
        if (tek != null) {pret = tek; tek = tek.sled;}
        return this;
    }

    public boolean imaTek () { return tek != null; } // Ima li tekućeg
                                                    // elementa.

    public int dohvatiTek () throws GNemaTek { // Dohvatanje tekućeg
        if (!imaTek ()) throw new GNemaTek (); // elementa.
        return tek.broj;
    }

    // Postavljanje tekućeg elementa.
    public abstract Iterator postaviTek (int b) throws GNemaTek;

    public Iterator brisiTek () throws GNemaTek { // Brisanje tekućeg
        if (!imaTek ()) throw new GNemaTek (); // elementa.
        pret.sled = tek = tek.sled; n--;
        return this;
    }
}
}

```

```

// NULista.java - Klasa neuredenih listi.

package zbirke;

public class NULista extends ALista {

    public Zbirka dodaj (int b) // Dodavanje
        { new Elem (b, posl); n++; return this; } // elementa.

    public Zbirka postavi (int i, int b) throws GIndeks // Postavljanje
        { nadji (i).broj = b; return this; } // elementa.

    public Iterator iterator () // Stvaranje
        { return new IterNULista (); } // iteratora.
}

```

```

// UNUTRAŠNJA KLASA ITERATORA ZA NEUREĐENE LISTE.
private class IterNUL extends AIterAL {

    public Iterator postaviTek (int b) throws GNemaTek { // Postavljanje
        if (!imaTek ()) throw new GNemaTek (); // tekućeg
        tek.broj = b; // elementa.
        return this;
    }
}
}

// ULista.java - Klasa uređenih listi.

package zbirke;

public class ULista extends ALista {

    public Zbirka dodaj (int b) { // Dodavanje elementa.
        Elek tek = prvi, pret = null;
        while (tek!=null && tek.broj<= b) { pret = tek; tek = tek.sled; }
        novi = new Elek (b, pret); npret = pret; n++;
        return this;
    }

    private Elek novi, npret; // Novododati i njemu prethodni element.

    public Zbirka postavi (int i, int b) throws GIndeks { // Postavljanje
        if (i<0 || i>=n) throw new GIndeks (i); // elementa.
        brisi (i); return dodaj (b);
    }

    public Iterator iterator () // Stvaranje
        { return new IterULista (); } // iteratora.

// UNUTRAŠNJA KLASA ITERATORA ZA UREĐENE LISTE.
private class IterUL extends AIterAL {

    public Iterator postaviTek (int b) throws GNemaTek { // Postavljanje
        if (!imaTek ()) throw new GNemaTek (); // tekućeg
        int i = 0; for (Elek t=prvi; t!=tek; t=t.sled) i++; // elementa.
        try { postavi (i, b); } catch (GIndeks g) {}
        tek = novi; pret = npret;
        return this;
    }
}
}

```

```

// ZbirkeT.java - Ispitivanje klase zbirk.
import zbirke.*;
public class ZbirkeT {
    public static void main (String[] varg) {
        Zbirka[] nizzb = { new NUNiz (), new UNiz (),
                           new NULista (), new ULista () };
        String[] nasl = { "NEUREDJEN NIZ", "UREDJEN NIZ",
                          "NEUREDJENA LISTA", "UREDJENA LISTA" };
        int[] niz = { 6, 3, 7, 9, 2, 0, 4, 1, 8, 5, 2, 5, 3, 8, 0 };
        for (int i=0; i<nizzb.length; i++)
            radi (nasl[i], nizzb[i], niz);
    }

    private static void radi (String nasl, Zbirka zb, int[] niz) {
        System.out.println ("\n" + nasl);
        try {
            for (int i=0; i<niz.length; zb.dodaj (niz[i++]));
            System.out.println ("Napravljena zbirka: " + zb);

            System.out.print ("Indeksiranjem:      ");
            for (int i=0; i<zb.vel(); i++)
                System.out.print (zb.dohvati(i) + " ");
            System.out.println ();

            System.out.print ("Iteratorom:      ");
            Iterator it = zb.iterator();
            for (it=it.naPocetak(); it.imaTek(); it.naSledeci())
                System.out.print (it.dohvatiTek() + " ");
            System.out.println ();

            System.out.println ("zb[4]=7 (indeksom): " + zb.postavi(4,7));

            it.naPocetak ();
            for (int i=0; i<7; i++) it.naSledeci();
            it.postaviTek (3);
            System.out.println ("zb[7]=3 (iterom): " + zb);
            System.out.println ("  tekuci:           " + it.dohvatiTek());

            it.brisiTek ();
            System.out.println ("Brisi tekuci:      " + zb);
            System.out.println ("  tekuci:           " + it.dohvatiTek());
            System.out.println ("Brisi zb[10]:       " + zb.brisi(10));
        } catch (GZbirka g) { System.out.println (g); }
    }
}

```

```
% java ZbirkeT
```

**NEUREDJEN NIZ**

Napravljena zborka: [6,3,7,9,2,0,4,1,8,5,2,5,3,8,0]  
 Indeksiranjem: 6 3 7 9 2 0 4 1 8 5 2 5 3 8 0  
 Iteratorom: 6 3 7 9 2 0 4 1 8 5 2 5 3 8 0  
 zb[4]=7 (indeksom): [6,3,7,9,7,0,4,1,8,5,2,5,3,8,0]  
 zb[7]=3 (iterom): [6,3,7,9,7,0,4,3,8,5,2,5,3,8,0]  
 tekuci: 3  
 Brisi tekuci: [6,3,7,9,7,0,4,8,5,2,5,3,8,0]  
 tekuci: 8  
 Brisi zb[10]: [6,3,7,9,7,0,4,8,5,2,3,8,0]

**UREDJEN NIZ**

Napravljena zborka: [0,0,1,2,2,3,3,4,5,5,6,7,8,8,9]  
 Indeksiranjem: 0 0 1 2 2 3 3 4 5 5 6 7 8 8 9  
 Iteratorom: 0 0 1 2 2 3 3 4 5 5 6 7 8 8 9  
 zb[4]=7 (indexom): [0,0,1,2,3,3,4,5,5,6,7,7,8,8,9]  
 zb[7]=3 (iterom): [0,0,1,2,3,3,3,4,5,6,7,7,8,8,9]  
 tekuci: 3  
 Brisi tekuci: [0,0,1,2,3,3,4,5,6,7,7,8,8,9]  
 tekuci: 3  
 Brisi zb[10]: [0,0,1,2,3,3,4,5,6,7,8,8,9]

**NEUREDJENA LISTA**

Napravljena zborka: [6,3,7,9,2,0,4,1,8,5,2,5,3,8,0]  
 Indeksiranjem: 6 3 7 9 2 0 4 1 8 5 2 5 3 8 0  
 Iteratorom: 6 3 7 9 2 0 4 1 8 5 2 5 3 8 0  
 zb[4]=7 (indeksom): [6,3,7,9,7,0,4,1,8,5,2,5,3,8,0]  
 zb[7]=3 (iterom): [6,3,7,9,7,0,4,3,8,5,2,5,3,8,0]  
 tekuci: 3  
 Brisi tekuci: [6,3,7,9,7,0,4,8,5,2,5,3,8,0]  
 tekuci: 8  
 Brisi zb[10]: [6,3,7,9,7,0,4,8,5,2,3,8,0]

**UREDJENA LISTA**

Napravljena zborka: [0,0,1,2,2,3,3,4,5,5,6,7,8,8,9]  
 Indeksiranjem: 0 0 1 2 2 3 3 4 5 5 6 7 8 8 9  
 Iteratorom: 0 0 1 2 2 3 3 4 5 5 6 7 8 8 9  
 zb[4]=7 (indeksom): [0,0,1,2,3,3,4,5,5,6,7,7,8,8,9]  
 zb[7]=3 (iterom): [0,0,1,2,3,3,3,4,5,6,7,7,8,8,9]  
 tekuci: 3  
 Brisi tekuci: [0,0,1,2,3,3,4,5,6,7,7,8,8,9]  
 tekuci: 4  
 Brisi zb[10]: [0,0,1,2,3,3,4,5,6,7,8,8,9]

## 7 Niti

### Zadatak 7.1 Vektori s konkurentnim izračunavanjem zbira i skalarnog proizvoda

Napisati na jeziku Java klasu vektora s realnim komponentama. Predviđeti:

- stvaranje vektora sa zadatim brojem komponenata,
- postavljanje i dohvatanje vrednosti zadate komponente,
- sastavljanje tekstualnog oblika vektora,
- stvaranje vektora kao zbir dva vektora konkurentnim sabiranjem komponenata,
- izračunavanje skalarnog proizvoda dva vektora konkurentnim množenjem komponenta, i
- glavnu funkciju za ispitivanje klase.

Rešenje:

```
// Vektorl.java - Klasa vektora s
//                 konkurentnom obradom.
// import usluge.Greska; Zadatak 6.5

public class Vektorl {

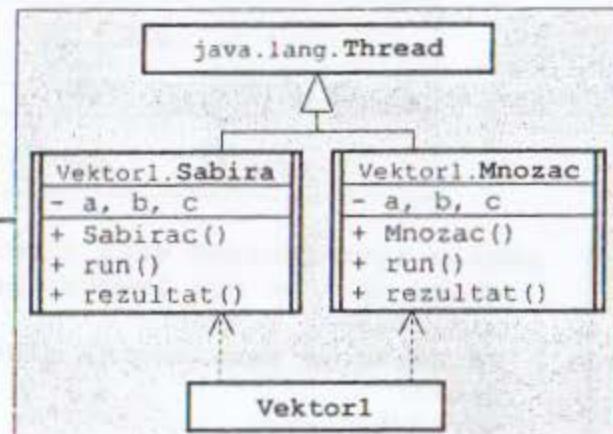
    private static class Sabirac extends Thread { // KLASA NITI ZA SABIRANJE:
        private double a, b, c; // Operandi i rezultat.
        public Sabirac (double p, double q) { a = p; b = q; } // Konstruktor.
        public void run () { c = a + b; } // Sadržaj niti.
        public double rezultat () { return c; } // Dohvatanje rezultata.
    }

    private static class Mnozac extends Thread { // KLASA NITI ZA MNOŽENJE:
        private double a, b, c; // Operandi i rezultat.
        public Mnozac (double p, double q) { a = p; b = q; } // Konstruktor.
        public void run () { c = a * b; } // Sadržaj niti.
        public double rezultat () { return c; } // Dohvatanje rezultata.
    }

    private double[] niz; // Komponente vektora.

    public Vektorl (int duz) { niz = new double [duz]; } // Konstruktor.
    // Postavljanje komponente.
    public Vektorl postavi (int ind, double broj) throws Greska {
        if (ind < 0 || ind >= niz.length)
            throw new Greska ("Nedozvoljen indeks!");
        niz[ind] = broj;
        return this;
    }

    public double uzmi (int ind) throws Greska { // Dohvatanje komponente.
        if (ind < 0 || ind >= niz.length)
            throw new Greska ("Nedozvoljen indeks!");
        return niz[ind];
    }
}
```



```

public String toString () {                                // Tekstualni oblik.
    String s = "(";
    for (int i=0; i<niz.length; i++)
        if (i > 0) s += ","; s += niz[i];
    return s + ")";
}

// Zbir dva vektora:
public static Vektorl zbir (Vektorl v1, Vektorl v2) throws Greska {
    int n = v1.niz.length, n2 = v2.niz.length;
    if (n != n2) throw new Greska ("Neusaglasene duzine vektora!");
    Sabirac[] sab = new Sabirac [n];
    for (int i=0; i<n; i++)                                // - pokretanje niti,
        (sab[i] = new Sabirac (v1.niz[i], v2.niz[i])).start ();
    Vektorl v = new Vektorl (n);
    for (int i=0; i<n; i++)                                // - sakupljanje rezultata.
        try { sab[i].join(); v.niz[i] = sab[i].rezultat (); }
        catch (InterruptedException g) {}
    return v;
}

// Skalarni proizvod:
public static double skalPro (Vektorl v1, Vektorl v2) throws Greska {
    int n = v1.niz.length, n2 = v2.niz.length;
    if (n != n2) throw new Greska ("Neusaglasene duzine vektora!");
    Mnozac[] mno = new Mnozac [n];
    for (int i=0; i<n; i++)                                // - pokretanje niti,
        (mno[i] = new Mnozac (v1.niz[i], v2.niz[i])).start ();
    double s = 0;
    for (int i=0; i<n; i++)                                // - sakupljanje rezultata.
        try { mno[i].join(); s += mno[i].rezultat (); }
        catch (InterruptedException g) {}
    return s;
}

public static void main (String[] vpar) {                // GLAVNA FUNKCIJA.
    while (true) {
        try {
            System.out.print ("n1? "); int n1 = Citaj.Int ();
            if (n1 < 0) break;
            System.out.print ("v1? "); Vektorl v1 = new Vektorl (n1);
            for (int i=0; i<n1; v1.postavi (i++, Citaj.Double ()));
            System.out.print ("n2? "); int n2 = Citaj.Int ();
            if (n2 < 0) break;
            System.out.print ("v2? ");
            Vektorl v2 = new Vektorl (n2);
            for (int i=0; i<n2;
                v2.postavi (i++,
                    Citaj.Double ()));
            System.out.println ("Zbir: "
                + Vektorl.zbir (v1, v2));
            System.out.println ("Skalpro: "
                + Vektorl.skalPro (v1, v2)
                + "\n");
        } catch (Greska g) {
            System.out.println (g + "\n");
        }
    }
}

```

```

% java Vektorl
n1? 5
v1? 1 2 3 4 5
n2? 5
v2? 5 4 3 2 1
Zbir: (6.0,6.0,6.0,6.0,6.0)
Skalpro: 35.0

n1? 5
v1? 1 2 3 4 5
n2? 2
v2? 1 2
*** Neusaglasene duzine vektora!
n1? -1

```

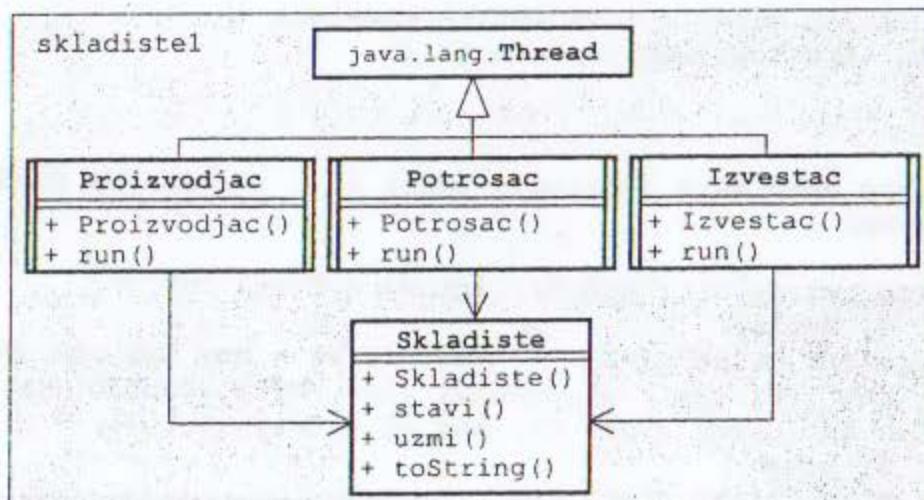
### Zadatak 7.2 Skladišta, aktivni proizvođači, potrošači i izveštaci

Napisati na jeziku Java sledeće tipove:

- *Skladište* celih brojeva ima jedinstven, automatski generisan identifikacioni broj i može da uskladištava zadati broj celih brojeva. Stvara se prazno, posle čega može da se dodaje i uzima po jedan ceo broj. Pri pokušaju stavljanja podatka u puno skladište, odnosno uzimanja podatka iz praznog skladišta, nit izvršioca radnje se privremeno blokira. Tekstualni oblik skladišta se sastoji od identifikacionog broja skladišta i niza sadržanih brojeva.
- Aktivan *proizvođač* ima jedinstven, automatski generisan identifikacioni broj. U slučajnim vremen-skim intervalima stavlja po jedan ceo broj u zadato skladište. Brojevi su oblika  $1000 id + br$ , gde su: *id* – identifikacioni broj proizvođača i *br* – uzastopni celi brojevi. Najkraće i najduže vreme proizvodnje brojeva je parametar proizvođača. Pre stavljanja u skladište na glavnom izlazu se prikazuje identifikacioni broj proizvođača i broj koji će se staviti u skladište.
- Aktivan *potrosač* ima jedinstven, automatski generisan identifikacioni broj. U slučajnim vremen-skim intervalima uzima po jedan ceo broj iz zadatog skladišta. Najkraće i najduže vreme potrošnje brojeva je parametar potrošača. Posle uzimanja iz skladišta na glavnom izlazu se prikazuje identifikacioni broj potrošača i uzeti broj.
- Aktivan *izveštac* ima jedinstven, automatski generisan identifikacioni broj. U regularnim vremen-skim intervalima prikazuje sadržaj zadatog skladišta na glavnom izlazu.

Napisati na jeziku Java program za ispitivanje prethodnih tipova.

**Rešenje:**



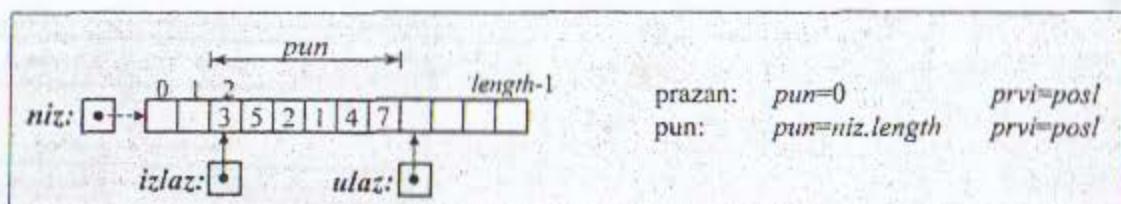
```
// Skladiste.java - Klasa za uskladištavanje podataka.

package skladistel;

public class Skladiste {

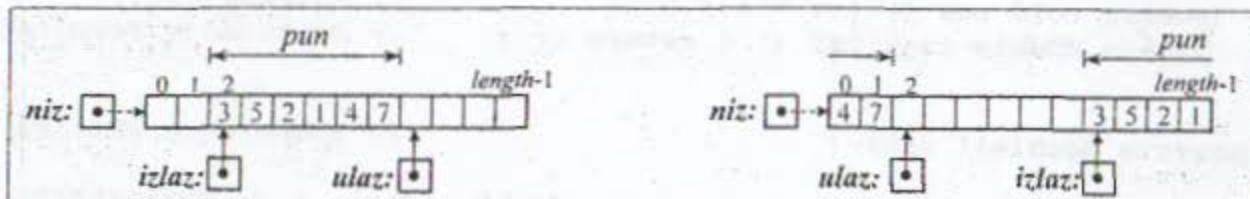
    private static int ukId = 0; // Poslednje korišćeni identifikator.
    private int id = ++ukId; // Identifikator skladišta.
    private int[] niz; // Niz za uskladištavanje podataka.
    private int ulaz, izlaz; // Mesto stavljanja i uzimanja podatka.
    private int pun; // Broj popunjениh mesta.

    public Skladiste (int kap) { niz = new int [kap]; } // Inicijalizacija.
```



```
public synchronized void stavi (int vredn) throws InterruptedException {
    while (pun == niz.length) wait (); // Stavljanje podatka.
    niz[ulaz++] = vredn;
    if (ulaz == niz.length) ulaz = 0;
    pun++;
    notifyAll ();
}

public synchronized int uzmi () throws InterruptedException {
    while (pun == 0) wait (); // Uzimanje podatka.
    int vredn = niz[izlaz++];
    if (izlaz == niz.length) izlaz = 0;
    pun--;
    notifyAll ();
    return vredn;
}
```



```
public synchronized String toString () { // Tekstualni oblik.
    StringBuffer s = new StringBuffer ("Sklad " + id + ": ");
    for (int i=0; i<pun; i++)
        s.append (niz[(izlaz+i) % niz.length]).append (' ');
    return s.toString ();
}
```

```

// Proizvodjac.java - Klasa proizvodača.

package skladistel;

public class Proizvodjac extends Thread {

    private static int ukId = 0;      // Poslednje korišćeni identifikator.
    private int id = ++ukId;          // Identifikator proizvodača.
    private Skladiste skladiste;     // Pridruženo skladište.
    private int minVreme, maxVreme;   // Najkraće i najduže vreme proizvodnje.
    private int broj;                // Poslednje proizvedeni podatak.

    public Proizvodjac (Skladiste sklad, int minVr, int maxVr) {
        minVreme = minVr; maxVreme = maxVr;           // Inicijalizacija.
        skladiste = sklad;
    }

    public Proizvodjac (Skladiste sklad) { this (sklad, 1000, 2000); }

    public void run () {               // Telo niti.
        System.out.println ("Proizv " + id + " krenuo");
        try {
            while (! interrupted ()) {
                sleep ((long) (minVreme + Math.random() * (maxVreme-minVreme)));
                int vredn = id*1000 + ++broj;
                System.out.println ("Proizv " + id + " stavlja " + vredn);
                skladiste.stavi (vredn);
            }
        } catch (InterruptedException g) {}
        System.out.println ("Proizv " + id + " zavrsio");
    }
}

```

```

// Potrosac.java - Klasa potrošača.

package skladistel;

public class Potrosac extends Thread {

    private static int ukId = 0;      // Poslednje korišćeni identifikator.
    private int id = ++ukId;          // Identifikator potrošača.
    private Skladiste skladiste;     // Pridruženo skladište.
    private int minVreme, maxVreme;   // Najkraće i najduže vreme potrošnje.

    public Potrosac (Skladiste sklad, int minVr, int maxVr) {
        minVreme = minVr; maxVreme = maxVr;                      // Inicijalizacija.
        skladiste = sklad;
    }

    public Potrosac (Skladiste sklad) { this (sklad, 1000, 2000); }

    public void run () {                         // Telo niti.
        System.out.println ("Potros " + id + " krenuo");
        try {
            while (! interrupted ()) {
                int vredn = skladiste.uzmi ();
                System.out.println ("Potros " + id + " uzeo " + vredn);
                sleep ((long) (minVreme + Math.random () * (maxVreme-minVreme)));
            }
        } catch (InterruptedException g) {}
        System.out.println ("Potros " + id + " zavrsio");
    }
}

```

```

// Izvestac.java - Klasa izveštaka.

package skladistel;

public class Izvestac extends Thread {
    private static int ukId = 0;      // Poslednje korišćeni identifikator.
    private int id = ++ukId;          // Identifikator izveštaka.
    private Skladiste skladiste;     // Pridruženo skladište.
    private int perioda;             // Perioda izveštavanja.

    public Izvestac (Skladiste sklad, int per)           // Inicijalizacija.
        { perioda = per; skladiste = sklad; }

    public Izvestac (Skladiste sklad) { this (sklad, 5000); }

    public void run () {                                // Telo niti.
        System.out.println ("Izvest " + id + " krenuo");
        try {
            while (! interrupted ()) {
                sleep (perioda);
                System.out.println ("Izvest " + id + " [" + skladiste + "]");
            }
        } catch (InterruptedException g) {}
        System.out.println ("Izvest " + id + " zavrsio");
    }
}

```

```
// SkladistelT.java - Ispitivanje klasa proizvodača i potrošača.

import skladistel.*;

public class SkladistelT {
    public static void main (String[] varg) {
        Skladiste s = new Skladiste (3);
        Thread[] niti = new Thread [11];
        for (int i=0; i<5; i++) {
            niti[i] = new Proizvodjac (s);
            niti[i+5] = new Potrosac (s);
        }
        niti[10] = new Izvestac (s);
        for (Thread nit: niti) nit.start ();
        try { Thread.sleep (15000); }
        catch (InterruptedException g) {}
        for (Thread nit: niti) nit.interrupt ();
        System.out.println ("Glavna - zavrsila");
    }
}
```

```
$ java SkladistelT
Proizv 1 krenuo
Proizv 2 krenuo
Proizv 3 krenuo
Proizv 4 krenuo
Proizv 5 krenuo
Potros 1 krenuo
Potros 2 krenuo
Potros 3 krenuo
Potros 4 krenuo
Potros 5 krenuo
Izvest 1 krenuo
Proizv 3 stavljaj 3001
Potros 1 uzeo 3001
Proizv 1 stavljaj 1001
Potros 2 uzeo 1001
Proizv 2 stavljaj 2001
Potros 3 uzeo 2001
Proizv 4 stavljaj 4001
Potros 4 uzeo 4001
Proizv 5 stavljaj 5001
Potros 5 uzeo 5001
Proizv 2 stavljaj 2002
Potros 1 uzeo 2002
Proizv 3 stavljaj 3002
Potros 2 uzeo 3002
Proizv 5 stavljaj 5002
Potros 3 uzeo 5002
Proizv 4 stavljaj 4002
Potros 4 uzeo 4002
Proizv 1 stavljaj 1002
Potros 5 uzeo 1002
Proizv 3 stavljaj 3003
Potros 1 uzeo 3003
Proizv 2 stavljaj 2003
Proizv 4 stavljaj 4003
Potros 2 uzeo 2003
Potros 3 uzeo 4003
Proizv 5 stavljaj 5003
Potros 4 uzeo 5003
```

```
Izvest 1 [Sklad 1: ]
Proizv 1 stavljaj 1003
Proizv 3 stavljaj 3004
Potros 5 uzeo 1003
Potros 1 uzeo 3004
Proizv 2 stavljaj 2004
Potros 2 uzeo 2004
Proizv 4 stavljaj 4004
Proizv 5 stavljaj 5004
Proizv 1 stavljaj 1004
Potros 4 uzeo 4004
Potros 3 uzeo 5004
Potros 1 uzeo 1004
Proizv 5 stavljaj 5005
Potros 5 uzeo 5005
Proizv 3 stavljaj 3005
Proizv 2 stavljaj 2005
Potros 2 uzeo 3005
Potros 3 uzeo 2005
Proizv 1 stavljaj 1005
Potros 4 uzeo 1005
Proizv 4 stavljaj 4005
Proizv 3 stavljaj 3006
Potros 5 uzeo 4005
Potros 2 uzeo 3006
Proizv 4 stavljaj 4006
Proizv 1 stavljaj 1006
Proizv 5 stavljaj 5006
Potros 4 uzeo 4006
Potros 1 uzeo 1006
Proizv 2 stavljaj 2006
Potros 3 uzeo 5006
Proizv 3 stavljaj 3007
Potros 2 uzeo 2006
Izvest 1 [Sklad 1: 3007 ]
Potros 5 uzeo 3007
Proizv 4 stavljaj 4007
Proizv 1 stavljaj 1007
Potros 4 uzeo 4007
Potros 1 uzeo 1007
```

```
Proizv 5 stavljaj 5007
Proizv 2 stavljaj 2007
Potros 2 uzeo 5007
Potros 3 uzeo 2007
Proizv 3 stavljaj 3008
Potros 5 uzeo 3008
Proizv 4 stavljaj 4008
Proizv 1 stavljaj 1008
Potros 2 uzeo 4008
Potros 1 uzeo 1008
Proizv 2 stavljaj 2008
Potros 4 uzeo 2008
Proizv 5 stavljaj 5008
Proizv 4 stavljaj 4009
Proizv 1 stavljaj 1009
Proizv 3 stavljaj 3009
Potros 2 uzeo 5008
Potros 3 uzeo 4009
Potros 5 uzeo 1009
Potros 4 uzeo 3009
Proizv 5 stavljaj 5009
Potros 1 uzeo 5009
Proizv 2 stavljaj 2009
Proizv 4 stavljaj 4010
Potros 4 uzeo 2009
Proizv 1 stavljaj 1010
Proizv 3 stavljaj 3010
Proizv 1 zavrsio
Proizv 3 zavrsio
Proizv 4 zavrsio
Potros 1 zavrsio
Potros 2 zavrsio
Potros 4 zavrsio
Izvest 1 [Sklad 1: 4010 1010 3010 ]
Proizv 2 zavrsio
Proizv 5 zavrsio
Potros 3 zavrsio
Glavna - zavrsila
Potros 5 zavrsio
Izvest 1 zavrsio
```

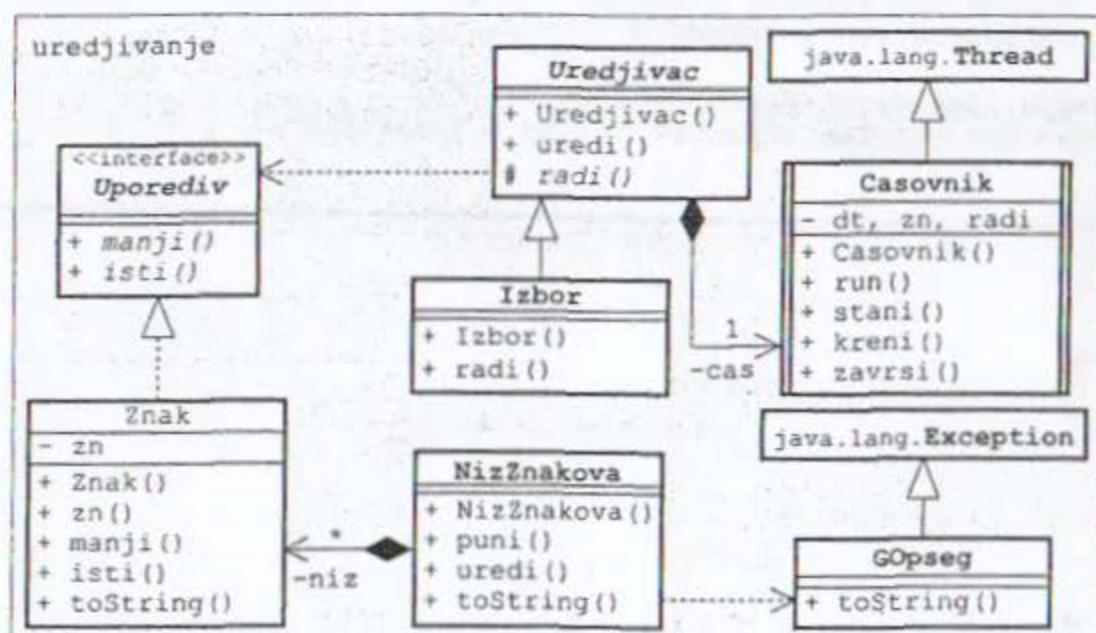
### Zadatak 7.3 Uporedivi znakovi, aktivni časovnici, uređivači i nizovi znakova

Napisati na jeziku Java sledeći paket tipova:

- **Uporediv** pojam može da ispita da li je manji od drugog pojma i da li je jednak drugom pojmu.
- Uporediv **znak** sadrži vrednost tipa **char** koja može da se dohvati i može da se sastavi tekstualni oblik znaka.
- Aktivan **časovnik** u zadatim vremenskim intervalima ispiše zadatu vrednost tipa **char** na glavnom izlazu. Rad časovnika može da se zaustavi, nastavi dalje i da se prekine.
- Apstraktan **uredivač** sadrži časovnik koji aktivira za vreme uređivanja. Može da uređuje zadati niz uporedivih pojmoveva.
- **Izbor** je uređivač koji primenjuje metodu izbora. Na kraju svakog koraka rada pauzira 1 ms.
- **Niz znakova** sadrži niz uporedivih znakova zadate dužine. Niz može da se napuni slučajnim znakovima unutar zadatog opsega vrednosti (podrazumevano od 'a' do 'z'; greška je ako je početni znak opsega iza krajnjeg), da se uredi pomoću zadatog uređivača i da se sastavi tekstualni oblik koji sadrži po 50 elemenata u svakom redu. Prilikom stvaranja niz se puni na podrazumevani način.

Napisati na jeziku Java program koji napravi jedan niz znakova, ispiše ga na glavnom izlazu, uredi niz i ponovo ispiše na glavnom izlazu. Svi parametri rada mogu da budu konstante (ne treba ništa čitati s glavnog ulaza).

**Rešenje:**



```

// Uporediv.java - Interfejs uporedivih pojmoveva.

package uredjivanje;

public interface Uporediv {

    boolean manji (Uporediv u); // Da li je manji?

    boolean isti (Uporediv u); // Da li je isti?
}
  
```

```
// Znak.java - Klasa uporedivih znakova.

package uredjivanje;

public class Znak implements Uporediv {
    private char zn;                                // Vrednost znaka.

    public Znak (char zn) { this.zn = zn; }           // Inicijalizacija.

    public char znak () { return zn; }                // Dohvatanje znaka.

    public boolean manji (Uporediv u)               // Da li je manji?
        { return zn < ((Znak)u).zn; }

    public boolean isti (Uporediv u)                 // Da li je isti?
        { return zn == ((Znak)u).zn; }

    public String toStrng ()                        // Tekstualni oblik.
        { return Character.toString (zn); }
}
```

```
// Casovnik.java - Klasa aktivnih časovnika.

package uredjivanje;

public class Casovnik extends Thread {
    private int dt;                                // Interval ispisivanja.
    private char zn;                                // Ispisivani znak.
    private boolean radi = false;                   // Da li treba da radi?

    public Casovnik (int dt, char zn)   // Inicijalizacija.
        { this.dt = dt; this.zn = zn; start (); }

    public void run () {                           // Telo niti.
        try {
            while (! interrupted ()) {
                if (! radi) synchronized (this) { wait (); }
                sleep (dt);
                System.out.print (zn);
            }
        } catch (InterruptedException g) {}
    }

    public synchronized void stani ()   // Privremeno zaustavljanje.
        { radi = false; }

    public synchronized void kreni ()    // Nastavak rada.
        { radi = true; notify (); }

    public void zavrси ()                  // Završetak niti.
        { interrupt (); }
}
```

```
// Uredjivac.java - Klasa apstraktnih uređivača.

package uredjivanje;

public abstract class Uredjivac {
    private Casovnik cas; // Sadržani časovnik.

    public Uredjivac (Casovnik cas) // Inicijalizacija.
        { this.cas = cas; }

    public void uredi (Uporediv[] u) { // Organizovanje uređivanja.
        cas.kreni ();
        radi (u);
        cas.stani ();
    }

    protected abstract void radi (Uporediv[] u); // Uredivanje.
}
```

```
// Izbor.java - Klasa uređivača metodom izbora.

package uredjivanje;

public class Izbor extends Uredjivac {

    public Izbor (Casovnik cas) { super (cas); } // Inicijalizacija.

    protected void radi (Uporediv[] niz) { // Uredivanje.
        for (int i=0; i<niz.length-1; i++)
            for (int j=i+1; j<niz.length; j++) {
                if (niz[j].manji (niz[i]))
                    { Uporediv u = niz[i]; niz[i] = niz[j]; niz[j] = u; }
                try { Thread.sleep (1); } catch (InterruptedException g) {}
            }
    }
}
```

```
// GOpseg.java - Klasa grešaka: Neispravan opseg vrednosti.

package uredjivanje;

public class GOpseg extends Exception {

    public String toString () { return "**** Neispravan opseg vrednosti!"; }
}
```

```
// NizZnakova.java - Klasa nizova uporedivih znakova.

package uredjivanje;

public class NizZnakova {
    private Znak[] niz; // Sadržani niz.

    public NizZnakova (int n) // Inicijalizacija.
        { niz = new Znak [n]; puni (); }
```

```

    // Popunjavanje niza.
public void puni (Znak p, Znak q) throws GOpseg { // - zadatim opsegom,
    if (q.manji (p)) throw new GOpseg ();
    for (int i=0; i<niz.length; i++)
        niz[i] = new Znak (
            (char) (p.znak() + Math.random()*(q.znak()-p.znak()+1))
        );
}

public void puni () { // - podrazumevanim
    try { puni (new Znak('a'), new Znak ('z')); } // opsegom.
    catch (GOpseg g) {}
}

public void uredi (Uredjivac u) // Uređivanje niza.
    ( u.uredi (niz); )

public String toString () { // Tekstualni oblik.
    StringBuffer s = new StringBuffer ();
    for (int i=0; i<niz.length; i++) {
        s.append (niz[i].znak());
        if (i%50==49 || i==niz.length-1)
            s.append ('\n');
    }
    return s.toString ();
}
}

```

```

// UredjivanjeT.java - Ispitivanje uređivanja praćenjem protoka vremena.

import uređivanje.*;

public class UredjivanjeT {
    public static void main (String[] varg) {
        Casovnik cas = new Casovnik (200, '**');
        Uredjivac izbor = new Izbor (cas);
        NizZnakova niz = new NizZnakova (75);
        System.out.print (niz + "\n");
        niz.uredi (izbor);
        System.out.print ("\n\n" + niz);
        cas.zavrsi ();
    }
}

```

```

# UredjivanjeT
znaarvlpmpfcncmixnxvguaehhinnbvonbewimnyengwcltmublt
zaggtsuityzjmjtrlsjsrqny
*****
```

```

aaabbcccccffffggggghiiijjjlllmmmmmmnnnnnnnnnopqrr
rssssttttuvwxyzxxxxzzz
```

## 8 Grafička korisnička površ

### Zadatak 8.1 Ispisivanje teksta u prozoru

Napisati na jeziku Java program s grafičkom korisničkom površi koji ispisuje tekst u glavnom prozoru.

**Rešenje:**

```
// Pozdrav2.java - Jednostavan program s prozorom.

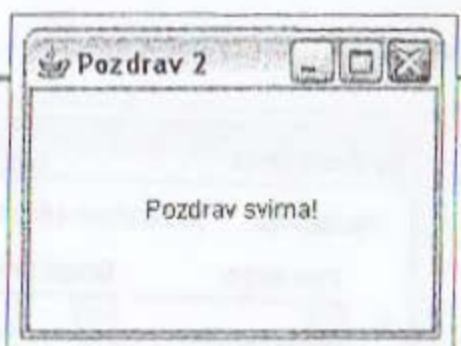
import java.awt.*;
import java.awt.event.*;

public class Pozdrav2 extends Frame {

    private Pozdrav2 () { // Sastavljanje prozora.
        super ("Pozdrav 2");
        setSize (300, 200);
        add (new Label ("Pozdrav svima!", Label.CENTER));
        addWindowListener (new ProzorDogadjaji ());
        setVisible (true);
    }

    private class ProzorDogadjaji extends WindowAdapter { // Rukovanje
        public void windowClosing (WindowEvent d)           // događajima
            { dispose (); }                                // prozora.
    }

    public static void main (String[] varg)             // Glavna funkcija.
        { new Pozdrav2 (); }
}
```



### Zadatak 8.2 Izračunavanje zbiru dva broja

Napisati na jeziku Java program s grafičkom korisničkom površi za izračunavanje zbiru dva cela broja.

Rešenje:

```
// Zbir2.java - Zbir dva broja u grafičkom
// okruženju.

import java.awt.*;
import java.awt.event.*;

public class Zbir2 extends Frame {

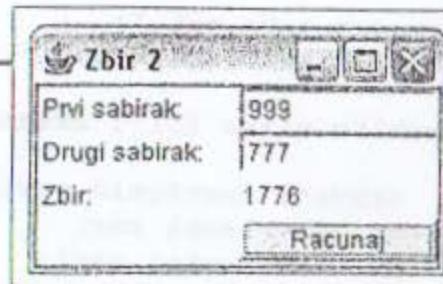
    private TextField prvi, drugi; // Prvi i drugi sabirak
    private Label rez;           // Rezultat.
    private Button radi;         // Pokretač računanja.

    private Zbir2 () {          // Sastavljanje prozora.
        super ("Zbir 2");
        setBounds (100, 100, 200, 120);
        addWindowListener (new ProzorDogadjaji ());
        setLayout (new GridLayout (4, 2));
        add (new Label ("Prvi sabirak:"));
        add (prvi = new TextField ("0"));
        add (new Label ("Drugi sabirak:"));
        add (drugi = new TextField ("0"));
        add (new Label ("Zbir:"));
        add (rez = new Label ("0"));
        add (new Label ());
        add (radi = new Button ("Racunaj"));
        radi.addActionListener (new RadiAkcija ());
        setVisible (true);
    }

    private class RadiAkcija implements ActionListener { // Rukovanje
        public void actionPerformed (ActionEvent d) {           // dogadajima
            try {                                              // dugmeta.
                rez.setText (Integer.toString (
                    Integer.parseInt (prvi.getText ()) +
                    Integer.parseInt (drugi.getText ()))
            );
            } catch (NumberFormatException g) { rez.setText ("GRESKA"); }
        }
    }

    private class ProzorDogadjaji extends WindowAdapter { // Rukovanje
        public void windowClosing (WindowEvent d) {           // dogadajima
            dispose (); }                                     // prozora.
    }

    public static void main (String[] varg) {               // Glavna funkcija.
        new Zbir2 ();
    }
}
```



```

// Zbir3.java - Zbir dva broja u grafičkom
// okruženju.

import java.awt.*;
import java.awt.event.*;

public class Zbir3 extends Frame {

    private TextField prvi, drugi; // Prvi i drugi sabirak
    private Label rez;           // Rezultat.
    private Button radi;         // Pokretač računanja.

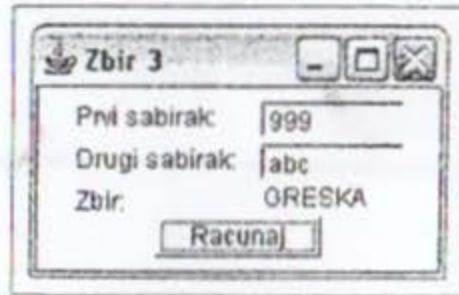
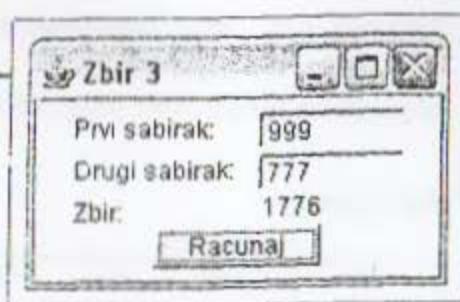
    private Zbir3 () {          // Sastavljanje prozora.
        super ("Zbir 3");
        setBounds (100, 100, 200, 120);
        addWindowListener (new ProzorDogadjaji ());
        setLayout (null);
        add (new Label ("Prvi sabirak:")) .setBounds ( 20, 36, 80, 15);
        add (new Label ("Drugi sabirak:")) .setBounds ( 20, 56, 80, 15);
        add (new Label ("Zbir:")) .setBounds ( 20, 76, 80, 15);
        add (prvi = new TextField ("0")) .setBounds (110, 36, 70, 18);
        add (drugi = new TextField ("0")) .setBounds (110, 56, 70, 18);
        add (rez = new Label ("0")) .setBounds (110, 76, 67, 10);
        add (radi = new Button ("Racunaj")).setBounds ( 60, 90, 80, 20);
        radi.addMouseListener (new RadiDogadjajiMisem ());
        setVisible (true);
    }

    private class RadiDogadjajiMisem extends MouseAdapter { // Rukovanje
        public void mouseClicked (MouseEvent d) {                  // dogadajima
            try {                                                 // dugmeta.
                rez.setText ( Integer.toString (
                    Integer.parseInt (prvi.getText ()) +
                    Integer.parseInt (drugi.getText ())
                ));
            } catch (NumberFormatException g) { rez.setText ("GRESKA"); }
        }
    }

    private class ProzorDogadjaji extends WindowAdapter { // Rukovanje
        public void windowClosing (WindowEvent d) {             // dogadajima
            dispose ();                                         // prozora.
        }
    }

    public static void main (String[] varg) {                  // Glavna funkcija.
        new Zbir3 ();
    }
}

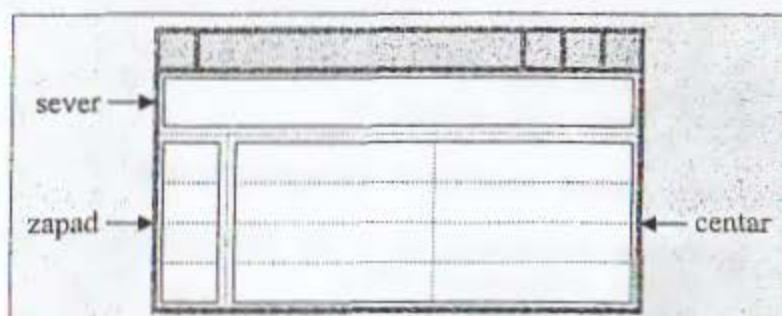
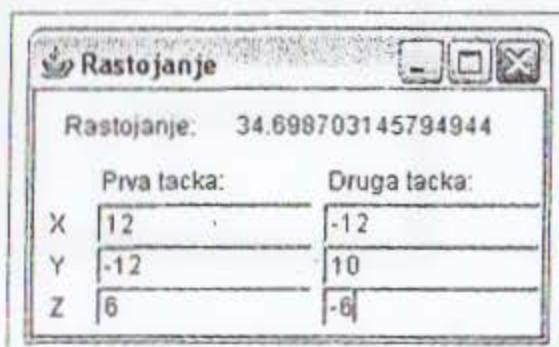
```



### Zadatak 8.3 Izračunavanje rastojanja između dve tačke u prostoru

Napisati na jeziku Java program s grafičkom korisničkom površi za izračunavanje rastojanja između dve tačke u prostoru.

**Rešenje:**



```

// Rastojanje.java - Rastojanje dve tačke u grafičkom okruženju.

import java.awt.*;
import java.awt.event.*;

public class Rastojanje extends Frame {

    private TextField[] koord = new TextField[6]; // Niz koordinata.
    private Label rez = new Label ("0"); // Rezultat.

    private Rastojanje () { // Inicijalizacija:
        super ("Rastojanje");
        setBounds (100, 100, 250, 150);
        popuniProzor (); // - popunjavanje prozora,
        setVisible (true);
        addWindowListener (new WindowAdapter () { // - obrada dogadaja prozora
            public void windowClosing (WindowEvent d) { dispose (); }
        });
    }

    private void popuniProzor () { // Popunjavanje prozora:
        Panel plo = new Panel (); add (plo, "Center"); // - ploča za koordinate,
        plo.setLayout (new GridLayout(4, 2, 4, 2));
        plo.add (new Label ("Prva tacka:"));
        plo.add (new Label ("Druga tacka:"));
        KoordPromena osmatrac = new KoordPromena ();
        for (int i=0; i<6; i++) {
            (koord[i] = new TextField ("0")).addTextListener (osmatrac);
            plo.add (koord[i]);
        }

        add (plo = new Panel (new GridLayout (4,1)), "West"); // - ploča za označavanje
        String[] ozn = {"", " X ", " Y ", " Z "}; // vrsta,
        for (int i=0; i<4; plo.add (new Label (ozn[i++]))); // vrsta,

        add (plo = new Panel (), "North"); // - ploča za rezultat.
        plo.add (new Label ("Rastojanje:"));
        plo.add (rez);
    }
}
  
```

```

private class KoordPromena implements TextListener { // Računanje
    public void textValueChanged (TextEvent d) {           // rastojanja,
        try {                                              // kad god se
            rez.setText (                                    // promeni neka
                Double.toString (                           // koordinata.
                    Math.sqrt (
                        Math.pow (Double.parseDouble(koord[0].getText())-
                            Double.parseDouble(koord[1].getText()), 2) +
                        Math.pow (Double.parseDouble(koord[2].getText())-
                            Double.parseDouble(koord[3].getText()), 2) +
                        Math.pow (Double.parseDouble(koord[4].getText())-
                            Double.parseDouble(koord[5].getText()), 2)
                    )
                );
        } catch (NumberFormatException g) {
            rez.setText ("GRESKA!");
        }
        validate (); // (da bi se komponente prozora prerasporedile
    }               // zbog promene broja cifara u rezultatu)
}

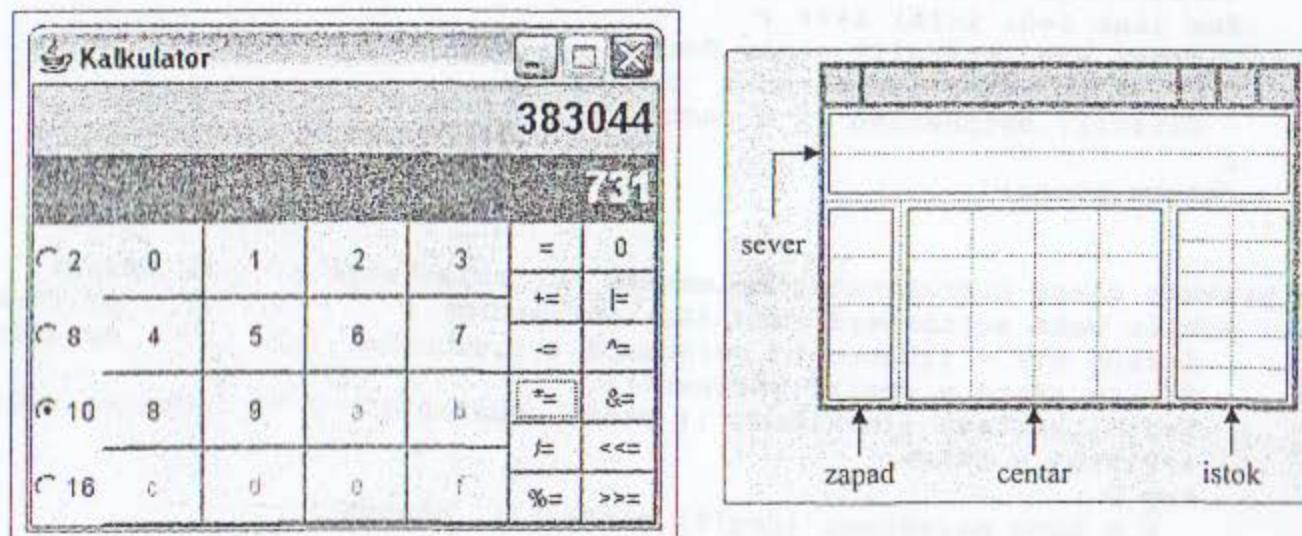
public static void main (String[] varg)                  // Glavna funkcija.
{ new Rastojanje (); }
}

```

#### Zadatak 8.4 Jednostavan kalkulator za cele brojeve

Napisati na jeziku Java program s grafičkom korisničkom površi za jednostavan kalkulator za rad s celim brojevima. Predvideti mogućnost dohvatanja rezultata poslednje izvršene operacije sa kalkulatorom.

**Rešenje:**



```
// Kalkulator.java - Jednostavan kalkulator u grafičkom okruženju.

import java.awt.*;
import java.awt.event.*;

public class Kalkulator extends Frame {

    private long x, y;                                // Registri kalkulatora.
    private Label[] reg = new Label [2];                // Prikaz registara.
    private static final int X = 0, Y = 1;              // Indeksi registara.
    private boolean noviBroj = true;                    // Unosi se novi broj.
    private int osnova = 10;                            // Osnova brojevnog sistema.
    private Button[] cifre = new Button [16];           // Dugmad s ciframa.
    private boolean unisti = false;                    // Da li unistiti prozor pri
                                                       // zatvaranju?

    private Panel ploRegistri () {                     // Polja teksta za registre.
        Panel ploca = new Panel (new GridLayout (2, 1));
        Color[] podloge = {Color.ORANGE, Color.GRAY};
        Color[] slova = {Color.BLUE, Color.WHITE};
        for (int i=0; i<2; i++) {
            reg[i] = new Label ("0", Label.RIGHT);
            reg[i].setFont (new Font (null, Font.BOLD, 20));
            reg[i].setForeground (slova[i]);
            reg[i].setBackground (podloge[i]);
            ploca.add (reg[i]);
        }
        return ploca;
    }
}
```

```

private void prikazi () { // Prikaz sadržaja registara.
    reg[X].setText (Long.toString (x, osnova));
    reg[Y].setText (Long.toString (y, osnova));
}

private Panel ploCifre () { // Dugmad s ciframa.
    Panel ploca = new Panel (new GridLayout (4, 4));
    CifraAkcija osmatrac = new CifraAkcija ();
    for (int i=0; i<16; i++) {
        ploca.add (cifre[i] = new Button (Integer.toString (i,16)));
        cifre[i].addActionListener (osmatrac);
        cifre[i].setEnabled (i < osnova);
    }
    return ploca;
}

private class CifraAkcija implements ActionListener { // Obrada
    public void actionPerformed (ActionEvent d) { // pritisaka
        String cif = ((Button)d.getSource()).getLabel(); // na cifre.
        String staro = reg[Y].getText();
        reg[Y].setText ((noviBroj || staro.equals("0")) ? "" : staro + cif);
        noviBroj = false;
        try {
            y = Long.parseLong (reg[Y].getText(), osnova);
        } catch (NumberFormatException g) {
            reg[Y].setText ("G R E S K A");
            noviBroj = true;
        }
    }
}

private Panel ploOsnove () { // Radio dugmad s osnovama
    String[] ozn = {"2", "8", "10", "16"}; // brojevnih sistema.
    Panel ploca = new Panel (new GridLayout (ozn.length, 1));
    ploca.setBackground (Color.YELLOW);
    CheckboxGroup grupa = new CheckboxGroup ();
    OsnovaPromena osmatrac = new OsnovaPromena ();
    for (String s: ozn) {
        Checkbox radio = new Checkbox (s, grupa,
                                       Integer.parseInt(s)==osnova);
        ploca.add (radio);
        radio.addItemListener (osmatrac);
    }
    return ploca;
}

private class OsnovaPromena implements ItemListener { // Obrada
    public void itemStateChanged (ItemEvent d) { // promene
        String ozn = ((Checkbox)(d.getSource())).getLabel(); // osnove
        osnova = Integer.parseInt (ozn); // brojevnog
        for (int i=0; i<16; i++) // sistema.
            cifre[i].setEnabled (i < osnova);
        prikazi ();
    }
}

```

```

private Panel ploOperatori () { // Dugmad s operatorima.
    String[] ozn = {"=", "0", "+=", "|=", "-=", "^=",
        "*=", "&=", "/=", "<<=", "%=", ">>="};
    Panel ploca = new Panel();
    ploca.setLayout (new GridLayout ((ozn.length+1)/2, 2));
    OperatorAkcija osmatrac = new OperatorAkcija ();
    for (String s: ozn) {
        Button dugme = new Button (s);
        dugme.setForeground (Color.BLUE);
        dugme.setBackground (Color.YELLOW);
        dugme.addActionListener (osmatrac);
        ploca.add (dugme);
    }
    return ploca;
}

private class OperatorAkcija implements ActionListener { // Obrada
    public void actionPerformed (ActionEvent d) { // pritisaka na
        try { // operatore.
            switch (((Button)d.getSource()).getLabel().charAt(0)) {
                case '=': x = y; break; case '0': y = 0; break;
                case '+': x += y; break; case '|': x |= y; break;
                case '-': x -= y; break; case '&': x &= y; break;
                case '*': x *= y; break; case '^': x ^= y; break;
                case '/': x /= y; break; case '<': x <= y; break;
                case '%': x %= y; break; case '>': x >= y; break;
            }
            prikazi ();
        } catch (ArithmetricException g) {
            reg[Y].setText ("GRESKA");
        }
        noviBroj = true;
    }
}

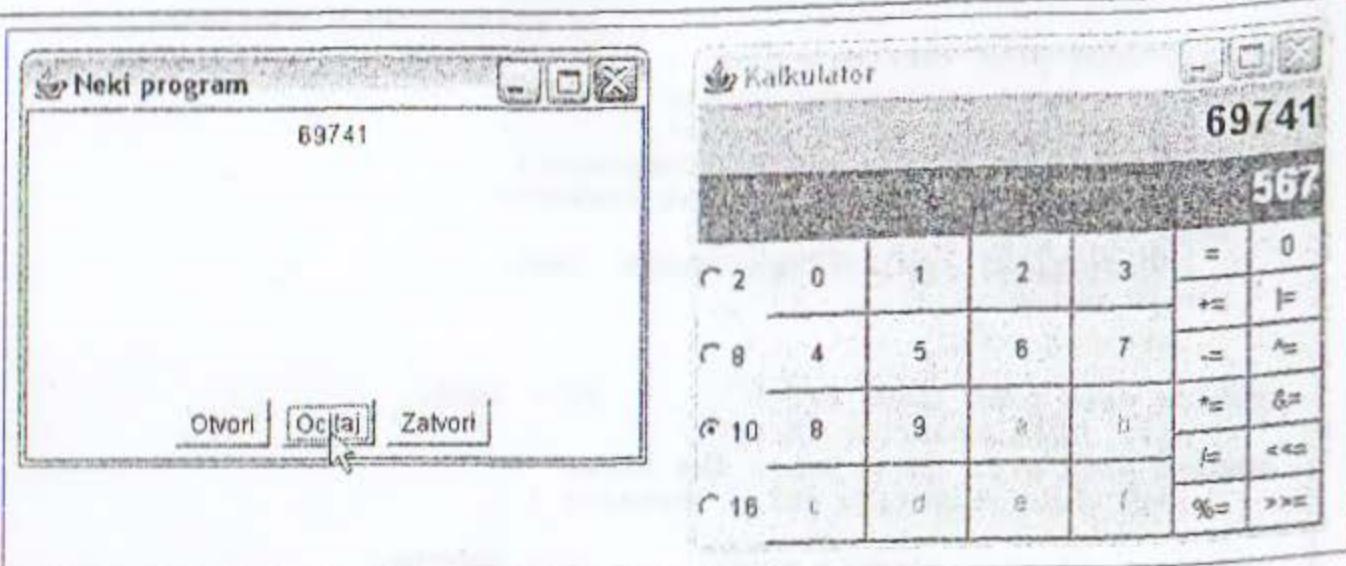
public Kalkulator (int x, int y) { // Inicijalizacija:
    super ("Kalkulator");
    setBounds (x, y, 305, 242);
    setResizable (false);
    add (ploRegistri(), BorderLayout.NORTH); // - popunjavanje prozora,
    add (ploCifre(), BorderLayout.CENTER);
    add (ploOsnove(), BorderLayout.WEST);
    add (ploOperatori(), BorderLayout.EAST);
    addWindowListener (new WindowAdapter () {
        public void windowClosing (WindowEvent d) // - uništavanje prozora.
        { if (unisti) dispose (); else setVisible (false); }
    });
}

public Kalkulator () { this (250, 50); } // Podrazumevani konstruktor

public long rezultat () { return x; } // Dohvatanje rezultata.

public static void main (String[] varg) { // Glavna funkcija.
    Kalkulator kalk = new Kalkulator ();
    kalk.setVisible (true);
    kalk.unisti = true;
}
}

```



```
// KalkulatorT.java - Nezavisan program koji koristi kalkulator.

import java.awt.*;
import java.awt.event.*;

public class KalkulatorT extends Frame {
    Kalkulator kalk = new Kalkulator (420, 50); // Prozor kalkulatora.
    Label rez = new Label ("", Label.CENTER); // Očitana vrednost.

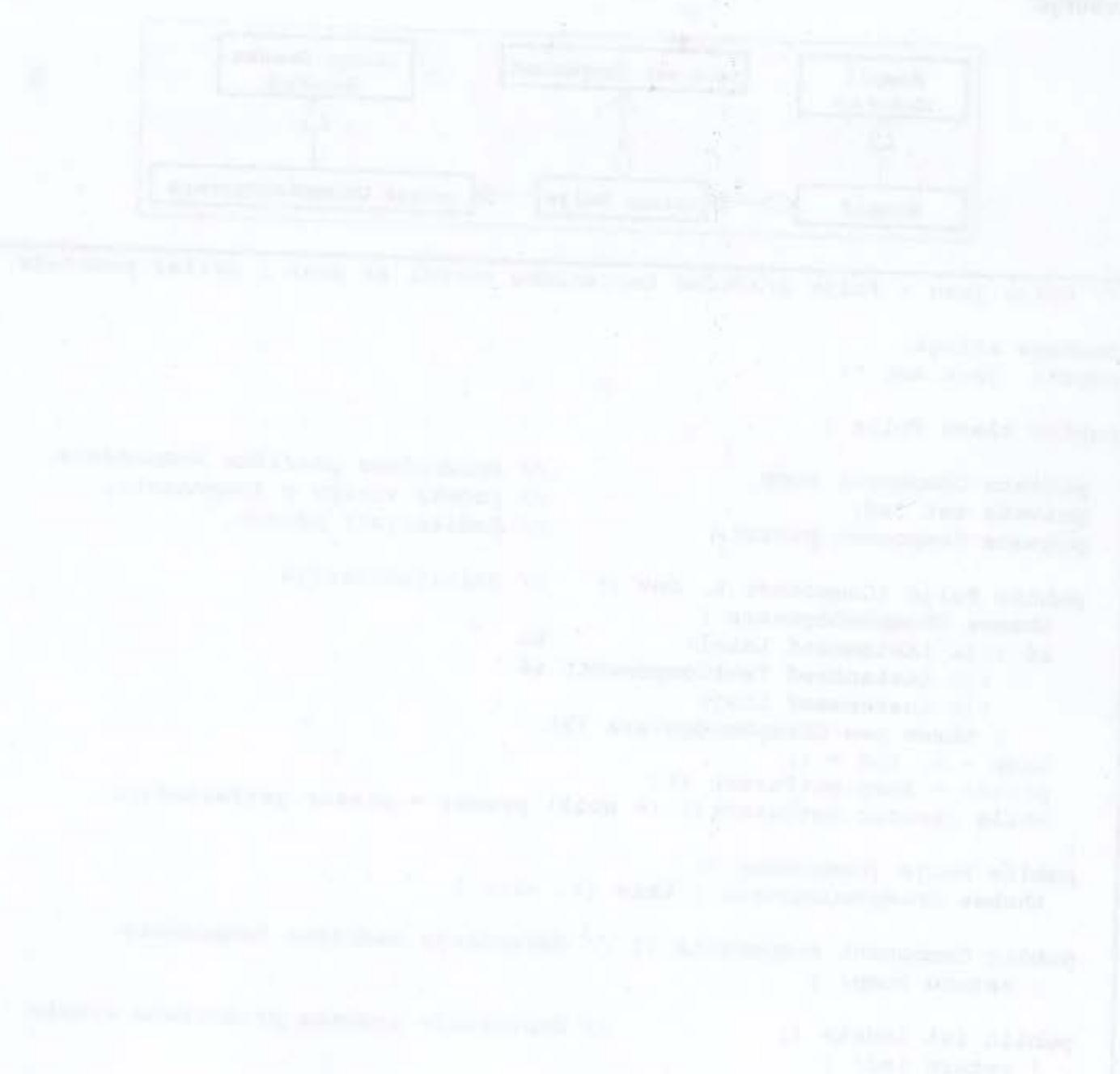
    private void popuniProzor () { // Popunjavanje prozora:
        add (rez, "North"); // - polje rezultata,
        Panel ploca = new Panel (); add (ploca, "South"); // - otvaranje
        Button dugme = new Button ("Otvori"); // kalkulatora,
        ploca.add (dugme);
        dugme.addActionListener (new ActionListener () {
            public void actionPerformed (ActionEvent d) {
                kalk.setVisible (true);
            }
        });
        ploca.add (dugme = new Button ("Ocitaj")); // - očitavanje
        dugme.addActionListener (new ActionListener () { // kalkulatora,
            public void actionPerformed (ActionEvent d) {
                rez.setText (Long.toString (kalk.rezultat ()));
            }
        });
        ploca.add (dugme = new Button ("Zatvori")); // - zatvaranje
        dugme.addActionListener (new ActionListener () { // kalkulatora.
            public void actionPerformed (ActionEvent d) {
                kalk.setVisible (false);
            }
        });
    }
}
```

```

private KalkulatorT () {                                // Inicijalizacija:
    super ("Neki program");
    setBounds (100, 50, 300, 200);
    popuniProzor ();                                // - popunjavanje prozora,
    addWindowListener (new WindowAdapter () {
        public void windowClosing (WindowEvent d) // - uništavanje oba
            { kalk.dispose (); dispose (); }      // prozora.
    });
}

public static void main (String[] varg) {           // Glavna funkcija.
    new KalkulatorT ().setVisible (true);
}
}

```



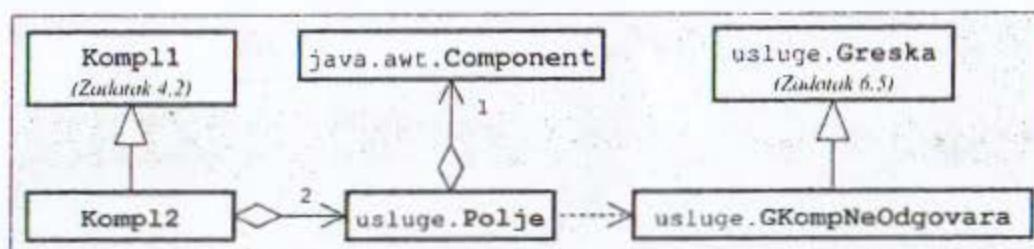
**Zadatak 8.5 Klasa za komunikaciju s grafičkom korisničkom površi i klasa kompleksnih brojeva s grafičkim prikazom**

Napisati na jeziku Java klasu koja omogućava prenos podataka standardnih tipova između date komponente grafičke korisničke površi i programa.

Proširiti klasu kompleksnih brojeva iz zadatka 4.2 tako da se svaka promena vrednosti kompleksnog broja (realnog ili imaginarnog dela) prikazuje na dve grafičke komponente pridružene kompleksnom broju. Omogućiti i menjanje vrednosti kompleksnog broja na osnovu trenutnog sadržaja pridruženih komponenti.

Napisati na jeziku Java program s grafičkom korisničkom površi za ispitivanje prethodnih klasa.

**Rešenje:**



```

// Polje.java - Polje grafičke korisničke površi za unos i prikaz podataka.

package usluge;
import java.awt.*;

public class Polje {

    private Component komp;           // Pridružena grafička komponenta.
    private int ind;                 // Indeks stavke u komponenti.
    private Component prozor;        // Roditeljski prozor.

    public Polje (Component k, int i) // Inicijalizacija.
        throws GKompNeOdgovara {
        if (!(k instanceof Label)      &&
            !(k instanceof TextComponent) &&
            !(k instanceof List))
            throw new GKompNeOdgovara (k);
        komp = k; ind = i;
        prozor = komp.getParent ();
        while (prozor.getParent() != null) prozor = prozor.getParent ();
    }
    public Polje (Component k)
        throws GKompNeOdgovara { this (k, -1); }

    public Component komponenta () // Dohvatanje sadržane komponente.
        { return komp; }

    public int indeks ()           // Dohvatanje indeksa pridružene stavke.
        { return ind; }
}
  
```

```

// PRIKAZIVANJE PODATAKA:
public void pisi (String tekst) { // - String,
    if (komp instanceof Label)
        ((Label)komp).setText (tekst);
    else if (komp instanceof TextComponent)
        ((TextComponent)komp).setText (tekst);
    else if (komp instanceof List)
        ((List)komp).replaceItem (tekst, ind);
    prozor.validate ();
}

public void pisi (long broj) // - long,
{ pisi (Long.toString (broj)); }
public void pisi (long broj, int baza)
{ pisi (Long.toString (broj, baza)); }

public void pisi (double broj) // - double,
{ pisi (Double.toString (broj)); }

public void pisi (boolean b) // - boolean,
{ pisi (new Boolean (b).toString ()); }

public void pisi (char znak) // - char,
{ pisi (new Character (znak).toString ()); }

public void pisi (Object o) // - Object.
{ pisi (o.toString ()); }

// UZIMANJE PODATAKA:
public String String () { // - String,
    if (komp instanceof Label)
        return ((Label)komp).getText ();
    else if (komp instanceof TextComponent)
        return ((TextComponent)komp).getText ();
    else if (komp instanceof List)
        return ((List)komp).getItem (ind);
    else return null;
}

public long Long () // - long,
{ return Long.parseLong (String ()); }

public long Long (int baza)
{ return Long.parseLong (String (), baza); }

public int Int () // - int,
{ return Integer.parseInt (String ()); }

public int Int (int baza)
{ return Integer.parseInt (String (), baza); }

public short Short () // - short,
{ return Short.parseShort (String ()); }

public short Short (int baza)
{ return Short.parseShort (String (), baza); }

```

```

public byte Byte ()           // - byte,
{ return Byte.parseByte (String()); }

public byte Byte (int baza)
{ return Byte.parseByte (String(), baza); }

public double Double ()       // - double,
{ return Double.parseDouble (String()); }

public float Float ()         // - float,
{ return Float.parseFloat (String()); }

public boolean Boolean ()     // - boolean,
{ return Boolean.parseBoolean (String()); }

public char Char ()           // - char.
{ return String().charAt (0); }
}

```

// GKompNeOdgovara.java - Klasa za greške: Tip komponente grafičke korisničke površi ne odgovara.

```

package usluge;
import java.awt.*;

public class GKompNeOdgovara extends usluge.Greska {
    public GKompNeOdgovara (Component k) {
        super ("Tip komponente " + k.getClass().getName() +
               " ne odgovara za polje ulaza/izlaza");
    }
}

```

Zadatak 6.5

// Kompl2.java - Klasa kompleksnih brojeva s grafičkim prikazom.

```

import usluge.Polje;           Zadatak 4.2
public class Kompl2 extends Kompl1 {

    private Polje rPolje, iPolje;      // Polja za prikaz delova broja.

    public Kompl2 (double r, double i, Polje rP, Polje iP) // Inicijalizacija.
    { super (r, i); rPolje = rP; iPolje = iP; prikazi (); }

    public Kompl2 (Polje rP, Polje iP) { this (0, 0, rP, iP); }

    public Kompl2 (double r, double i) { super (r, i); }

    public Kompl2 () {}

    public Kompl2 postavi (double r, double i)          // Postavljanje
    { super.postavi (r, i); prikazi (); return this; } // vrednosti.

    public Kompl2 postavi (Kompl1 z) { return postavi (z.re(), z.im()); }
}

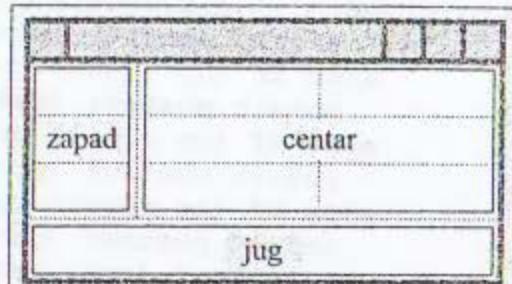
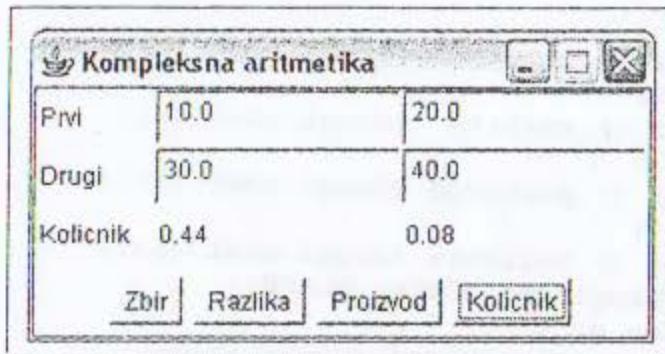
```

```

public Kompl2 uzmi () {                                // Uzimanje vrednosti
    if (rPolje != null) postaviRe (rPolje.Double ());
    if (iPolje != null) postaviIm (iPolje.Double ());
    return this;
}

public Kompl2 prikazi () {                            // Prikaz vrednosti
    if (rPolje != null) rPolje.pisi (re ());
    if (iPolje != null) iPolje.pisi (im ());
    return this;
}
}

```



```

// PoljeT.java - Ispitivanje klase polja grafičke korisničke površi.

import usluge.*;
import java.awt.*;
import java.awt.event.*;

public class PoljeT extends Frame {

    private Label rez;                                // Oznaka vrste rezultata.
    private Kompl2 prvi, drugi, rezult;              // Operandi i rezultat.
    private final String[] oper =                   // Natpisi dugmadi.
        {"Zbir", "Razlika", "Proizvod", "Kolicnik"};

    private void popuniProzor () {                    // Popunjavanje prozora:
        Panel plo = new Panel(new GridLayout (3, 1)); // - ploča za oznake
        add (plo, "West");                           // podataka,
        plo.add (new Label ("Prvi"));
        plo.add (new Label ("Drugi"));
        plo.add (rez = new Label ());

        add (plo = new Panel (new GridLayout (3, 2)), "Center"); // - ploča za
        try {                                         // podatke,
            TextField re = new TextField (), im = new TextField ();
            plo.add (re); plo.add (im);
            prvi = new Kompl2 (new Polje(re), new Polje (im));
            re = new TextField (); im = new TextField ();
            plo.add (re); plo.add (im);
            drugi = new Kompl2 (new Polje(re), new Polje (im));
            Label r = new Label (), i = new Label ();
            plo.add (r); plo.add (i);
            rezult = new Kompl2 (new Polje(r), new Polje (i));
        } catch (GKompNeOdgovara g) {}
    }
}

```

```

    add (plo = new Panel (), "South");           // - ploča za dugmad.
    DugmeAkcija osmatrac = new DugmeAkcija ();
    for (String s: oper) {
        Button dugme = new Button (s);
        plo.add (dugme);
        dugme.addActionListener (osmatrac);
    }
} // popuniProzor()

private class DugmeAkcija implements ActionListener { // Obrada dogadaja
    public void actionPerformed (ActionEvent d) {          // dugmadi.
        String op = ((Button) (d.getSource ())).getLabel ();
        try {
            if      (op.equals(oper[0]))
                rezult.postavi (prvi.uzmi () .zbir      (drugi.uzmi ()));
            else if (op.equals(oper[1]))
                rezult.postavi (prvi.uzmi () .razlika  (drugi.uzmi ()));
            else if (op.equals(oper[2]))
                rezult.postavi (prvi.uzmi () .proizvod (drugi.uzmi ()));
            else if (op.equals(oper[3]))
                rezult.postavi (prvi.uzmi () .kolicnik (drugi.uzmi ()));
            rez.setText (op); rez.setForeground (Color.BLACK);
        } catch (NumberFormatException g) {
            rez.setText ("GRESKA"); rez.setForeground (Color.RED);
            validate();
        }
    }
} // class DugmeAkcija

private PoljeT () {                                // Inicijalizacija:
    super ("Kompleksna aritmetika");
    setBounds (100, 100, 300, 150); setResizable (false);
    popuniProzor ();                               // - popunjavanje prozora,
    addWindowListener (new WindowAdapter () { // - obrada dogadaja prozora.
        public void windowClosing (WindowEvent d) { dispose (); }
    });
} // PoljeT()

public static void main (String[] varg)           // GLAVNA FUNKCIJA.
    { new PoljeT ().setVisible (true); }

} // class PoljeT

```

### Zadatak 8.6 Uređivanje niza celih brojeva

Koristeći klase za uređivanje nizova iz zadatka 5.9 napisati na jeziku Java program s grafičkom korisničkom površi za uređivanje nizova celih brojeva.

Rešenje:

```

// Uredi4.java - Uređivanje
// nizova celih brojeva u
// grafičkom okruženju.

import java.awt.*;
import java.awt.event.*;
import uredjivaci.*;      => Zadatak
import usluge.Uporediv;   => 5.9

public class Uredi4 extends Frame {

    private static int[] duzine =           // Moguće dužine niza.
        {100, 200, 500, 1000, 2000, 5000, 10000};
    private static Uredjivac[] uredjivaci = {          // Algoritmi uređivanja.
        new MetodaIzbora (),      new MetodaIzbora2 (),
        new MetodaUmetanja (),     new MetodaUmetanja2 (),
        new MetodaZameneSuseda (), new MetodaPodele ()};

    private Uporediv[] niz = new Ceo [100];           // Niz za uređivanje.
    private Uredjivac uredjivac = uredjivaci[0];       // Trenutni uređivač.
    private TextArea prikaz = new TextArea ();         // Polje za prikaz niza.

    private void puni () {                           // Punjenje niza slučajnim brojevima.
        for (int i=0; i<niz.length; i++)
            niz[i] = new Ceo ((int)(Math.random ()*10));
        prikazi ();      => Zadatak 5.9
    }

    private void prikazi () {                      // Prikazivanje sadržaja niza.
        int k = (prikaz.getWidth () - 20) / 10; String s = "";
        for (int i=0; i<niz.length; i++)
            s += niz[i] + ((k==0 || i==k-1 || i==niz.length-1) ? "\n" : " ");
        prikaz.setText (s);
    }

    private void popuniProzor () {                  // Popunjavanje prozora.
        // - višeredno polje za tekst za prikazivanje niza,
        prikaz.addComponentListener (new ComponentAdapter () {
            public void componentResized (ComponentEvent d) { prikazi (); }
        });
        puni (); add (prikaz, "Center");

        Panel ploca = new Panel (new GridLayout (0, 1)); add (ploca, "East");
    }
}

```

```

Choice duzina = new Choice (); // - padajuća
for (int d: duzine) duzina.addItem (Integer.toString (d)); // lista za
duzina.addItemListener (new ItemListener () { // izbor
    public void itemStateChanged (ItemEvent d) { // dužine,
        int duz = Integer.parseInt
            . ((Choice)d.getSource ()).getSelectedItem ();
        niz = new Ceo [duz]; puni ();
    }
});
ploca.add (duzina);

CheckboxGroup grupa = new CheckboxGroup (); // - radio-dugmad za
RadioPromena osmatrac = new RadioPromena (); // izbor algoritma.
for (int i=0; i<uredjivaci.length; i++) {
    Checkbox radio = new Checkbox (uredjivaci[i].toString (),
        grupa, i==0);
    radio.addItemListener (osmatrac); ploca.add (radio);
}

private class RadioPromena implements ItemListener { // Obrada promene
    public void itemStateChanged (ItemEvent d) { // stanja radio-
        String naziv = ((Checkbox)d.getSource ()).getLabel (); // -dugmeta.
        int i; for (i=0; i<uredjivaci.length; i++)
            if (naziv.equals (uredjivaci[i].toString ()) ) break;
        uredjivac = uredjivaci[i].uredi (niz); prikazi ();
    }
}

private void dodajMeni () { // Sastavljanje menija.
    MenuBar traka = new MenuBar (); setMenuBar (traka);
    Menu meni = new Menu ("Akcija"); traka.add (meni);
    MenuItem stavka = new MenuItem ("Pravi", new MenuShortcut ('P'));
    stavka.addActionListener (new ActionListener () {
        public void actionPerformed (ActionEvent d) { puni (); }
    });
    meni.add (stavka);
    stavka = new MenuItem ("Radi", new MenuShortcut ('R'));
    stavka.addActionListener (new ActionListener () {
        public void actionPerformed (ActionEvent d)
            { uredjivac.uredi (niz); prikazi (); }
    });
    meni.add (stavka);
    meni.addSeparator();
    stavka = new MenuItem ("Zavrsi", new MenuShortcut ('Z'));
    stavka.addActionListener (new ActionListener () {
        public void actionPerformed (ActionEvent d) { dispose (); }
    });
    meni.add (stavka);
}

private Uredi4 () { // Inicijalizacija.
    super ("Uredjivanje nizova"); setBounds (100, 100, 400, 200);
    popuniProzor (); dodajMeni (); setVisible (true);
    addWindowListener (new WindowAdapter () {
        public void windowClosing (WindowEvent d) { dispose (); }
    });
}

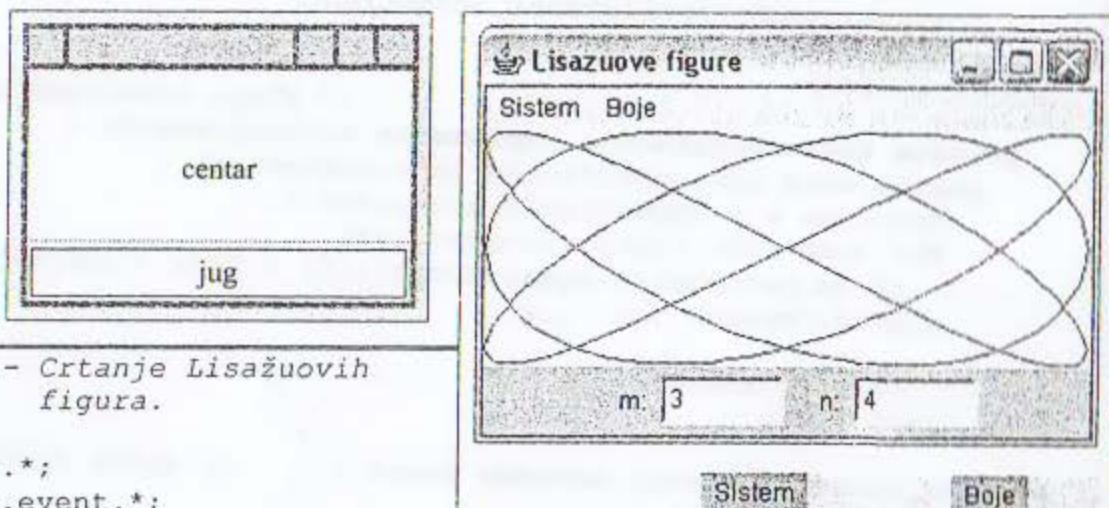
public static void main (String[] vpar) { new Uredi4 (); }
}

```

### Zadatak 8.7 Crtanje Lisažuovih figura

Napisati na jeziku Java program s grafičkom korisničkom površi za crtanje Lisažuovih (*Lissajous*) figura  $y(x) = a \cos m\phi, y(\phi) = b \sin n\phi$  ( $0 \leq \phi \leq 2\pi$ ), gde su  $m$  i  $n$  relativno prosti celi brojevi.

Rešenje:



```
// Lisazu.java - Crtanje Lisažuovih
// figura.

import java.awt.*;
import java.awt.event.*;

public class Lisazu extends Frame {

    private static final int N = 300;      // Broj tačaka za crtanje.
    private static final String[] boje = // Moguće boje.
        { "Crna", "Siva", "Plava", "Zelena", "Zuta", "Crvena" };
    private static final Color[] cboje = { Color.BLACK, Color.GRAY,
        Color.BLUE, Color.GREEN, Color.YELLOW, Color.RED };
    private Color boja = Color.RED;           // Boja u upotrebi.
    private int m = 3, n = 4;                 // Učestanost po x i y osi.
    private TextField tksM = new TextField (""+m, 5), // Izbor učestanosti.
                  tksN = new TextField (""+n, 5);
    private Platno platno = new Platno ();   // Platno za crtanje.

    private class Platno extends Canvas {      // KLASA PLATNA ZA CRTANJE.
        public void paint (Graphics g) {
            int a = (getWidth() - 1) / 2, b = (getHeight() - 1) / 2;
            g.translate (a, b);
            g.setColor (boja);
            int xl = a, yl = 0;
            for (int i=0; i<=N; i++) {
                double fi = 2 * Math.PI / N * i;
                int x2 = (int)(a * Math.cos (m * fi)),
                    y2 = (int)(b * Math.sin (n * fi));
                g.drawLine (xl, yl, x2, y2);
                xl = x2; yl = y2;
            }
        }
    } // class Platno

    private class Traka extendsMenuBar {        // KLASA TRAKE MENIJA.
        public Traka () {
            Menu meni = new Menu ("Sistem"); add (meni);
            MenuItem stavka = new MenuItem ("Zavrsi"); meni.add (stavka);
            stavka.setShortcut (new MenuShortcut ('Z'));
        }
    }
}
```

```

stavka.addActionListener (new ActionListener () {
    public void actionPerformed (ActionEvent d) { dispose (); }
});
add (meni = new Menu ("Boje"));
BojaAkcija osmatrac = new BojaAkcija ();
for (String b: boje) {
    meni.add (stavka = new MenuItem (b));
    stavka.addActionListener (osmatrac);
}
} // Traka()                                // Klasa osmatrača za izbor boje.

private class BojaAkcija implements ActionListener {
    public void actionPerformed (ActionEvent d) {
        MenuItem m = (MenuItem)d.getSource ();
        for (int i=0; i<boje.length; i++)
            if (m.getLabel ().equals (boje[i])) { boja = cboje[i]; break; }
        platno.repaint ();
    }
} // class BojaAkcija
} // class Traka

private class Parametri extends Panel {      // KLASA PLOČE ZA PARAMETRE.
    public Parametri () {
        setBackground (new Color(192, 192, 255));
        add (new Label ("m:", Label.RIGHT)); add (tksM);
        add (new Label ("n:", Label.RIGHT)); add (tksN);
        TextListener osluskivac = new TextListener () {
            public void textValueChanged (TextEvent d) {
                try {
                    m = Integer.parseInt (tksM.getText ());
                    n = Integer.parseInt (tksN.getText ());
                } catch (NumberFormatException g) {
                } finally { platno.repaint (); }
            }
        };
        tksM.addTextListener (osluskivac);
        tksN.addTextListener (osluskivac);
    } // Parametri ()
} // class Parametri

private Lisazu () {                         // Konstruktor klase Lisazu.
    super ("Lisazuove figure");
    setSize (300, 200);
    setLocationRelativeTo (null);
    add (platno, "Center");
    add (new Parametri (), "South");
    setMenuBar (new Traka ());
    setVisible (true);
    addWindowListener (new WindowAdapter () {
        public void windowClosing (WindowEvent d) { dispose (); }
    });
} // Lisazu()

public static void main (String[] varg)          // GLAVNA FUNKCIJA.
    { new Lisazu (); }
} // class Lisazu.

```

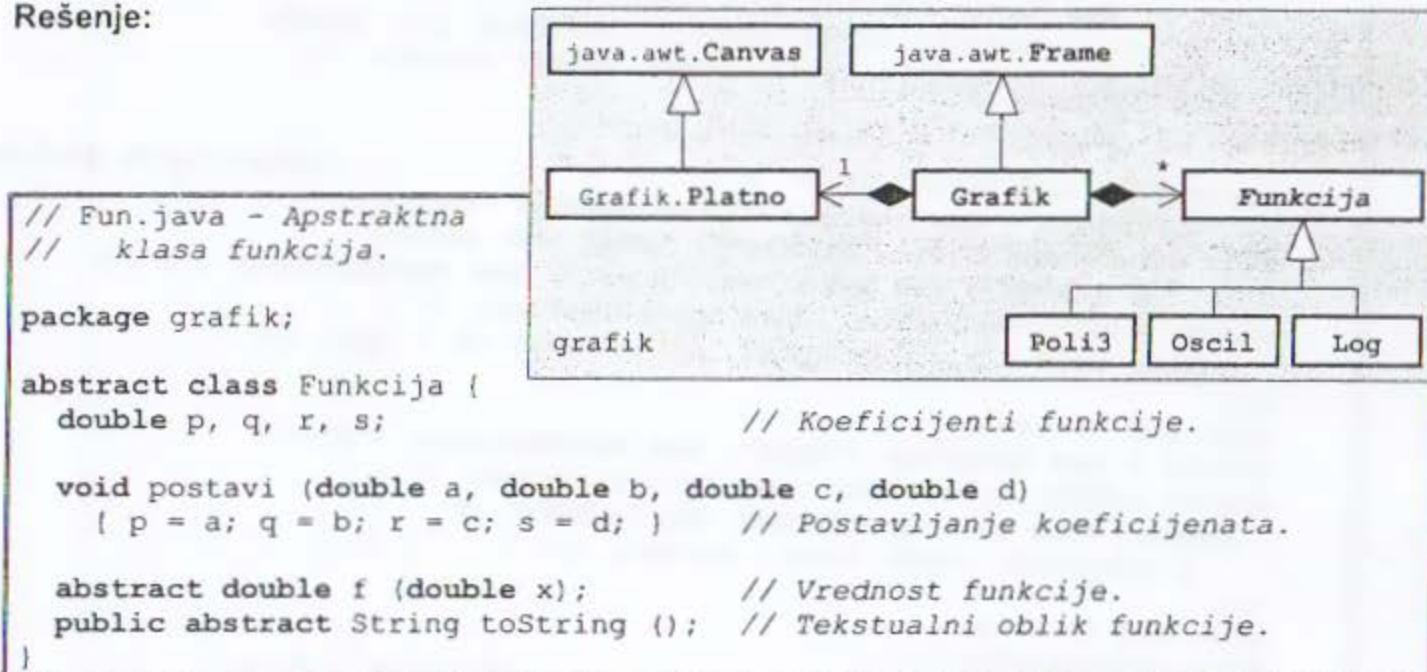
### Zadatak 8.8 Apstraktne funkcije, polinomi, oscilacije, logaritmi i grafici funkcija

Napisati na jeziku Java sledeće klase:

- Apstraktna *funkcija* ima četiri koeficijenta ( $p, q, r$  i  $s$ ) koji mogu da utiču na izračunatu vrednost funkcije. Mogu da se postave vrednosti koeficijenata, da se izračuna realna vrednost funkcije za datu vrednost nezavisne promenljive  $x$  i da se sastavi tekstualni oblik funkcije.
- *Polinom* trećeg reda je funkcija koja je zadata izrazom  $px^3 + qx^2 + rx + s$  i čiji je tekstualni oblik " $px^3 + qx^2 + rx + s$ ".
- Prigušena *oscilacija* je funkcija koja je zadata izrazom  $p e^{qx} \sin(rx+s)$  i čiji je tekstualni oblik " $p e^{(qx)} \sin(rx+s)$ ".
- *Logaritam* je funkcija koja je zadata izrazom  $p + q \ln x$  i čiji je tekstualni oblik " $p + q \ln x$ ".
- *Grafik* na platnu u prozoru korisničke grafičke površi crta grafike funkcija koje sadrži. Može da se bira prikazani opseg vrednosti duž  $x$  i  $y$  osa, da se odabere funkcija čiji se grafik crta, da se postave vrednosti koeficijenata  $p, q, r$  i  $s$  funkcije i da se funkcija nacrti. Dok se miš pomera iznad platna grafika, na dva natpisa se prikazuju vrednost nezavisne promenljive  $x$  koja odgovara trenutnom položaju pokazivača miša i odgovarajuća izračunata vrednost funkcije.

Napisati na jeziku Java program za ispitivanje prethodnih klasa.

Rešenje:



*// Poli3.java - Klasa polinoma trećeg reda.*

```

package grafik;
class Poli3 extends Funkcija {

    double f (double x) // Vrednost polinoma.
        { return ((p * x + q) * x + r) * x + s; }

    public String toString () // Tekstualni oblik.
        { return "px^3+qx^2+rx+s"; }
}
  
```

```
// Oscil.java - Klasa prigušenih oscilacija.

package grafik;

class Oscil extends Funkcija {

    double f (double x)           // Vrednost funkcije.
    { return p * Math.exp (q*x) * Math.sin (r*x+s); }

    public String toString ()      // Tekstualni oblik.
    { return "p e^(qx) sin(rx+s)"; }
}
```

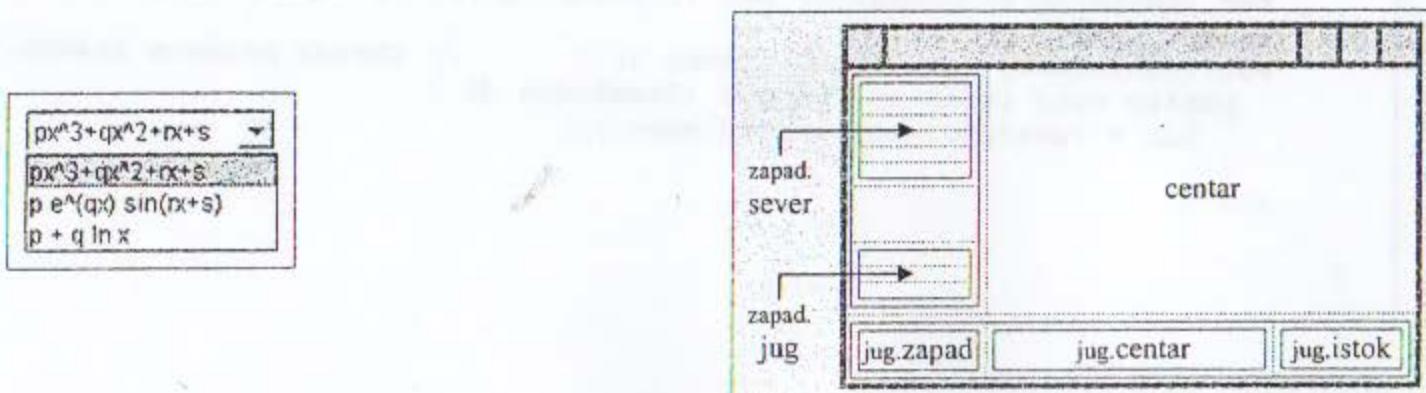
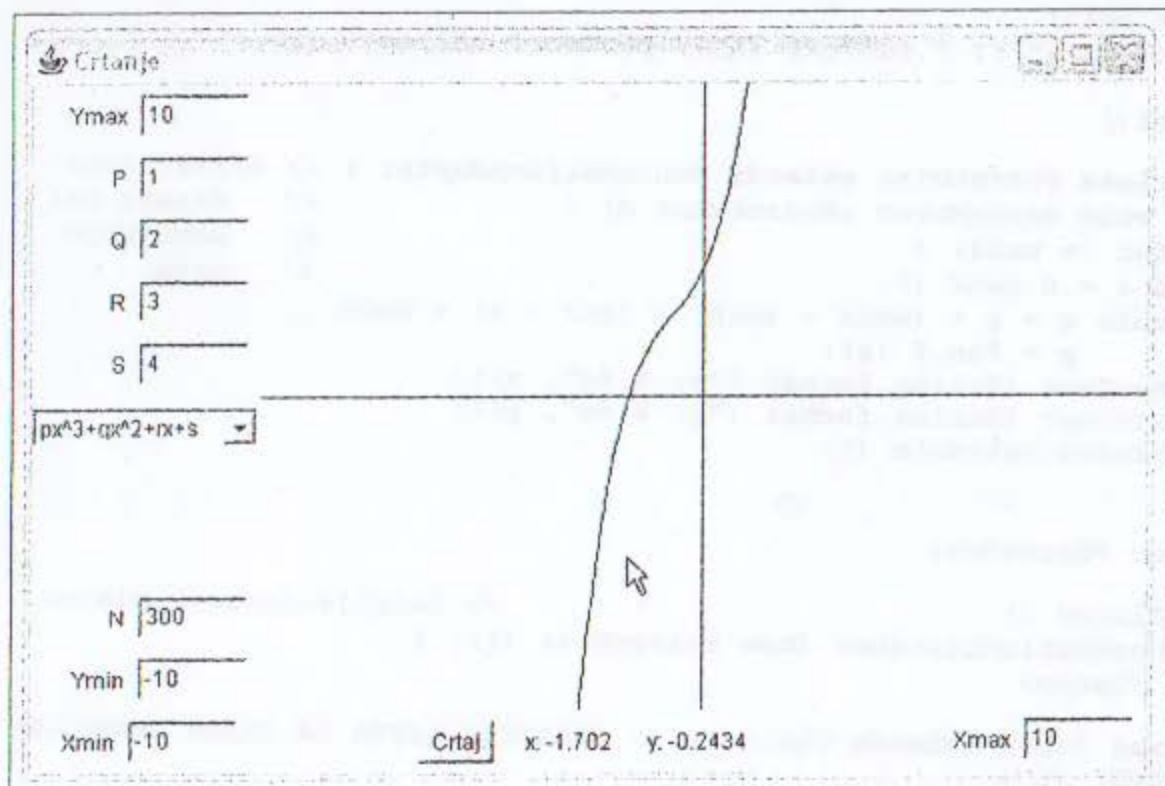
```
// Log.java - Klasa logaritamskih funkcija.

package grafik;

class Log extends Funkcija {

    double f (double x)           // Vrednost funkcije.
    { return p + q * Math.log (x); }

    public String toString ()      // Tekstualni oblik.
    { return "p + q ln x"; }
}
```



```

// Grafik.java - Crtanje grafika funkcija.

package grafik;
import java.awt.*; import java.awt.event.*; import java.util.*;
public class Grafik extends Frame {
    private Funkcija[] funkcije = // Postojeće funkcije.
        { new Poli3(), new Oscil(), new Log() };
    private Funkcija fun = funkcije[0]; // Funkcija koja se koristi.
    private Platno platno = new Platno(); // Platno po kome se crta.
    private class DogadjajiTastera extends KeyAdapter { // Ponovno crtanje
        public void keyPressed (KeyEvent d) { // pritiskom na
            if (d.getKeyCode () == KeyEvent.VK_ENTER) // 'enter'.
                platno.repaint ();
        }
    }
    private class DogadjajiZize extends FocusAdapter { // Obeležavanje
        public void focusGained (FocusEvent d) { // teksta pri
            ((TextComponent)d.getComponent()).selectAll (); // dobijanju žiže.
        }
    }
    private class Param extends Panel { // PLOČA ZA UNOS PARAMETRA:
        private TextField par = new TextField (5); // - polje za vrednost,
        { // - inicializacioni blok,
            setLayout(new FlowLayout (FlowLayout.RIGHT));
            par.addKeyListener (new DogadjajiTastera());
            par.addFocusListener (new DogadjajiZize());
        }
        public Param (String natpis, double vr) { // - inicializacija
            add (new Label (natpis, Label.RIGHT)); // realnim brojem,
            add (par);
            par.setText (Double.toString (vr));
        }
        public Param (String natpis, int vr) { // - inicializacija
            add (new Label (natpis, Label.RIGHT)); // celim brojem,
            add (par);
            par.setText (Integer.toString (vr));
        }
        public double Double () { // - dohvatanje kao
            return Double.parseDouble (par.getText()); // realan broj,
        }
        public int Int () { // - dohvatanje kao
            return Integer.parseInt (par.getText()); // ceo broj.
        }
    } // class Param // Ploče za parametre:
    private Param Xmin = new Param ("Xmin", -10), // - opseg x-a,
        Xmax = new Param ("Xmax", 10),
        Ymin = new Param ("Ymin", -10), // - opseg y-a,
        Ymax = new Param ("Ymax", 10),
        N = new Param ("N", 300), // - broj tačaka,
        P = new Param ("P", 1), // - koeficijenti za
        Q = new Param ("Q", 2), // funkcije.
        R = new Param ("R", 3),
        S = new Param ("S", 4);
}

```

```

    {
        Color boja = new Color (255, 240, 240);           // Postavljanje boje
        Xmax.setBackground (boja);                         // podloge ploča za
        Xmin.setBackground (boja);                         // opsege koordinata.
        Ymax.setBackground (boja);
        Ymin.setBackground (boja);
    }

    private double xmin, xmax, ymin, ymax;             // Parametri crtanja.
    private int sir, vis, n;                           // Koeficijenti funkcija.
    private double p, q, r, s;                         // Uzimanje parametara.

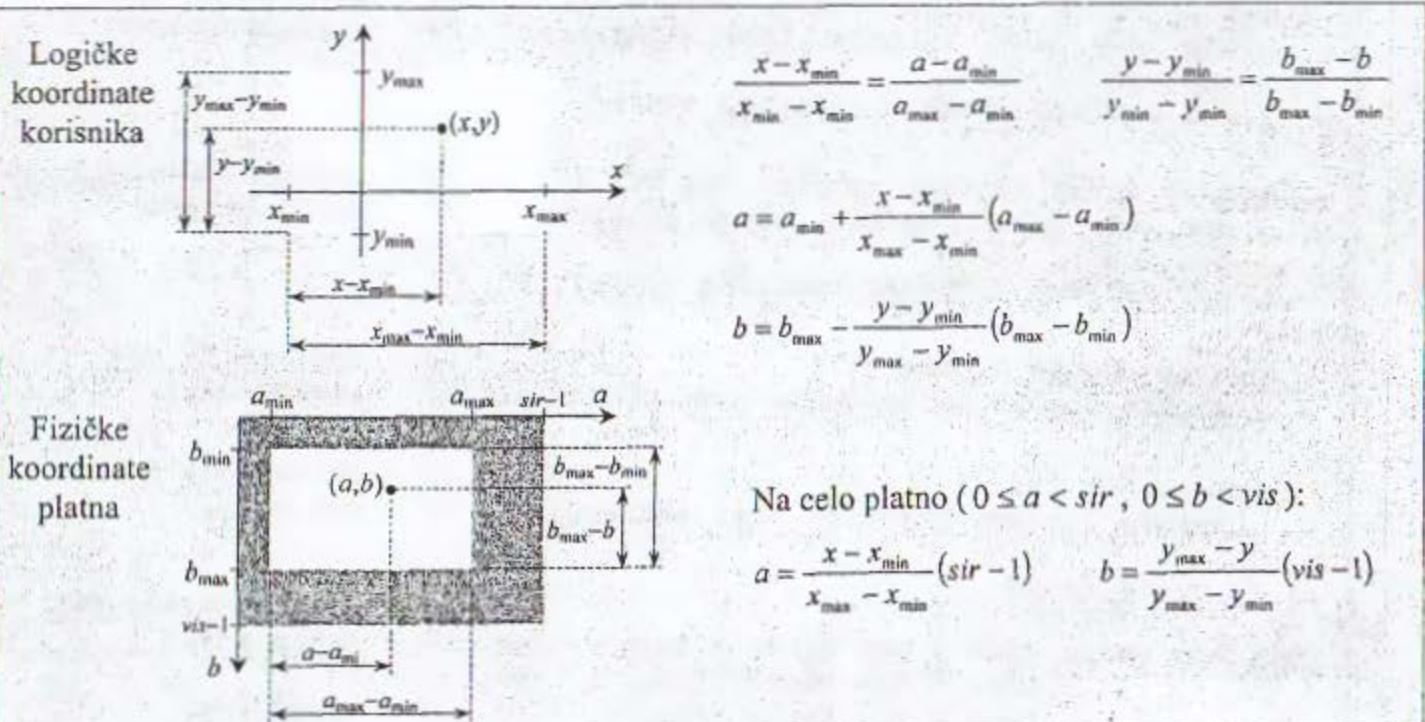
    private void uzmiParametre () {
        xmin = Xmin.Double();      xmax = Xmax.Double();
        ymin = Ymin.Double();      ymax = Ymax.Double();
        sir = platno.getWidth();   vis = platno.getHeight();
        n = N.Int ();
        p = P.Double(); q = Q.Double(); r = R.Double(); s = S.Double();
    }

    private Label X = new Label ("",Label.RIGHT), // Natpisi za prikaz
                 Y = new Label ("",Label.RIGHT); // koordinata.
    Panel ploKoord = new Panel ();                // Ploča za prikaz koordinata.

    private Label greska = new Label ("GRESKA!", Label.CENTER); // Natpis za
    {                                                 // grešku.
        greska.setForeground (Color.RED);
        greska.setFont (new Font (null, Font.BOLD, 20));
        greska.setVisible (false);
    }

    private class Platno extends Canvas {           // PLATNO PO KOME SE CRTA.

```



```

    private int a (double x)                      // Preslikavanje koordinate x.
    { return (int) ((x-xmin) / (xmax-xmin) * (sir-1)); }

    private int b (double y)                      // Preslikavanje koordinate y.
    { return (int) ((ymax-y) / (ymax-ymin) * (vis-1)); }

```

```

public void paint (Graphics g) { // Crtanje krive:
    try {
        greska.setVisible (false); // - uklanjanje poruke o grešci,
        g.setColor (new Color(240,240,240)); // - popunjavanje podloge,
        g.fillRect (0,0,getWidth()-1,getHeight()-1);
        uzmiParametre (); // - uzimanje parametara,
        fun.postavi (p, q, r, s); // - postavljanje koeficijenata,
        g.setColor (Color.RED); // - crtanje koordinatnih osa,
        g.drawLine (0, b(0), sir-1, b(0));
        g.drawLine (a(0), 0, a(0), vis-1);
        double dx = (xmax - xmin) / n; // - kوتak duž x-ose,
        int a0 = 0, b0 = 0;
        boolean prva = true;
        g.setColor (Color.BLACK);
        for (double x=xmin; x<=xmax; x+=dx) { // - crtanje funkcije,
            double y = fun.f (x);
            if (! Double.isNaN (y)) {
                int a = a (x), b = b (y);
                if (! prva) g.drawLine (a0, b0, a, b); else prva = false;
                a0 = a; b0 = b;
            } else prva = true;
        }
    } catch (NumberFormatException gr) { // - obrada greške.
        greska.setVisible (true);
        fun = null;
        X.setText (""); Y.setText (""); ploKoord.validate ();
    }
} // paint()

private class PokretMisa extends MouseMotionAdapter { // Prikaz koor-
    public void mouseMoved (MouseEvent d) { // dinata pri
        if (fun != null) { // pomeranju
            int a = d.getX ();
            double x = a * (xmax - xmin) / (sir - 1) + xmin,
            y = fun.f (x);
            X.setText (String.format ("x: %.4g", x));
            Y.setText (String.format ("y: %.4g", y));
            ploKoord.validate ();
        }
    }
} // class PokretMisa

private Platno () // Inicijalizacija platna.
{ addMouseMotionListener (new PokretMisa ());
} // class Platno;

private class Izbor extends Choice { // PADAJUĆA LISTA ZA IZBOR FUNKCIJE.
    public Izbor () {
        for (Funkcija f: funkcije) add (f.toString()); // Popunjavanje liste.
        select (0);
        addItemListener (new ItemListener () { // Obrada promene stavke.
            public void itemStateChanged (ItemEvent d) {
                fun = funkcije [getSelectedIndex()];
            }
        });
    }
}

```

```

private void popuniProzor () { // Popunjavajne prozora:
    add (platno, "Center"); // - platno za crtanje,
    add (ploca, "West"); // - zapadna ploča za
    ploca.setBackground (new Color (255, 255, 240)); // unos podataka,
    Panel plo = new Panel (new GridLayout (0,1));
    ploca.add (plo, "North");
    plo.add (Ymax); plo.add (P); plo.add (Q); plo.add (R); plo.add (S);
    plo.add (new Izbor ());
    plo = new Panel (new GridLayout (0,1)); ploca.add (plo, "South");
    plo.add (greska); plo.add (N); plo.add (Ymin);

    ploca = new Panel (new BorderLayout()); // - južna ploča za unos
    add (ploca, "South"); // i prikaz podataka,
    ploca.setBackground (new Color (255, 255, 240));
    Button crtaj = new Button ("Crtaj");
    crtaj.addActionListener (new ActionListener () {
        public void actionPerformed (ActionEvent d) { platno.repaint (); }
    });
    ploca.add (Xmin, "West"); ploca.add (Xmax, "East");
    ploca.add (ploKoord, "Center");
    ploKoord.add (crtaj); ploKoord.add (X); ploKoord.add (Y);
} // popuniProzor()

public Grafik () { // Inicijalizacija.
    super ("Crtanje");
    setSize (600, 400); setLocationRelativeTo (null);
    popuniProzor ();
    setVisible (true);
    addWindowListener (new WindowAdapter () {
        public void windowClosing (WindowEvent d) { dispose (); }
    });
} // Grafik()

} // class Grafik

```

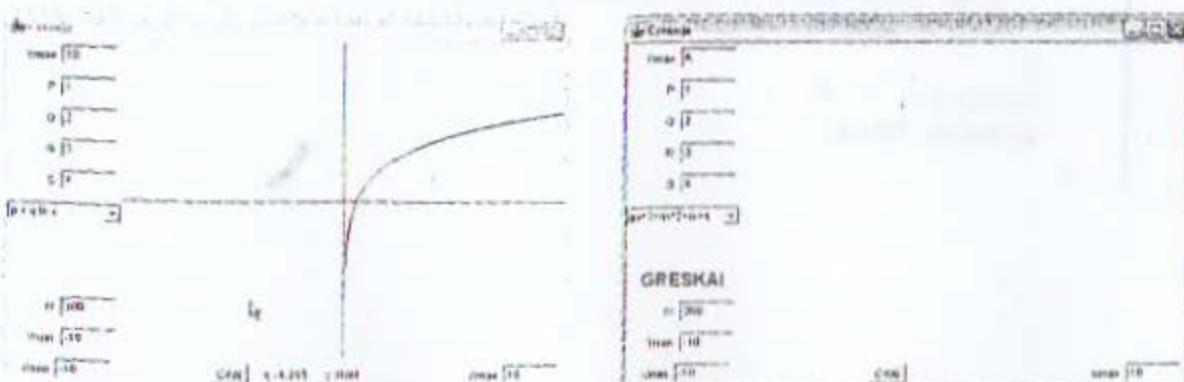
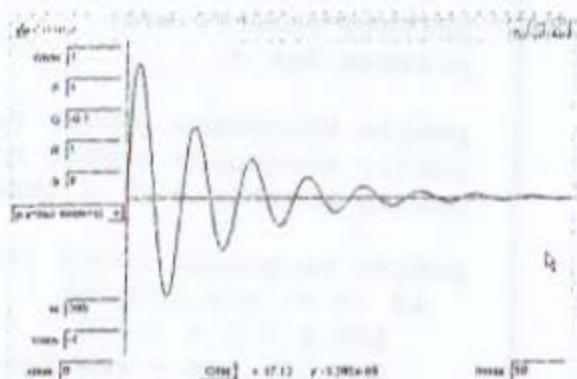
```

// GrafikT.java - Ispitivanje klase za
// crtanje grafika funkcije.

import grafik.Grafik;

public class GrafikT {
    public static void main (String[] varg)
        ( new Grafik (); )
}

```



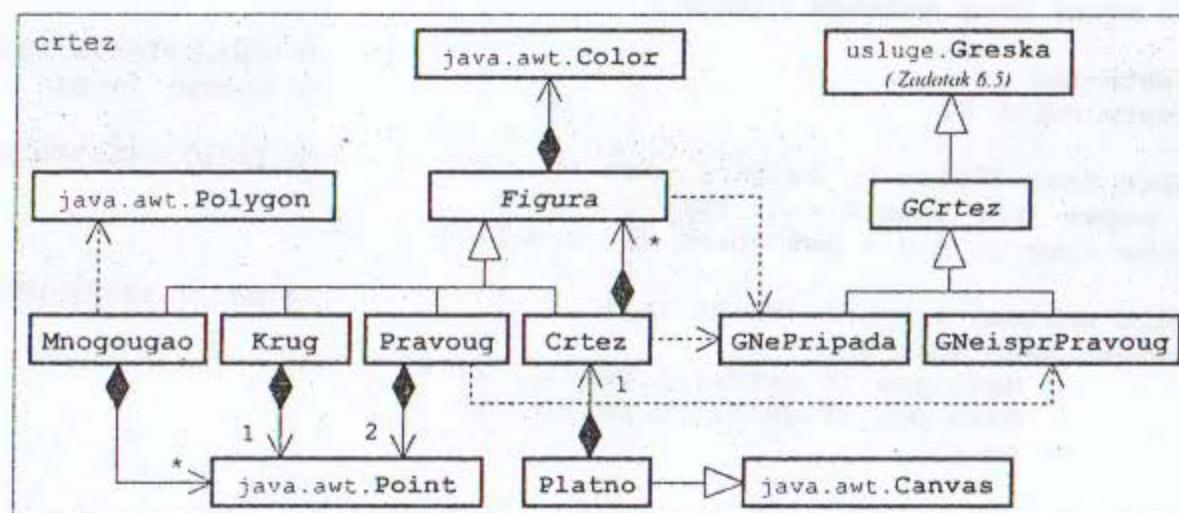
### Zadatak 8.9 Popunjene figure, krugovi, pravougaonici, mnogouglovi, crteži i platna

Koristeći standardnu podršku za rad s grafičkim elementima napisati na jeziku Java sledeće klase:

- Apstraktna popunjena *figura* u ravni ima zadatu boju (podrazumevano crnu). Može da se ispita da li neka tačka pripada figuri, da se dohvati boja figure i boja zadate tačke u figuri i da se figura iscrtava na zadatom grafičkom objektu. Greška je ako tačka, čija se boja traži, ne pripada figuri. Iscrtavanje figure na grafičkom objektu se vrši relativno u odnosu na zadatu tačku (podrazumevano u odnosu na koordinatni početak). Usmerenje *x*-ose je udesno, a *y*-ose nagore (napomena: kod standardnih grafičkih objekata koordinatni početak je u gornjem levom uglu i *y*-osa je usmerena nadole).
- *Krug* je popunjena figura zadatog poluprečnika (podrazumevano 1) sa centrom u zadatoj tački (podrazumevano (0,0)).
- *Pravougaonik*, sa ivicama paralelnim koordinatnim osama, je popunjena figura zadata donjim levinim (podrazumevano (0,0)) i gornjim desnim (podrazumevano (1,1)) temenom.
- *Mnogougao* je popunjena figura zadata nizom temena. Stvara se prazan zadatog kapaciteta (podrazumevano 5), posle čega može da se dodaje proizvoljan broj temena. Po potrebi kapacitet se povećava za 10%, ali najmanje za 5 mesta.
- *Crtež* je pravougaona popunjena figura čije su ivice paralelne koordinatnim osama. Može da sadrži proizvoljan broj figura. Zadaje se donjim levim temenom, širinom i visinom. Crtež se stvara prazan, posle čega se figure dodaju jedna po jedna. Koordinate dodatih figura se računaju relativno u odnosu na donji levi ugao crteža. Delovi dodatih figura, koji se nalaze izvan crteža, se ne iscrtavaju. Kasnije dodata figura pokriva ranije dodate figure. Dimenzije crteža mogu da se promene, pri čemu sadržane figure ostaju nepromjenjene u odnosu na crtež.
- *Platno* je grafički objekat koji sadrži jedan crtež koji pokriva celu površinu platna. Stvara se s praznim crtežom, posle čega mogu da se dodaju figure. Može da se dohvati boja zadate tačke platna.

Napisati na jeziku Java program s grafičkom korisničkom površi za ispitivanje prethodnih klasa.

**Rešenje:**



```
// Figura.java - Apstraktna klasa figura.

package crtez;
import java.awt.*;

public abstract class Figura {
    private Color boja;                                // Boja figure.

    public Figura (int c, int z, int p)                // Inicijalizacija.
        { boja = new Color (c, z, p); }
    public Figura (Color b) { boja = b; }
    public Figura () { boja = Color.black; }

    public abstract boolean pripada (Point T);         // Da li tačka pripada?
    public final Color boja () { return boja; }          // Boja figure.

    public Color boja (Point T) throws GNePripada {   // Boja tačke u figuri.
        if (! pripada (T)) throw new GNePripada ();
        return boja;
    }

    public abstract void crtaj (Graphics g, int x0, int y0); // Crtanje
    public void crtaj (Graphics g) { crtaj (g, 0, 0); }      // figure.
}
```

```
// Krug.java - Klasa krugova.

package crtez;
import java.awt.*;

public class Krug extends Figura {
    private int r;                                     // Poluprečnik kruga.
    private Point C;                                   // Centar kruga.

    public Krug (Color b, Point C, int r)             // Inicijalizacija.
        { super (b); this.C = C; this.r = r; }
    public Krug () { C = new Point (); r = 1; }

    public boolean pripada (Point T) {                  // Da li tačka pripada?
        return Math.sqrt (
            Math.pow (T.getX()-C.getX(), 2) +
            Math.pow (T.getY()-C.getY(), 2)
        ) <= r;
    }

    public void crtaj (Graphics g, int x0, int y0) { // Crtanje kruga.
        g.setColor (boja ());
        g.fillOval (x0+(int)C.getX()-r, y0-(int)C.getY()-r, 2*r, 2*r);
    }
}
```

```

// Pravoug.java - Klasa pravougaonika.

package crtez;
import java.awt.*;

public class Pravoug extends Figura {
    private Point A, C;                                // Naspramna temena.
                                                       // Inicijalizacija.
    public Pravoug (Color b, Point A, Point C) throws GNeisprPravoug {
        super (b);
        if (A.getX() >= C.getX() || A.getY() >= C.getY())
            throw new GNeisprPravoug ();
        this.A = A; this.C = C;
    }
    public Pravoug () { A = new Point (); C = new Point (1, 1); }

    public boolean pripada (Point T) {                  // Da li tačka pripada?
        return A.getX() <= T.getX() && T.getX() <= C.getX() &&
               A.getY() <= T.getY() && T.getY() <= C.getY();
    }

    public void crtaj (Graphics g, int x0, int y0) { // Crtanje pravougaonika.
        g.setColor (boja ());
        g.fillRect (x0+(int)A.getX(), y0-(int)C.getY(),
                    (int)(C.getX()-A.getX()), (int)(C.getY()-A.getY()));
    }
}

```

```

// Mnogougao.java - Klasa mnogouglova.

package crtez;
import java.awt.*;

public class Mnogougao extends Figura {
    private Point[] niz;                                // Temena mnogougla.
    private int n;                                     // Broj temena.
                                                       // Inicijalizacija.
    public Mnogougao (Color b, int kap) { super (b); niz = new Point [kap]; }
    public Mnogougao (Color b) { this (b, 5); }
    public Mnogougao () { this (Color.black, 5); }

    public Mnogougao dodaj (Point T) {                  // Dodavanje temena.
        if (n == niz.length) {
            int k = n + (n<50 ? 5 : n/10);
            Point[] pom = new Point [k];
            for (int i=0; i<n; pom[i]=niz[i++]);
            niz = pom;
        }
        niz[n++] = T;
        return this;
    }
}

```

```

public boolean pripada (Point T) { // Da li tačka pripada?
    Polygon p = new Polygon ();
    for (int i=0; i<n; i++)
        p.addPoint ((int)niz[i].getX(), (int)niz[i].getY());
    return p.contains (T.getX(), T.getY());
}

public void crtaj (Graphics g, int x0, int y0) { // Crtanje mnogougla.
    Polygon p = new Polygon ();
    for (int i=0; i<n; i++)
        p.addPoint (x0+(int)niz[i].getX(), y0-(int)niz[i].getY());
    g.setColor (boja ());
    g.fillPolygon (p);
}
}

```

```

// Crtez.java - Klasa crteža.

package crtez;
import java.awt.*;

public class Crtez extends Figura {

    private Elem prvi, posl; // Prvi i poslednji element liste figura.

    private class Elem { // Element liste figura:
        Figura fig; // - sadržana figura,
        Elem pret, sled; // - prethodni i sledeći element,
        Elem (Figura f) { // - inicijalizacija.
            fig = f;
            sled = null; pret = posl;
            if (prvi == null) prvi = this; else posl.sled = this;
            posl = this;
        }
    }

    private int x, y; // Donje levo teme crteža.
    private int sir, vis; // Širina i visina crteža.

    public Crtez (Color b, Point T, int s, int v) // Inicijalizacija.
        { super (b); x = (int)T.getX(); y = (int)T.getY(); sir = s; vis = v; }

    public Crtez promeniDim (int s, int v) // Promena dimenzija crteža.
        { sir = s; vis = v; return this; }

    public Crtez dodaj (Figura f) // Dodavanje figure.
        { new Elem (f); return this; }

    public boolean pripada (Point T) { // Da li tačka pripada?
        int dx = (int)T.getX () - x, dy = (int)T.getY () - y;
        return dx >= 0 && dx <= sir && dy >= 0 && dy <= vis;
    }
}

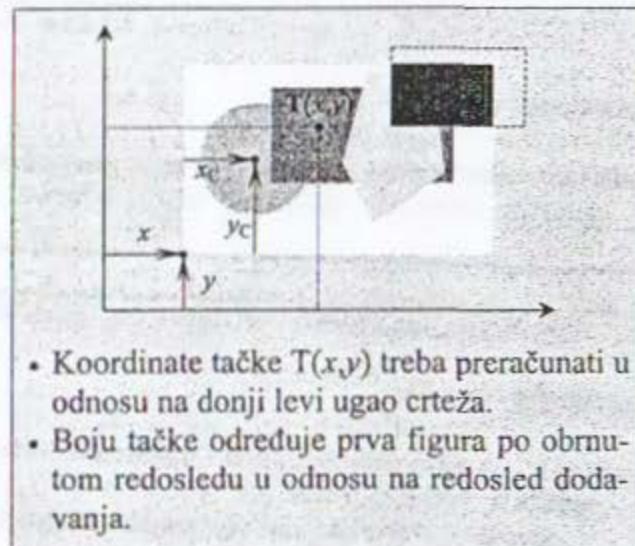
```

```

// Boja tačke.
public Color boja (Point T)
throws GNePripada {
if (! pripada (T))
throw new GNePripada ();
Point T2 = new Point (T);
T2.translate (-x, -y);
for (Elem tek=posl; tek!=null;
tek=tek.pret)
try { return tek.fig.boja (T2); }
catch (GNePripada g) {}
return boja ();
}

public void crtaj (Graphics g, int x0, int y0) { // Crtanje crteža.
Rectangle staro = g.getClipBounds ();
g.clipRect (x0+x, y0-y-vis, sir, vis);
g.setColor (boja ());
g.fillRect (x0+x, y0-y-vis, sir, vis);
for (Elem tek=prvi; tek!=null; tek=tek.sled)
tek.fig.crtaj (g, x0+x, y0-y);
g.setClip (staro);
}
}

```



```

// Platno.java - Klasa platna za crtanje.

package crtez;
import java.awt.*;
import java.awt.event.*;

public class Platno extends Canvas {
    private Crtez crt = new Crtez (Color.yellow, new Point(0,getHeight()), 0, 0); // Crtež na platnu.

    public Platno () { // Inicijalizacija.
        addComponentListener (new ComponentAdapter () { // obrada promene
            public void componentResized (ComponentEvent d) { // veličine
                crt.promeniDim (getWidth(), getHeight()); // platna)
            }
        });
    }

    public Platno dodaj (Figura fig) // Dodavanje figure.
    { crt.dodaj (fig); return this; }

    public Color boja (Point T) { // Boja tačke.
        Point TT = new Point ((int)T.getX(), getHeight()-(int)T.getY());
        try { return crt.boja (TT); }
        catch (GNePripada g) { return null; }
    }

    public void paint (Graphics g) { // Ponovno iscrtavanje platna.
        crt.crtaj (g, 0, getHeight());
    }
}

```

```
// GCrtez.java - Apstraktna klasa za greške crteža.
```

```
package crtez;

public abstract class GCrtez extends usluge.Greska {
    public GCrtez (String s) { super (s); }
}
```

Zadatak 6.5

```
// GNePripada.java - Klasa za greške: Tačka ne pripada figuri.
```

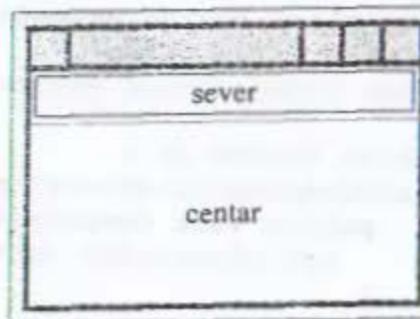
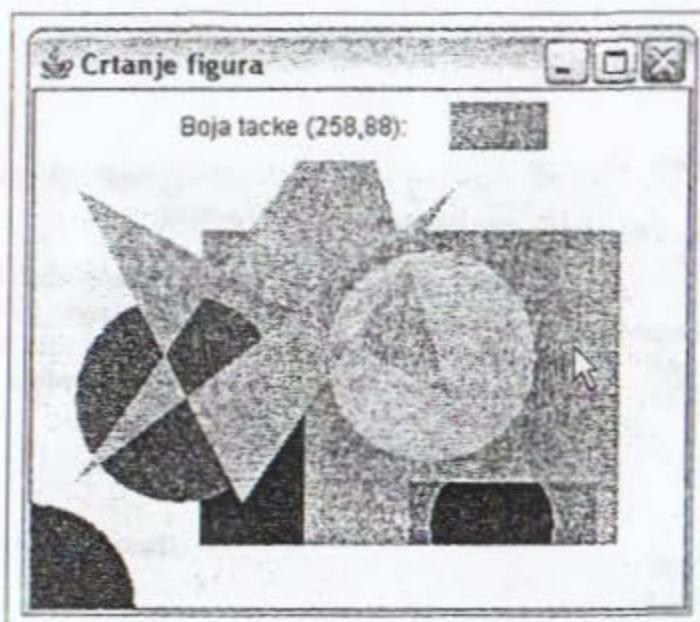
```
package crtez;

public class GNePripada extends GCrtez {
    public GNePripada () {
        super ("Tacka ne pripada figuri!");
    }
}
```

```
// GNeisprPravoug.java - Klasa za greške: Neispravan pravougaonik.
```

```
package crtez;

public class GNeisprPravoug extends GCrtez {
    public GNeisprPravoug () {
        super ("Neispravna temena pravougaonika!");
    }
}
```



```

// CrtezT.java - Ispitivanje klasa za crtanje.

import crtez.*;
import java.awt.*;
import java.awt.event.*;

public class CrtezT extends Frame {
    private Platno platno = new Platno ();           // Platno za crtanje.
    private Label tacka = new Label ("Boja tacke:"); // Prikaz koordinata.
    private Label boja = new Label ("");              // Prikaz boje tačke.

    private void popuniProzor () {                   // Popunjavanje prozora:
        add (platno, "Center");
        platno.addMouseMotionListener (new MouseMotionAdapter () {
            public void mouseMoved (MouseEvent d) { // - obrada pomeranja miša,
                Point p = d.getPoint ();
                tacka.setText ("Boja tacke (" + (int)p.getX () + "," +
                               (int)p.getY () + "): " );
                boja.setBackground (platno.boja (p));
                validate ();
            }
        });
        Panel ploca = new Panel ();                  // - ploča za prikaz podataka.
        add (ploca, "North"); ploca.add (tacka); ploca.add (boja);
    }

    private CrtezT () {                           // Inicijalizacija.
        super ("Crtanje figura");
        setBounds (100, 100, 320, 280);
        popuniProzor ();
        addWindowListener (new WindowAdapter () {
            public void windowClosing (WindowEvent d) { dispose (); }
        });
    }

    public static void main (String[] varg) {      // Glavna funkcija.
        CrtezT crt = new CrtezT ();
        try {
            crt.platno
                .dodaj (new Crtez (Color.GREEN, new Point (80,30), 200, 150)
                    .dodaj (new Pravoug (Color.BLACK, new Point (-10,-10),
                        new Point (50,70)) )
                .dodaj (new Crtez (Color.GRAY, new Point(100,-20), 90, 50)
                    .dodaj (new Krug (Color.BLUE, new Point(40,30), 30)))
                .dodaj (new Krug (Color.CYAN, new Point (110,90), 50)))
            .dodaj (new Krug (Color.MAGENTA, new Point (70,100), 50))
            .dodaj (new Krug (Color.RED, new Point(0,0), 50))
            .dodaj (new Mnogougao (new Color (160,160,160)))
                .dodaj (new Point ( 20, 60))
                .dodaj (new Point (150,250))
                .dodaj (new Point (200,100))
                .dodaj (new Point ( 20,200))
                .dodaj (new Point (100, 50))
                .dodaj (new Point (200,200))
        };
        crt.setVisible (true);
    } catch (GCrtez g) { System.out.println (g); }
    }
}

```

### Zadatak 8.10 Skladišta, aktivni proizvođači i potrošači

Napisati na jeziku Java sledeće klase (uporediti sa zadatkom 7.2):

- **Skladište** celih brojeva ima jedinstven, automatski generisan identifikacioni broj i može da uskladištava zadati broj celih brojeva. Stvara se prazno, posle čega može da se dodaje i uzima po jedan ceo broj. Pri pokušaju stavljanja podatka u puno skladište, odnosno uzimanja podatka iz praznog skladišta, nit izvršioca radnje se privremeno blokira. Tekstualni oblik skladišta se sastoji od niza sadržanih brojeva, jedan broj po redu. Posle svake promene stanje skladišta može da se prikaže u polju grafičke korisničke površi koje je pridruženo skladištu.
- Aktivan **proizvođač** ima jedinstven, automatski generisan identifikacioni broj. U slučajnim vremenjskim intervalima stavlja po jedan ceo broj u zadato skladište. Brojevi su oblika  $1000 \ id + br$ , gde su:  $id$  – identifikacioni broj proizvođača i  $br$  – uzastopni celi brojevi. Najkraće i najduže vreme proizvodnje brojeva je parametar proizvođača. Neposredno pre stavljanja u skladište broj može da se prikaže u polju grafičke korisničke površi koje je pridruženo proizvođaču. Za vreme čekanja na ispraznjeno mesto u skladištu pridruženo polje treba da ima izmenjen izgled.
- Aktivan **potrošač** ima jedinstven, automatski generisan identifikacioni broj. U slučajnim vremenjskim intervalima uzima po jedan ceo broj iz zadatog skladišta. Najkraće i najduže vreme potrošnje brojeva je parametar potrošača. Odmah posle uzimanja iz skladišta broj može da se prikaže u polju grafičke korisničke površi, koje je pridruženo proizvođaču u obliku  $100000 \ id + br$ , gde su:  $id$  – identifikacioni broj potrošača i  $br$  – vrednost uzetog broja. Za vreme čekanja na popunjeno mesto u skladištu pridruženo polje treba da ima izmenjen izgled.

Napisati na jeziku Java program za ispitivanje prethodnih klasa.

**Rešenje:**

```
// Skladiste.java -
// Klasa za uskladištanje podataka.

package skladiste2;
import usluge.Polje;

Zadatak 8.5

public class Skladiste {

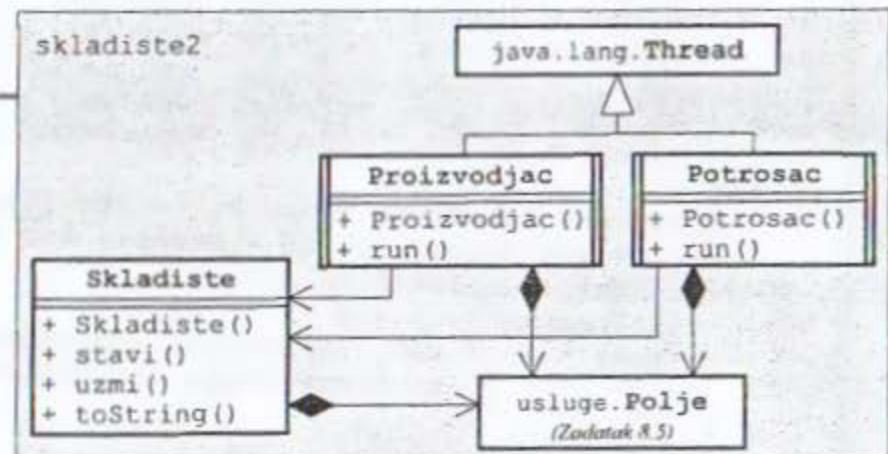
    private static int ukId = 0; // Poslednje korišćeni identifikator.
    private int id = ++ukId; // Identifikator skladišta.
    private int[] niz; // Niz za uskladištanje podataka.
    private int ulaz, izlaz; // Mesto stavljanja i uzimanja podatka.
    private int pun; // Broj popunjenih mesta.
    private Polje polje; // Polje za prikaz stanja skladišta.

    public Skladiste (int kap, Polje p) // Inicijalizacija.
        { niz = new int [kap]; polje = p; }

    public Skladiste (int kap) { this (kap, null); }

    public synchronized void stavi (int vredn) throws InterruptedException {
        while (pun == niz.length) wait (); // Stavljanje podatka.
        niz[ulaz++] = vredn; if (ulaz == niz.length) ulaz = 0;
        pun++; notifyAll ();
        if (polje != null) polje.pisi (toString ());
    }

    public synchronized int uzmi () throws InterruptedException {
        if (pun == 0) return -1; // Brisanje podatka.
        int vredn = niz[ulaz];
        pun--; notifyAll ();
        if (polje != null) polje.pisi (toString ());
        return vredn;
    }
}
```



```

        while (pun == 0) wait (); // Uzimanje podatka.
        int vredn = niz[izlaz++];
        if (izlaz == niz.length) izlaz = 0;
        pun--;
        notifyAll ();
        if (polje != null) polje.pisi (toString ());
        return vredn;
    }

    public synchronized String toString () { // Tekstualni oblik.
        StringBuffer s = new StringBuffer ();
        for (int i=0; i<pun; i++)
            s.append (niz[(izlaz+i) % niz.length]).append ('\n');
        return s.toString ();
    }
}

```

// Proizvodjac.java - Klasa proizvođača.

```

package skladiste2;
import usluge.Polje; → Zadatak 8.5
import java.awt.List;

public class Proizvodjac extends Thread {

    private static int ukId = 0; // Poslednje korišćeni identifikator.
    private int id = ++ukId; // Identifikator proizvođača.
    private Skladiste skladiste; // Pridruženo skladište.
    private int minVreme, maxVreme; // Najkratče i najduže vreme proizvodnje.
    private int broj; // Poslednje proizvedeni podatak.
    private Polje polje; // Polje za prikaz stanja proizvođača.

    public Proizvodjac // Inicijalizacija.
        (Skladiste sklad, int minVr, int maxVr, Polje p) {
        minVreme = minVr; maxVreme = maxVr;
        skladiste = sklad; polje = p;
    }

    public Proizvodjac (Skladiste sklad, int minVr, int maxVr)
        { this (sklad, minVr, maxVr, null); }

    public Proizvodjac (Skladiste sklad) { this (sklad, 1000, 2000); }

    public void run () { // Telo niti.
        try {
            while (! interrupted ()) {
                sleep ((long)(minVreme + Math.random() * (maxVreme-minVreme)));
                int vredn = id*1000 + ++broj;
                if (polje != null) polje.pisi (vredn);
                if (polje!=null && polje.indeks()!=-1)
                    ((List)polje.komponenta ()).select (polje.indeks ());
                skladiste.stavi (vredn);
                if (polje!=null && polje.indeks()!=-1)
                    ((List)polje.komponenta ()).deselect (polje.indeks ());
            }
        } catch (InterruptedException g) {}
    }
}

```

```

// Potrosac.java - Klasa potrošača.

package skladiste2;
import usluge.Polje; → Zadatak 8.5
import java.awt.List;

public class Potrosac extends Thread {

    private static int ukId = 0;      // Poslednje korišćeni identifikator.
    private int id = ++ukId;          // Identifikator potrošača.
    private Skladiste skladiste;     // Pridruženo skladiste.
    private int minVreme, maxVreme;   // Najkrace i najduže vreme potrošnje.
    private Polje polje;             // Polje za prikaz stanja potrošača.

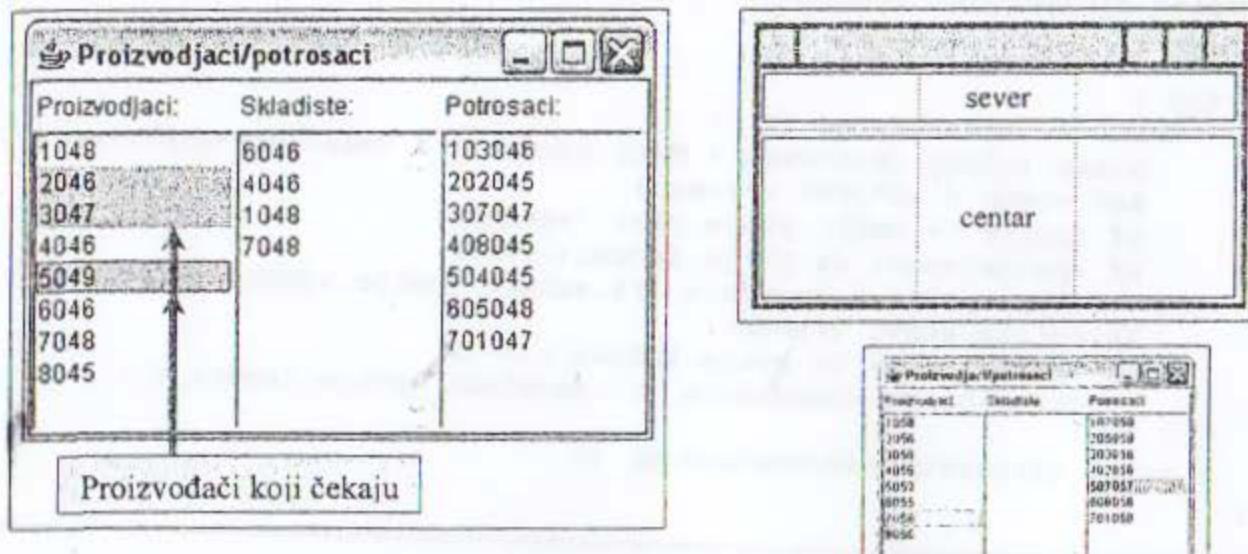
    public Potrosac                               // Inicijalizacija.
        (Skladiste sklad, int minVr, int maxVr, Polje p) {
        minVreme = minVr; maxVreme = maxVr;
        skladiste = sklad; polje = p;
    }

    public Potrosac (Skladiste sklad, int minVr, int maxVr)
        { this (sklad, minVr, maxVr, null); }

    public Potrosac (Skladiste sklad) { this (sklad, 1000, 2000); }

    public void run () {                         // Telo niti.
        try {
            while (! interrupted ()) {
                if (polje!=null && polje.indeks()!=-1)
                    ((List)polje.komponenta ()).select (polje.indeks ());
                int vredn = skladiste.uzmi ();
                if (polje!=null && polje.indeks()!=-1)
                    ((List)polje.komponenta ()).deselect (polje.indeks ());
                if (polje != null) polje.pisi (id*100000 + vredn);
                sleep ((long) (minVreme + Math.random() * (maxVreme-minVreme)));
            }
        } catch (InterruptedException g) {}
    }
}

```



```

// Skladiste2T.java - Ispitivanje klase proizvođača i potrošača.

import java.awt.*; import java.awt.event.*;
import usluge.*; import skladiste2.*;

class Skladiste2T extends Frame {
    private Thread[] niti; // Niti proizvođača i potrošača.

    public Skladiste2T (String[] vpar) { // SASTAVLJANJE PROZORA I
        super ("Proizvodjaci/potrosaci"); // INICIJALIZACIJA SISTEMA:
        setBounds (100, 100, 300, 200);

        Panel ploca = new Panel (new GridLayout (1, 3)); // Ploča za natpise.
        add (ploca, "North");
        ploca.add (new Label ("Proizvodjaci:"));
        ploca.add (new Label ("Skladiste:"));
        ploca.add (new Label ("Potrosaci:")); // Ploča za podatke.

        add (ploca = new Panel (new GridLayout (1, 3)), "Center");
        List lstProizvodjaci = new List ();
        TextArea tksSkladiste = new TextArea ();
        List lstPotrosaci = new List ();
        lstProizvodjaci.setMultipleMode (true);
        lstPotrosaci.setMultipleMode (true);
        ploca.add (lstProizvodjaci);
        ploca.add (tksSkladiste);
        ploca.add (lstPotrosaci); // Parametri rada:

        int kap = Integer.parseInt (vpar[0]), // - kapacitet skladista,
            brPro = Integer.parseInt (vpar[1]), // - broj proizvođača,
            minPro = Integer.parseInt (vpar[2]), // - najkraće i najduže vreme
            maxPro = Integer.parseInt (vpar[3]), // proizvodnje,
            brPot = Integer.parseInt (vpar[4]), // - broj potrošača,
            minPot = Integer.parseInt (vpar[5]), // - najkraće i najduže vreme
            maxPot = Integer.parseInt (vpar[6]); // potrošnje.

        try {
            Skladiste skladiste = new Skladiste (kap, new Polje (tksSkladiste));
            niti = new Thread [brPro + brPot]; int k = 0;
            for (int i=0; i<brPro; i++) {
                lstProizvodjaci.add ("****");
                (niti[k++]) = new Proizvodjac (skladiste, minPro, maxPro,
                    new Polje (lstProizvodjaci, i))).start (); Zadatak 8.5
            }
            for (int i=0; i<brPot; i++) {
                lstPotrosaci.add ("****");
                (niti[k++]) = new Potrosac (skladiste, minPot, maxPot,
                    new Polje (lstPotrosaci, i))).start ();
            }
        } catch (GKompNeOdgovara g) {} Zadatak 8.5
        addWindowListener (new WindowAdapter () {
            public void windowClosing (WindowEvent d)
            { for (Thread nit: niti) nit.interrupt (); dispose (); }
        });
    }

    public static void main (String[] vpar) // GLAVNA FUNKCIJA.
    { new Skladiste2T (vpar).setVisible (true); }
}

```

% java SkladisteT 4 8 4000 8000 7 3500 7000

**Zadatak 8.11 Aktivni časovnici**

Napisati na jeziku Java aktivnu klasu časovnika koji je u stanju da trenutni datum i vreme prikazuje u dva odvojena polja grafičke korisničke površi. Predvideti mogućnost zaustavljanja, nastavljanja i prekidanja rada časovnika.

Napisati na jeziku Java program za ispitivanje prethodne klase.

**Rešenje:**

```
// Casovnik.java - Klasa časovnika.

package usluge;
import java.util.Calendar;

public class Casovnik extends Thread {

    private boolean radi = false; // Da li nit treba da radi?
    private Polje datum, vreme; // Polja za prikaz datuma i vremena.

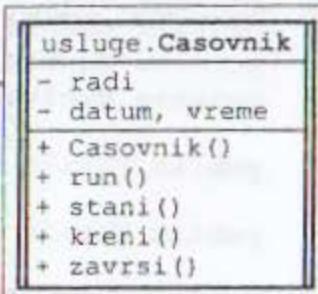
    public Casovnik (Polje d, Polje v) // Inicijalizacija.
        { vreme = v; datum = d; start (); }

    public void run () { // TELO NITI:
        try {
            while (! interrupted ()) { // - uslovni završetak niti,
                if (! radi) synchronized (this) // - uslovno pauziranje niti,
                    ( while (! radi) wait ());
                Calendar t = Calendar.getInstance (); // - koristan rad,
                if (vreme != null) vreme.pisi(
                    t.get(Calendar.HOUR_OF_DAY) + ":" +
                    t.get(Calendar.MINUTE) + ":" +
                    t.get(Calendar.SECOND));
                if (datum != null) datum.pisi(
                    t.get(Calendar.DATE) + "." +
                    (t.get(Calendar.MONTH)+1) + "." +
                    t.get(Calendar.YEAR));
            };
            sleep (1000); // - čekanje 1000 ms.
        }
        catch (InterruptedException g) {}
    }

    public void stani () { radi = false; } // Privremeno zaustavljanje.

    public synchronized void kreni () // Nastavak rada.
        { radi = true; notify (); }

    public void zavrsi () { interrupt (); } // Završetak rada.
}
```



```

// CasovnikT.java -
// Ispitivanje klase
// časovnika.

import java.awt.*;
import java.awt.event.*;
import usluge.*;

public class CasovnikT extends Frame {
    private Casovnik casovnik; // Korišćeni časovnik.

    private void popuniProzor () { // Popunjavanje prozora:
        Label datum = new Label("", Label.CENTER), // - oznake za datum
        vreme = new Label("", Label.CENTER); // i vreme,
        vreme.setFont (new Font (null, Font.BOLD, 24));
        add (datum, "North"); add (vreme, "Center");
        try { casovnik = new Casovnik (new Polje (datum), // - stvaranje
                                      new Polje (vreme)); } // časovnika,
        catch (GKompNeOdgovara g) {}

        Panel ploca = new Panel(); add (ploca, "South"); // - ploča za dugmad,
        Button dugme = new Button ("Kreni"); // - dugme za pokretanje časovnika,
        ploca.add (dugme);
        dugme.addActionListener (new ActionListener () {
            public void actionPerformed (ActionEvent d)
                { casovnik.kreni (); }
        });

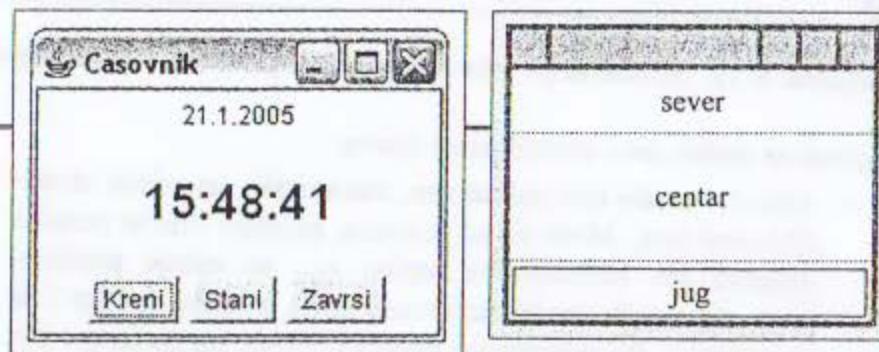
        ploca.add (dugme = new Button ("Stani")); // - dugme za zaustavljanje časovnika,
        dugme.addActionListener (new ActionListener () {
            public void actionPerformed (ActionEvent d)
                { casovnik.stani (); }
        });

        ploca.add (dugme = new Button ("Zavrsi")); // - dugme za uništavanje časovnika i zatvaranje prozora.
        dugme.addActionListener (new ActionListener () {
            public void actionPerformed (ActionEvent d)
                { casovnik.zavrsi (); dispose (); }
        });
    }

    private CasovnikT () { // Inicijalizacija:
        super ("Casovnik");
        setBounds (100, 100, 200, 150);
        popuniProzor (); // - popunjavanje prozora,
        addWindowListener (new WindowAdapter () { // - obrada zatvaranja
            public void windowClosing (WindowEvent d) // prozora.
                { casovnik.zavrsi (); dispose (); }
        });
    }

    public static void main (String[] varg) // GLAVNA FUNKCIJA.
    { new CasovnikT ().setVisible (true); }
}

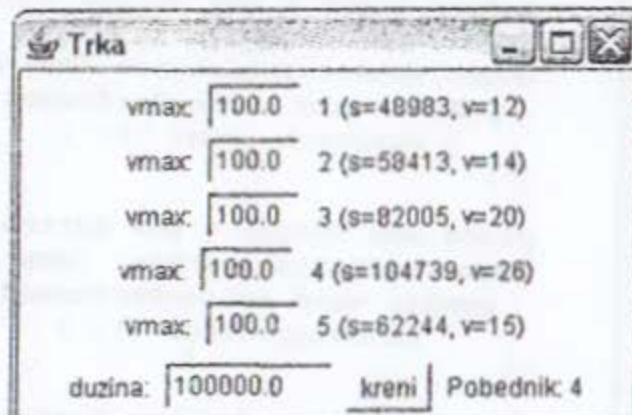
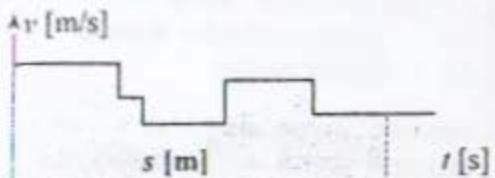
```



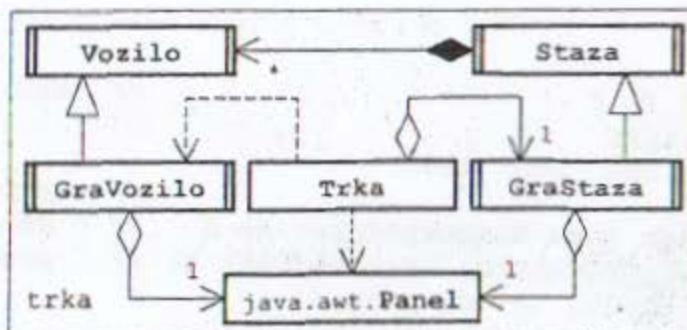
### Zadatak 8.12 Simuliranje trke automobila (aktivna i grafička vozila i staze)

Napisati na jeziku Java sledeći paket tipova:

- Aktivno *vozilo* ima jedinstven, automatski generisan identifikacioni broj. Može da se pokrene, zaustavi i da se prekine njegova nit. Maksimalna brzina  $v_{max}$  se zadaje prilikom stvaranja vozila (podrazumevano 100), a može kasnije i da se promeni. Pri pokretanju početna brzina vozila iznosi 5% od  $v_{max}$ , posle čega se u slučajnim vremenskim intervalima, koji traju od 100 do 200 ms, promeni za slučajan iznos od -5% do +15% trenutne brzine, ali ne iznad brzine  $v_{max}$  i ne ispod brzine 5% od  $v_{max}$ . Može da se dohvati identifikacioni broj vozila, trenutna brzina i pređeni put od poslednjeg pokretanja. Tekstualni oblik vozila sadrži identifikacioni broj i celobrojne delove pređenog puta i trenutne brzine.
- Na aktivnoj *stazi* može biti nekoliko vozila koja se kreću po njoj. Stvara se prazna, posle čega se vozila dodaju jedno po jedno. Dužina staze se zadaje prilikom stvaranja staze (podrazumevano 100000), a može kasnije i da se promeni. Može da se zatraži pokretanje svih vozila na stazi i prekidanje niti staze i svih vozila na njoj. Po pokretanju vozila staza svakih 20 ms proverava da li je bar jedno vozilo stiglo do kraja staze. Ako jeste, staza zaustavi sva vozila i pronalazi vozilo koje je prevalilo najduži put.
- Grafičko vozilo* je vozilo koje se inicijalizuje grafičkom pločom (Panel) koju popunjava komponentama tako da može prilikom pokretanja vozila postaviti  $v_{max}$  i pri svakom sastavljanju tekstualnog oblika vozila prikazati isti.
- Grafička staza* je staza koja se inicijalizuje grafičkom pločom (Panel) koju popunjava komponentama tako da korisnik može postaviti dužinu staze, pokrenuti sva vozila i na kraju videti identifikacioni broj pobedničkog vozila. Dok se vozila kreću novo pokretanje vozila mora biti onemogućeno.
- Trka* je program koji simulira trku i na grafičkoj korisničkoj površi prema slici prikazuje 5 grafičkih vozila na jednoj grafičkoj stazi, tako da se prati trenutno stanje trke.



Rešenje:



```

// Vozilo.java - Klasa aktivnih vozila.

package trka;

public class Vozilo extends Thread {

    private static int ukId = 0; // Poslednje korišćeni identifikator.
    private int id = ++ukId; // Identifikator vozila.
    protected double vmax; // Maksimalna brzina.
    private long t; // Vreme poslednje promene brzine.
    private double v; // Trenutna brzina.
    private double suma_vdt; // Pređeni put do poslednjeg računanja.
    private boolean radi = false; // Da li treba da vozi?

    public Vozilo (double vmax) { postavi (vmax); } // Konstruktori.

    public Vozilo () { this (100); }

    public void postavi (double vmax) // Postavljanje maksimalne brzine.
        { this.vmax = vmax; }

    public void run () // Telo niti.
        try {
            while (! interrupted ()) {
                if (! radi) { v = 0; synchronized (this) { wait (); } }
                long u = System.currentTimeMillis ();
                suma_vdt += v * (u - t);
                t = u;
                v *= (0.95 + 0.2 * Math.random ());
                if (v > vmax) v = vmax;
                if (v < 0.05*vmax) v = 0.05 * vmax;
                sleep (100 + (int) (Math.random () * 100));
            }
        } catch (InterruptedException g) {}
    }

    public void pocni () { start (); } // Pokretanje niti.

    public synchronized void kreni () // Polazak vozila.
        suma_vdt = 0;
        t = System.currentTimeMillis ();
        radi = true; notify ();
    }

    public void stani () { radi = false; } // Zaustavljanje vozila.

    public void zavrsi () { interrupt (); } // Prekidanje niti.

    public int id () { return id; } // Identifikacioni broj.

    public double v () { return v; } // Trenutna brzina.

    public double s () // Pređeni put.
        { return suma_vdt + (System.currentTimeMillis () - t) * v; }

    public String toString () // Tekstualni oblik.
        { return id + " (s=" + (int)s () + ", v=" + (int)v () + ")"; }
}

```

```

// Staza.java - Klasa aktivnih staza.

package trka;

public class Staza extends Thread {

    protected class Elem {           // Element liste vozila:
        Vozilo vozilo;             // - vozilo u elementu,
        Elem sled;                 // - sledeći element,
        Elek (Vozilo v) {          // - konstruktor.
            vozilo = v;
            if (prvi == null) prvi = this; else posl.sled = this;
            posl = this;
        }
    }

    protected Elek prvi = null, posl = null; // Prvi i poslednji element.
    protected double duz;                   // Dužina staze.
    protected boolean gotova;              // Da li je trka gotova?
    private boolean radi = false;          // Da li je nit aktivna?

    public Staza (double duz) { this.duz = duz; } // Konstruktori.

    public Staza () { this (100000); }

    public Staza dodaj (Vozilo v)           // Dodavanje vozila.
        { new Elek (v); return this; }

    public void run () {                    // Telo niti.
        try {
            while (! interrupted ()) {
                if (! radi) synchronized (this) { wait (); }
                sleep (20); proveri ();
            }
        } catch (InterruptedException g) {}
    }

    public void pocni () { start (); }      // Pokretanje niti.

    public synchronized void kreni () {      // Početak trke.
        for (Elek tek=prvi; tek!=null; tek=tek.sled)
            tek.vozilo.kreni ();
        gotova = false; radi = true; notify ();
    }

    protected void proveri () {             // Provera završetka trke.
        for (Elek tek=prvi; tek!=null; tek=tek.sled)
            if (tek.vozilo.s () >= duz) { stani (); gotova = true; break; }
    }

    public Vozilo naCelu () {               // Vozilo na čelu trke.
        double maxs = 0; Vozilo vodi = null;
        for (Elek tek=prvi; tek!=null; tek=tek.sled) {
            double s = tek.vozilo.s ();
            if (s > maxs) { maxs = s; vodi = tek.vozilo; }
        }
        return vodi;
    }
}

```

```

private synchronized void stani () { // Zaustavljanje vozila.
    radi = false;
    for (Elem tek=prvi; tek!=null; tek=tek.sled)
        tek.vozilo.stani ();
}

public void zavrsi () { // Prekidanje svih niti.
    interrupt ();
    for (Elem tek=prvi; tek!=null; tek=tek.sled)
        tek.vozilo.zavrsi ();
}
}

```

```

// GraVozilo.java - Klasa grafičkih vozila.

package trka;
import java.awt.*;

public class GraVozilo extends Vozilo {

    private Panel plo; // Ploča pridružena vozilu.
    private TextField tksVmax = // Polje za unos maksimalne brzine.
        new TextField (Double.toString(vmax), 3);
    private Label oznVozilo = // Natpis za prikaz tekstualnog oblika.
        new Label (toString ());

    public GraVozilo (double vmax, Panel plo) // Konstruktori.
        { super (vmax); popuniPlocu (plo); }

    public GraVozilo (Panel plo)
        { super (); popuniPlocu (plo); }

    private void popuniPlocu (Panel plo) { // Popunjavanje ploče.
        this.plo = plo;
        plo.add (new Label ("vmax:", Label.RIGHT));
        plo.add (tksVmax);
        plo.add (oznVozilo);
    }

    public void kreni () { // Pokretanje vozila.
        try {
            postavi (Double.parseDouble(tksVmax.getText()));
        } catch (NumberFormatException g) {}
        super.kreni ();
    }

    public String toString () { // Prikaz tekstualnog oblika.
        String s = super.toString ();
        if (oznVozilo != null) {
            oznVozilo.setText (s);
            plo.validate ();
        }
        return s;
    }
}

```

```

// GraStaza.java - Klasa grafičkih staza.

package trka;
import java.awt.*;
import java.awt.event.*;

public class GraStaza extends Staza implements ActionListener {

    private Panel plo;           // Ploča pridružena stazi.
    private TextField tksDuz =   // Polje za unos dužine staze.
        new TextField (Double.toString (duz), 8);
    private Button dgmKreni =    // Dugme za pokretanje trke.
        new Button ("kreni");
    private Label oznPobednik =  // Natpis za prikaz pobednika.
        new Label ();

    public GraStaza (double duz, Panel plo) // Konstruktori.
    { super (duz); popuniPlocu (plo); }

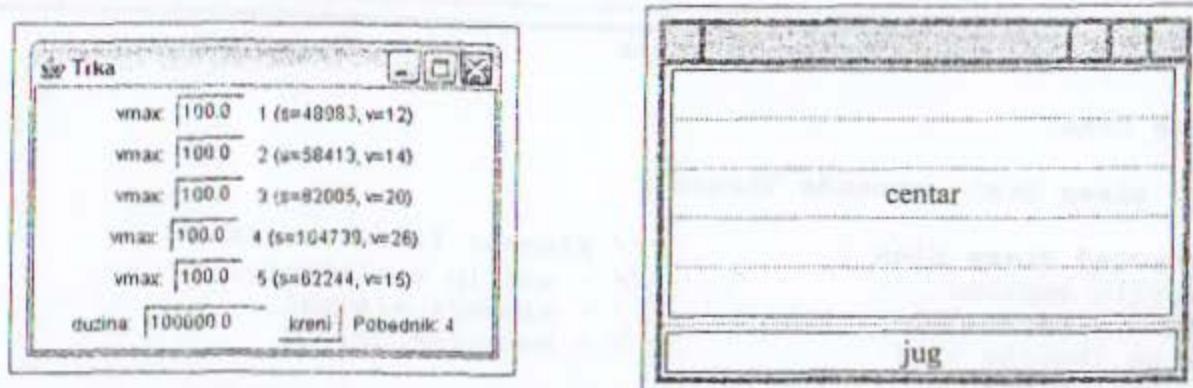
    public GraStaza (Panel plo)
    { super (); popuniPlocu (plo); }

    private void popuniPlocu (Panel plo) { // Popunjavanje ploče.
        this.plo = plo;
        plo.add (new Label ("duzina:", Label.RIGHT));
        plo.add (tksDuz);
        plo.add (dgmKreni);
        plo.add (oznPobednik);
        dgmKreni.addActionListener (this);
    }

    public void actionPerformed (ActionEvent d) { // Obrada pritisaka
        dgmKreni.setEnabled (false);           // dugmeta "kreni".
        oznPobednik.setText ("");
        plo.validate ();
        try {
            duz = Double.parseDouble (tksDuz.getText ());
        } catch (NumberFormatException g) {}
        kreni ();
    }

    protected void proveri () {                // Provera kraja trke i
        super.proveri ();                     // prikaz vozila na čelu.
        oznPobednik.setText
            ((gotova ? "Pobednik: " : "Vodi: ") + naCelu ().id ());
        plo.validate ();
        if (gotova) dgmKreni.setEnabled (true);
        for (Elem tek=prvi; tek!=null; tek=tek.sled)
            tek.vozilo.toString ();
    }
}

```



```
// Trka.java - Program za organizovanje trke na grafičkoj površi.

package trka;
import java.awt.*;
import java.awt.event.*;

public class Trka extends Frame {

    private static final int BR_VOZILA = 5; // Broj vozila u trci.
    private Staza staza; // Staza za trku.

    private void popuniProzor () { // Popunjavanje prozora:
        Panel plo = new Panel (); // - ploča upravljanje trkom,
        add (plo, "South");
        (staza = new GraStaza (plo)).pocni ();

        add (plo = new Panel (new GridLayout (0,1)), "Center"); // - ploča za
        for (int i=0; i<BR_VOZILA; i++) { // prikaz
            Panel p = new Panel (); plo.add (p); // vozila.
            Vozilo v = new GraVozilo (p); v.pocni ();
            staza.dodaj (v);
        }
    }

    public Trka () { // Inicijalizacija.
        super ("Trka");
        setSize (300, 200);
        popuniProzor ();
        setVisible (true);
        addWindowListener (new WindowAdapter () {
            public void windowClosing (WindowEvent d)
            { staza.zavrsi (); dispose (); }
        });
    }

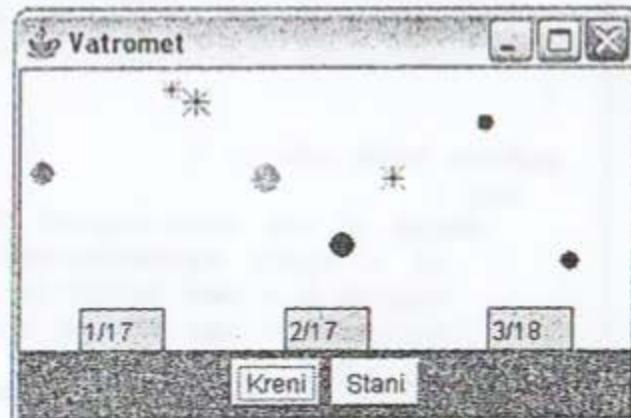
    public static void main (String[] varg) // Glavna funkcija.
    { new Trka (); }
}
```

% java trka.Trka

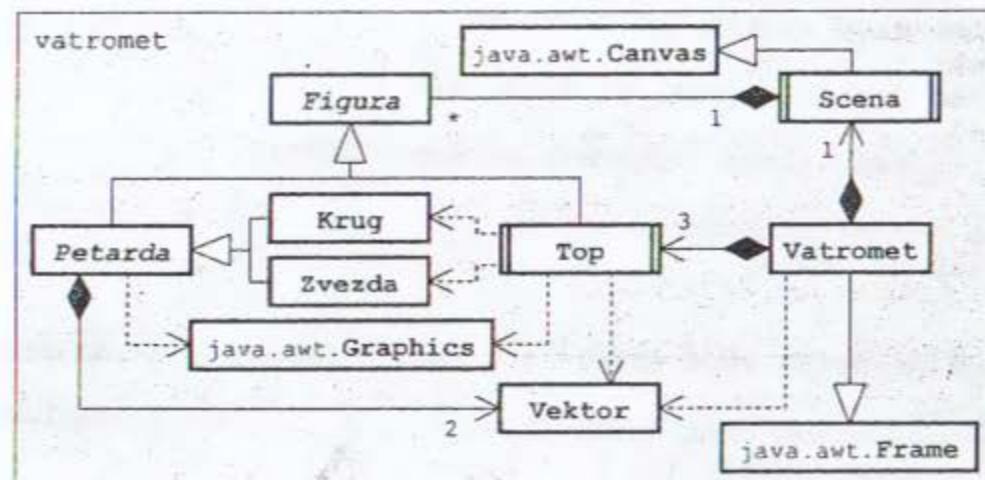
Zadatak 8.13 Vatromet (klase: Vektor, Figura, Petarda, Krug, Zvezda, Top, Scena, Vatromet)

Napisati na jeziku Java sledeći paket tipova:

- **Vektor** u x-y ravni se zadaje realnim koordinatama koje mogu da se dohvate.
  - Apstraktna **figura** u ravni predviđa iscrtavanje figure na grafičkoj sceni i uništavanje figure. Uništavanje podrazumeva uklanjanje figure iz grafičke scene.
  - Aktivna **scena** je grafička komponenta koja može da sadrži proizvoljan broj figura koje iscrtava svakih 20 ms. Stvara se prazna, posle čega se figure mogu pojedinačno dodavati i izbacivati.
  - Apstraktna **petarda** je figura koja se zadaje vektorom početnog položaja centra ( $x_0, y_0$ ) i vektorom početne brzine ( $v_{x0}, v_{y0}$ ), bojom i scenom na kojoj se iscrtava. Može da se dohvati vektor trenutnog položaja petarde čije su komponente  $x = x_0 + v_{x0} t$  i  $y = y_0 + v_{y0} t - 10 t^2$ , gde je  $t$  [s] proteklo vreme od momenta stvaranja petarde. Petarda se uništava kad njen centar napušta okvire scene.
  - **Krug** je petarda koja se iscrtava kao popunjeno krug zadatog prečnika.
  - **Zvezda** je petarda koja se iscrtava kao osmokraka zvezda unutar kvadrata zadate dužine ivica.
  - Aktivan **top** je figura koja u vremenskim intervalima između 0,5 i 1 sekunde ispaljuje po jednu petardu. Top se zadaje vektorom položaja i ima jedinstven, automatski generisan identifikacioni broj. Tekstualni oblik sadrži identifikacioni broj i broj ispaljenih petardi. Iscrtava se u obliku svetlosivo popunjenoj i crno uokvirenog pravougaonika širine 40 i visine 20 po kojem je crno ispisani tekstualni oblik topa. Vektor položaja označava sredinu donje ivice topa. Rad topa može da se zaustavi, da se ponovo pokrene i da se sasvim zaustavi. Ispaljene petarde se pojavljuju u sredini gornje ivice topa. Kao vrsta petarde nasumice se bira krug ili zvezda s podjednakim verovatnoćama. Dimenzije petardi se kreću između 4 i 16, početne brzine između -100 i 100 u pravcu x-ose, odnosno između 50 i 150 u pravcu y-ose. Boja ima slučajne komponente crvene, zelene i plave boje između 0 i 1.
  - **Vatromet** na grafičkoj korisničkoj površi prema slici prikazuje scenu sa tri topa ravnomođno raspoređena po širini donje ivice scene. Rad topova se pokreće i zaustavlja istovremeno.



**Rešenje:**



```
// Vektor.java - Klasa vektora u ravni.

package vatromet;

public class Vektor {
    private double x, y;                                // Komponente vektora.

    Vektor (double xx, double yy) { x = xx; y = yy; } // Konstruktor.

    double x () { return x; }                          // Dohvatanje
    double y () { return y; }                          // komponenata.
}
```

```
// Figura.java - Klasa apstraktnih figura.

package vatromet;
import java.awt.Canvas;

public abstract class Figura {
    protected Scena scena;                            // Korišćena scena.

    protected Figura (Scena s) {                      // Konstruktor.
        (scena = s).dodaj (this);
    }

    public abstract void crtaj (); // Crtanje figure.

    public void unisti () {                         // Izbacivanje iz scene.
        scena.izbaci (this);
    }
}
```

```
// Petarda.java - Klasa apstraktnih petardi.

package vatromet;
import java.awt.*;

public abstract class Petarda extends Figura {

    private Vektor r0, v0;                            // Vektor i početnog položaja i brzine.
    private long t0;                                 // Vreme stvaranja petarde.
    protected Color boja;                            // Boja petarde.
    protected Figura (Scena s, Vektor rr0, Vektor vv0, Color bboja) {
        super (s);
        r0 = rr0; v0 = vv0; boja = bboja;
        t0 = System.currentTimeMillis ();
    }

    public Vektor položaj () {                      // Trenutni položaj petrade.
        double t = (System.currentTimeMillis () - t0) / 1000.0;
        return new Vektor (r0.x () + v0.x () * t, r0.y () + v0.y () * t - 10 * t * t);
    }

    public void crtaj () {                           // Zajednički deo crtanja petarde.
        int sir = scena.getWidth ();
        int vis = scena.getHeight ();
        Vektor p = položaj ();
    }
}
```

```

int x = (int)p.x();
int y = vis - (int)p.y();
if (x>=0 && x<sir && y>=0 && y<vis) {
    Graphics g = scena.getGraphics ();
    g.setColor (boja);
    crt (g, x, y);
} else scena.izbaci (this);
}
// Specifični deo crtanja petarde.
protected abstract void crt (Graphics g, int x, int y);
}

```

```

// Krug.java - Klasa krugova.

package vatromet;
import java.awt.*;

public class Krug extends Petarda {

    private int a;           // Prečnik kruga.
                           // Konstruktor.
    public Krug (Scena s, Vektor r0, Vektor v0, Color boja, int aa)
        { super (s, r0, v0, boja); a = aa; }

    public void crt (Graphics g, int x, int y) // Crtanje kruga.
        { g.fillOval (x-a/2, y-a/2, a, a); }
}

```

```

// Zvezda.java - Klasa zvezdi.

package vatromet;
import java.awt.*;

public class Zvezda extends Petarda {

    private int a;           // Ivica obuhvatnog kvadrata.
                           // Konstruktor.
    public Zvezda (Scena s, Vektor r0, Vektor v0, Color boja, int aa)
        { super (s, r0, v0, boja); a = aa; }

    public void crt (Graphics g, int x, int y) { // Crtanje zvezde.
        g.drawLine (x-a/2, y-a/2, x+a/2, y+a/2);
        g.drawLine (x+a/2, y-a/2, x-a/2, y+a/2);
        g.drawLine (x, y-a/2, x, y+a/2);
        g.drawLine (x-a/2, y, x+a/2, y);
    }
}

```

```

// Top.java - Klasa topova.

package vatromet;
import java.awt.*;

public class Top extends Figura implements Runnable {

    private static int ukId = 0; // Poldjenje korišćeni identifikator.
    private int id = ++ukId;     // Identifikator topa.
    private int x, y;           // Koordinate topa.
}

```

```

private int n = 0;                                // Broj petardi.
private Thread nit = new Thread (this);           // Nit topa.
private boolean radi = false;                     // Da li treba da radi?

public Top (Scena s, Vektor p) {                 // Konstruktor.
    super (s);
    scena = s;
    prenesti (p);
    nit.start();
}

public void prenesti (Vektor p)                   // Premeštanje topa.
{ x = (int)p.x(); y = (int)p.y(); }

public int id () { return id; }                    // Identifikacioni broj.

public String toString() { return id + "/" + n; } // Tekstualni oblik.

public void crtaj () {                            // Crtanje topa.
    int vis = scena.getHeight ();
    int y = vis - this.y;
    Graphics g = scena.getGraphics ();
    g.setColor (Color.LIGHT_GRAY);
    g.fillRect (x-20, y-20, 40, 20);
    g.setColor (Color.BLACK);
    g.drawRect (x-20, y-20, 40, 20);
    g.drawString (toString(), x-18, y-4);
}

public void run () {                             // Telo niti.
try {
    while (! nit.interrupted ()) {
        if (! radi) synchronized (this) { wait (); }
        Vektor p = new Vektor(x, y+20);
        Vektor v = new Vektor (-100+Math.random()*200,
                               50+Math.random()*100);
        Color b = new Color((float)(Math.random()),
                           (float)(Math.random()),
                           (float)(Math.random()));
        int a = (int)(4+Math.random()*12);
        if (Math.random() < 0.5)
            new Krug (scena, p, v, b, a);
        else
            new Zvezda (scena, p, v, b, a);
        nit.sleep ((long)(500+Math.random()*500));
    }
} catch (InterruptedException g) {}
scena.izbaci (this);
}

public synchronized void stani () { radi = false; } // Zaustavljanje rada.

public synchronized void kreni ()                  // Nastavak rada.
{ radi = true; notify (); }

public void zavrsi () { nit.interrupt (); }         // Prekidanje niti.
}

```

```

// Scena.java - Klasa aktivnih scena.

package vatromet;
import java.awt.*;

public class Scena extends Canvas implements Runnable {

    private class Elem {                                     // Element liste figura:
        Figura fig;                                       // - figura u elementu,
        Elem sled;                                         // - sledeći element,
        Elem (Figura f) {                                // - konstruktor.
            fig = f;
            if (prvi == null) prvi = this; else posl.sled = this;
            posl = this;
        }
    }

    private Elem prvi, posl;                            // Početak i kraj liste.
    private Thread nit = new Thread (this);           // Nit scene.

    public Scena () { nit.start (); }                  // Konstruktor.

    public synchronized void dodaj (Figura fig)      // Dodavanje figure.
    { new Elem (fig); }

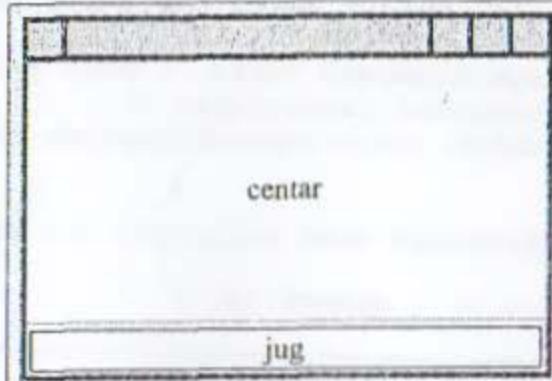
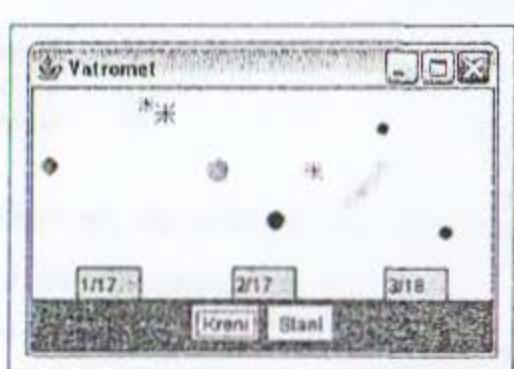
    public synchronized void izbaci (Figura fig) { // Izbacivanje figure.
        Elem tek = prvi, pret = null;
        while (tek!=null && tek.fig!=fig) { pret = tek; tek = tek.sled; }
        if (tek != null) {
            if (pret == null) prvi = tek.sled; else pret.sled = tek.sled;
            if (tek.sled == null) posl = pret;
            if (prvi == null) posl = null;
        }
    }

    public synchronized void paint (Graphics g)       // Crtanje figura u sceni.
    { for (Elem tek=prvi; tek!=null; tek=tek.sled) tek.fig.crtaj (); }

    public void run () {                               // Telo niti.
        try {
            while (! nit.interrupted()) {
                repaint ();
                nit.sleep (20);
            }
        } catch (InterruptedException g) {}
    }

    public void zavrsi () { nit.interrupt (); } // Prekidanje niti.
}

```



```

// Vatromet.java - Klasa vatrometa.

package vatromet;
import java.awt.*;
import java.awt.event.*;

public class Vatromet extends Frame {

    private Scena scena = new Scena();                      // Scena po kojoj se crta.
    private Top[] topovi = new Top[3];                       // Niz topova.

    private void popuniProzor () {                           // Popunjavanje prozora:
        add (scena, "Center");                             // - scena za crtanje,
        scena.addComponentListener (new ComponentAdapter () {
            public void componentResized(ComponentEvent d){ // - raspoređivanje
                for (int i=0; i<topovi.length; i++)          // topova pri
                    topovi[i].premesti (new Vektor (           // promeni
                        scena.getWidth()*(2*i+1)/(2*topovi.length), // veličine
                        0);                                     // scene,
                    );
            }
        });
    }

    Panel plo = new Panel (); add (plo, "South");           // - ploča za dugmad
    plo.setBackground (Color.GRAY);
    Button dgm = new Button ("Kreni"); plo.add (dgm);
    dgm.addActionListener (new ActionListener () {           // - pokretanje
        public void actionPerformed (ActionEvent d)         // topova,
            { for (Top top: topovi) top.kreni(); }
        });
    plo.add (dgm = new Button ("Stani"));
    dgm.addActionListener (new ActionListener () {           // - zaustavljanje
        public void actionPerformed (ActionEvent d)         // topova.
            { for (Top top: topovi) top.stani(); }
        });
    }

    public Vatromet () {                                    // Inicijalizacija:
        super ("Vatromet");
        setSize (300, 200);
        popuniProzor ();                                // - popunjavanje prozora,
        for (int i=0; i<topovi.length; i++)             // - stvaranje topova,
            topovi[i] = new Top (scena, new Vektor(0, 0));
        setVisible (true);
        addWindowListener (new WindowAdapter () {         // - obrada zatvaranja
            public void windowClosing (WindowEvent d) { // prozora.
                for (Top top: topovi) top.zavrsi();
                scena.zavrsi (); dispose ();
            }
        });
    }

    public static void main (String[] varg)               // Glavna funkcija.
        { new Vatromet (); }
}

```

```
$ java vatromet.Vatromet
```

**Zadatak 8.14 Simuliranje rada samoposluge (klase: Ulaz, Kupac, Red, Kasa, Samoposluga)**

Napisati na jeziku Java sledeće klase za simuliranje rada samoposluge:

- Aktivan *ulaz* samoposluge ima jedinstven, automatski generisan identifikacioni broj koji može da se dohvata. Ulaz može da se otvori, zatvori i uništi. Kada je ulaz otvoren kupci ulaze u slučajnim vremenskim intervalima. Najduže vreme između ulazaka kupaca je parametar ulaza. Prikaz ulaza u polju grafičke korisničke površi sadrži identifikacioni broj ulaza. Ako je ulaz otvoren prikaz sadrži i identifikacioni broj kupca koji je poslednji ušao.
- Aktivan *kupac* ima jedinstven, automatski generisan identifikacioni broj koji može da se dohvata. Po ulasku u samoposlugu neko slučajno vreme bira robu (najduže vreme je parametar kupca), potom stane u red ispred slučajno odabrane kase. Posle plaćanja, napušta samoposlugu.
- *Red* kupaca ispred kase je neograničenog kapaciteta. Kupci se dodaju na kraj reda, a uzimaju sa početka reda. Prikaz reda u polju grafičke korisničke površi sadrži identifikacione brojeve kupaca u redu.
- Aktivna *kasa* ima jedinstven, automatski generisan, identifikacioni broj koji može da se dohvata. Ispred kase postoji jedan *red* kupaca. Kasa može da se otvori, zatvori i uništi. Kada je kasa otvorena naplaćuje od kupaca koji stoji u redu ispred nje. Naplaćivanje traje slučajno vreme. Najduže vreme naplaćivanja je parametar kase. Prikaz kase u polju grafičke korisničke površi sadrži identifikacioni broj kase. Ako je naplaćivanje u toku prikaz sadrži i identifikacioni broj kupca koji se opslužuje.
- *Samoposluga* ima jedinstven, automatski generisan, identifikacioni broj koji može da se dohvata. U samoposluzi postoji zadati broj ulaza i kasa. U početku ulazi su zatvoreni, a kase su spremne za rad. Samoposluga može da se otvori, zatvori i uništi. Pri otvaranju svi ulazi se odmah otvore. Pri zatvaranju svi ulazi se odmah zatvore. Zatvaranje se smatra završenim kada i poslednji kupac napusti samoposlugu. Prikaz samoposluge u polju grafičke korisničke površi sadrži broj kupaca u samoposluzi.

Napisati na jeziku Java program s grafičkom korisničkom površi za ispitivanje prethodnih klasa. Predviđeni istovremeno postojanje više samoposluga.

**Rešenje:**

```
// Ulaz.java - Klasa ulaza u samoposlugu.

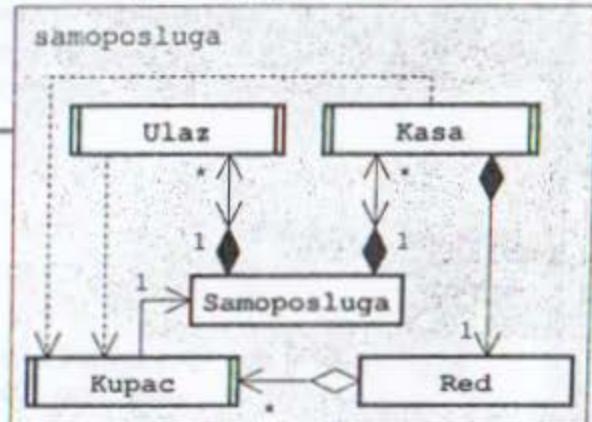
package samoposluga;

import usluge.Polje; → Zadatak 8.5
import java.awt.Color;

class Ulaz extends Thread {

    private static int ukId = 0; // Poslednje korišćeni identifikator.
    private int id = ++ukId; // Identifikator ulaza.
    private int maxUlaz; // Najduže vreme između ulazaka kupaca.
    private int maxKupovina; // Najduže vreme biranja robe.
    private Samoposluga samoposl; // Vlasnička samoposluga.
    private Polje polje; // Polje za prikaz stanja ulaza.
    private boolean radi = false; // Da li radi?

    Ulaz (Samoposluga samop, int maxUl, int maxKup, Polje p) { // Inicijalizacija
        samoposl = samop; maxUlaz = maxUl; maxKupovina = maxKup; // zacijsa.
        if ((polje = p) != null) {
            polje.komponenta ().setForeground (Color.red);
            polje.pisi ("Ulaz" + id);
        }
        start ();
    }
}
```



```

public void run () {                                // Telo niti.
    try {
        while (! interrupted ()) {
            if (! radi) synchronized (this) { wait (); }
            sleep ((long) (Math.random () * maxUlaz));
            samoposl.usaoKupac ();
            Kupac kupac = new Kupac (samoposl, maxKupovina);
            if (polje != null) polje.pisi ("Ulaz" + id + ":" + kupac.id ());
        }
    } catch (InterruptedException g) {}
}

synchronized void zatvori () {                     // Zatvaranje ulaza.
    radi = false;
    if (polje != null) {
        polje.pisi ("Ulaz" + id);
        polje.komponenta ().setForeground (Color.red);
    }
}

synchronized void otvori () {                      // Otvaranje ulaza.
    radi = true; notify ();
    if (polje != null) polje.komponenta ().setForeground (Color.black);
}

synchronized void unisti () {                      // Završetak niti.
    interrupt ();
}

int id () { return id; }                          // Dohvatanje identifikatora ulaza.
}

```

```

// Kupac.java - Klasa kupaca samoposluze.

package samoposluga;

class Kupac extends Thread {

    private static int ukId = 0;      // Poslednje korišćeni identifikator.
    private int id = ++ukId;          // Identifikator kupca.
    private int maxKupovina;          // Najduže vreme biranja robe.
    private Samoposluga samoposl;    // Samoposluga u kojoj se nalazi.

    Kupac (Samoposluga samop, int maxKup)           // Inicijalizacija.
    { samoposl = samop; maxKupovina = maxKup; start (); }

    public void run () {                         // Telo niti:
        try {                                 // - izbor robe,
            sleep ((long) (Math.random () * maxKupovina));
            Kasa[] kase = samoposl.dohvatiKase (); // - izbor kase,
            Kasa kasa = kase[(int) (Math.random () * kase.length)];
            kasa.staviURed (this);
            synchronized (this) { wait (); }        // - čekanje u redu,
            samoposl.izasaoKupac ();              // - izlazak iz
        } catch (InterruptedException g) {}        // samoposluge.
    }

    synchronized void naplaceno () { notify(); } // Dojava da je roba naplaćena.

    int id () { return id; }                   // Dohvatanje identifikatora.
}

```

```

// Red.java - Klasa reda kupaca.

package samoposluga;
import usluge.Polje; → Zadatak 8.5

class Red {

    private class Elem {           // ELEMENT REDA:
        Kupac kupac;             // - kupac u redu,
        Elem sled;               // - sledeći element reda,
        Elem (Kupac k) {         // - stavljanje elementa u listu.
            kupac = k;
            if (prvi == null) prvi = this; else posl.sled = this;
            posl = this;
        }
    }

    private Elem prvi, posl;      // Početak i kraj reda.
    private Polje polje;         // Polje za prikaz stanja reda.

    Red (Polje p) { polje = p; } // Dohvatanje pridruženog polja.

    synchronized void stavi (Kupac kupac) { // Stavljanje kupca u red.
        new Elem (kupac); notifyAll ();
        if (polje != null) polje.pisi (toString ());
    }

    synchronized Kupac uzmi () throws InterruptedException { // Vađenje
        while (prvi == null) wait ();                                // prvog
        Kupac kupac = prvi.kupac;                                     // kupca iz
        prvi = prvi.sled; if (prvi == null) posl = null;           // reda.
        if (polje != null) polje.pisi (toString ());
        return kupac;
    }

    public synchronized String toString () {                         // Tekstualni oblik
        StringBuffer s = new StringBuffer ();                      // sadržaja reda.
        for (Elem tek=prvi; tek!=null; tek=tek.sled)
            s.append (tek.kupac.id()).append ('\n');
        return s.toString ();
    }
}

```

```
// Kasa.java - Klasa kasa samoposluge.
```

```

package samoposluga;
import usluge.Polje; → Zadatak 8.5
import java.awt.Color;

class Kasa extends Thread {

    private static int ukId = 0; // Poslednje korišćeni identifikator.
    private int id = ++ukId;     // Identifikator kase.
    private int maxNaplata;     // Najduže vreme naplaćivanja.
    private Samoposluga samoposl; // Vlasnička samoposluga.
    private Polje polje;        // Polje za prikaz stanja kase.
    private Red red;            // Red ispred kase.
    private boolean radi = true; // Da li radi?
}

```

```

Kasa (Samoposluga samop, int maxNapl, Polje p, Polje pRed) { // Inicijalizacija.
    samoposl = samop; maxNaplata = maxNapl; // zacija.
    red = new Red (pRed);
    if ((polje = p) != null) {
        polje.komponenta ().setForeground (Color.black);
        polje.pisi ("Kasa" + id);
    }
    start ();
}

public void run () { // Telo niti.
    try {
        while (! interrupted ()) {
            if (!radi) synchronized (this) { wait (); }
            Kupac kupac = red.uzmi ();
            if (polje != null) polje.pisi ("Kasa" + id + ":" + kupac.id ());
            sleep ((long) (Math.random () * maxNaplata));
            kupac.naplacono ();
            if (polje != null) polje.pisi ("Kasa" + id);
        }
    } catch (InterruptedException g) {}
}

synchronized void zatvori () { // Zatvaranje kase.
    radi = false;
    if (polje != null) polje.komponenta ().setForeground (Color.red);
}

synchronized void otvori () { // Otvaranje kase.
    radi = true; notify ();
    if (polje != null) polje.komponenta ().setForeground (Color.black);
}

synchronized void uniisti () { // Završetak niti.
    interrupt ();
}

void staviURed (Kupac kupac) { // Stavljanje kupca u red.
    red.stavi (kupac);
}

int id () { return id; } // Dohvatanje identifikatora kase.
}

```

```

// Samoposluga.java - Klasa samoposluga.

package samoposluga;
import usluge.Polje; → Zadatak 8.5
import java.awt.Color;

public class Samoposluga {

    private static int ukId = 0; // Poslednje korišćeni identifikator.
    private int id = ++ukId; // Identifikator samoposluge.
    private Uzorak[] ulazi; // Niz ulaza samoposluge.
    private Kasa[] kase; // Niz kasa samoposluge.
    private Polje polje; // Polje za prikaz stanja samoposluge.
    private int brojKupaca = 0; // Broj kupaca u samoposluži.
    private boolean radi = false; // Da li je samoposluga otvorena?
}

```

```

public Samoposluga {                                     // Inicijalizacija:
    int brUlaza, int brKasa, int maxUlaz, int maxKupovina, int maxNaplata,
    Polje p, Polje[] pUlazi, Polje[] pKase, Polje[] pRedovi {
        if ((polje = p) != null) {
            polje.komponenta().setForeground (Color.red);
            polje.pisi ("Kupaca: 0");
        }
        ulazi = new Ulaz [brUlaza];                      // - stvaranje ulaza,
        for (int i=0; i<brUlaza; i++) {
            ulazi[i] = new Ulaz (this, maxUlaz, maxKupovina,
                (pUlazi!=null ? pUlazi[i] : null));
        }
        kase = new Kasa [brKasa];                         // - stvaranje kasa.
        for (int i=0; i<brKasa; i++)
            kase[i] = new Kasa (this, maxNaplata,
                (pKase!=null ? pKase[i] : null),
                (pRedovi!=null ? pRedovi[i] : null));
    }

    public void otvori () {                            // Otvaranje samoposluge.
        if (polje != null) polje.komponenta().setForeground (Color.black);
        radi = true;
        for (int i=0; i<ulazi.length; ulazi[i++].otvori());
    }

    public void zatvori () {                           // Zatvaranje samoposluge.
        if (polje != null) polje.komponenta().setForeground (Color.red);
        radi = false;
        for (int i=0; i<ulazi.length; ulazi[i++].zatvori());
        synchronized (this) {
            while (brojKupaca > 0)
                try { wait (); } catch (InterruptedException g) {}
        }
    }

    synchronized void usaoKupac () {      // Registrovanje ulaska kupca.
        brojKupaca++;
        if (polje != null) polje.pisi ("Kupaca: " + brojKupaca);
    }

    synchronized void izasaoKupac () {     // Registrovanje izlaska kupca.
        if (--brojKupaca == 0 && !radi) notify ();
        if (polje != null) polje.pisi ("Kupaca: " + brojKupaca);
    }

    public int id () { return id; } // Dohvatanje identifikatora samoposluge.

    Kasa[] dohvatiKase () { return kase; } // Dohvatanje kasa samoposluge.

    public void uništi () {               // Uništavanje samoposluge.
        if (radi) zatvori ();
        for (Ulaz u: ulazi) u.uništi();
        for (Kasa k: kase) k.uništi();
    }
}

```

**Samoposluga 1**

```

// SamoposlugaT.java - Ispitivanje
// rada samoposluge.

import samoposluga.Samoposluga;
import usluge.*;
import java.awt.*;
import java.awt.event.*;

public class SamoposlugaT extends Frame {

    private Samoposluga samoposl; // Samoposluga koja se obradjuje.
    private Button dgmOtvori = new Button ("Otvori"), // Upravljačka
                  dgmZatvori = new Button ("Zatvori"), // dugmad.
                  dgmUnisti = new Button ("Unisti");
    private boolean smeZatv = true; // Da li sme prozor da se zatvara?

    private void omoguci (boolean da) { // Podešavanje upotrebljivosti
        dgmOtvori.setEnabled (da); // upravljačke dugmadi.
        dgmZatvori.setEnabled (da);
        dgmUnisti.setEnabled (da);
        smeZatv = da;
    }

    private void ploUlazi (int brUlaza, Polje[] pUlazi) { // Ploča za ulaze.
        Panel plo = new Panel (new GridLayout(0,1));
        add (plo, "West");
        plo.setBackground (new Color(192, 255, 192));
        try {
            for (int i=0; i<brUlaza; i++) {
                Label oznaka = new Label ();
                plo.add (oznaka);
                pUlazi[i] = new Polje(oznaka);
            }
        } catch (GKompNeOdgovara g) {}
    }

    // Ploča za kase i redove.
    private void ploKaseRedovi (int brKasa, Polje[] pKase, Polje[] pRedovi) {
        Panel plo = new Panel (new BorderLayout ());
        add (plo, "Center");
        Panel ploKase = new Panel (new GridLayout (1, brKasa)),
              ploRedovi = new Panel (new GridLayout (1, brKasa));
        plo.add (ploKase, "North"); plo.add (ploRedovi, "Center");
        ploKase.setBackground (new Color(192, 192, 255));
        ploRedovi.setBackground (new Color(192, 192, 192));
    }
}

```

```

try {
    for (int i=0; i<brKasa; i++) {
        Label oznaka = new Label ();
        ploKase.add (oznaka);
        pKase[i] = new Polje (oznaka);
        TextArea tekst = new TextArea ();
        ploRedovi.add (tekst);
        pRedovi[i] = new Polje (tekst);
    }
} catch (GKompNeOdgovara g) {}

private void ploKomande (Polje[] pKupci) { // Ploča za upravljačku dugmad.
    Panel plo = new Panel();
    add (plo, "North");
    plo.setBackground (new Color(255, 192, 192));
    plo.add (dgmOtvari );
    plo.add (dgmZatvori);
    plo.add (dgmUnisti );
    DugmeAkcija akcija = new DugmeAkcija ();
    dgmOtvari.addActionListener (akcija);
    dgmZatvori.addActionListener (akcija);
    dgmUnisti.addActionListener (akcija);
    Label oznKupci = new Label ();
    plo.add (oznKupci);
    try { pKupci[0] = new Polje (oznKupci); }
    catch (GKompNeOdgovara g) {}
}

// Obrada pritisaka na upravljačku dugmad.
class DugmeAkcija implements ActionListener {
    public void actionPerformed (ActionEvent d) {
        Component k = (Component)d.getSource ();
        if (k == dgmOtvari ) samoposl.otvori ();
        if (k == dgmZatvori) new Zatvori ().start ();
        if (k == dgmUnisti ) new Unisti ().start ();
    }
}

private class Zatvori extends Thread { // Nit za zatvaranje samoposluge.
    public void run () {
        omoguci (false); samoposl.zatvori (); omoguci (true); }
}

private class Unisti extends Thread { // Nit za uništavanje samoposluge.
    public void run () {
        omoguci (false); samoposl.unisti (); dispose (); }
}

private void popuniProzor (String[] vpar) { // Popunjavanje prozora i
    // stvaranje samoposluge:
    int brUlaza = Integer.parseInt (vpar[1]),
        brKasa = Integer.parseInt (vpar[2]);

    Polje[] pUlazi = new Polje [brUlaza]; // - ploča za ulaze,
    ploUlazi(brUlaza, pUlazi);

    Polje[] pKase = new Polje [brKasa]; // - ploča za kase i redove,
    Polje[] pRedovi = new Polje [brKasa];
    ploKaseRedovi(brKasa, pKase, pRedovi);
}

```

Dugotrajne radnje ne treba izvršavati unutar osmatrača, jer za to vreme nijedan drugi događaj grafičke korisničke površi (na primer, pomeranje miša) ne može da se obradi.

Za takve radnje potrebno je stvarati zasbne niti.

```

Polje[] pKupci = new Polje [1];           // - ploča za komandnu dugmad,
ploKomande(pKupci);

samoposl = new Samoposluga (           // - stvaranje samoposluge.
    brUlaza, brKasa,
    Integer.parseInt (vpar[3]),          //   (maxUlaz)
    Integer.parseInt (vpar[4]),          //   (maxKupovina)
    Integer.parseInt (vpar[5]),          //   (maxNaplata)
    pKupci[0], pUlazi, pKase, pRedovi
);
}

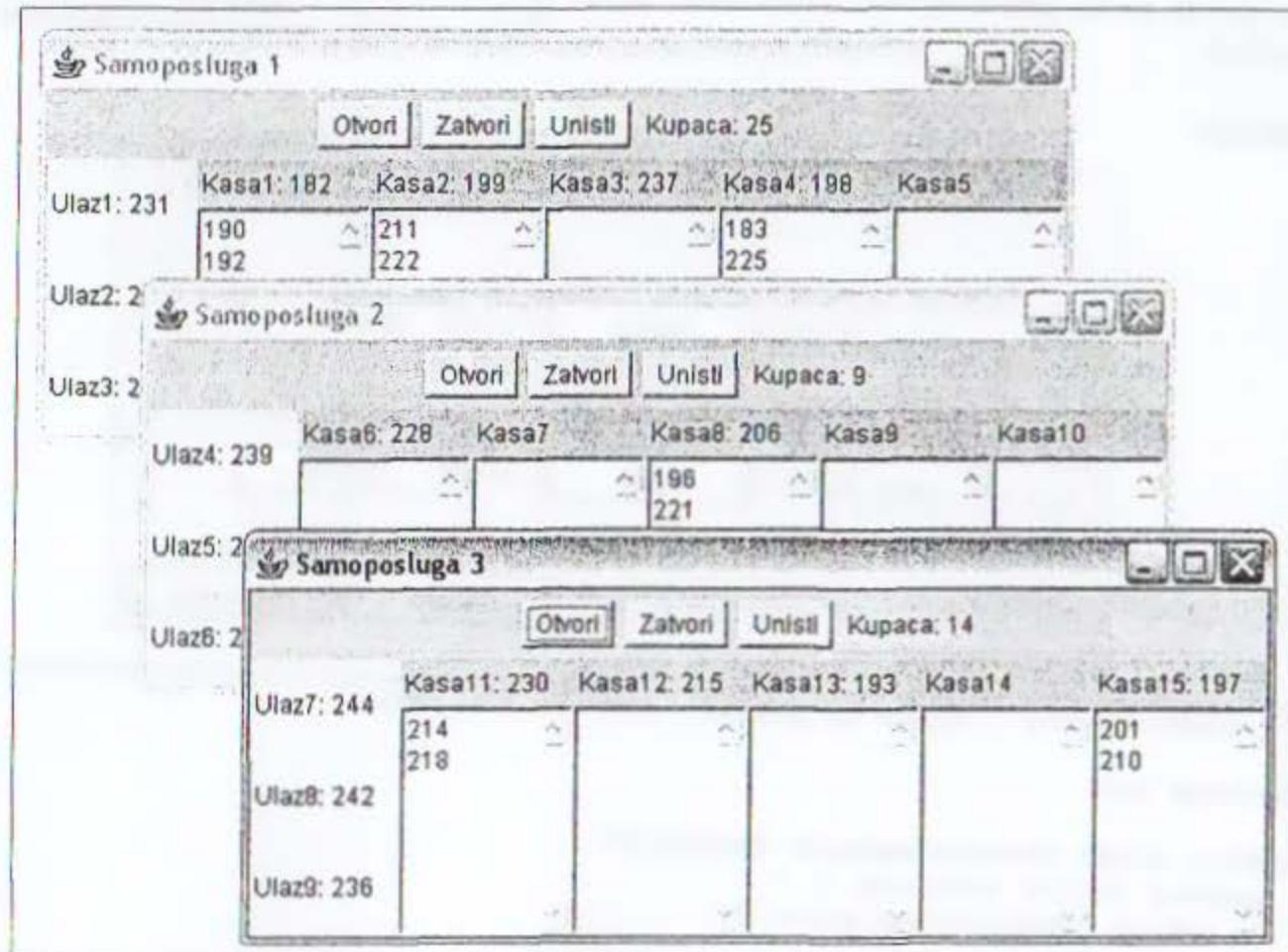
private SamoposlugaT (String[] vpar, int x, int y) { // Inicijalizacija:
    setBounds (x, y, 500, 200);
    popuniProzor (vpar);                // - popunjavanje prozora,
    setTitle ("Samoposluga " + samoposl.id()); // - postavljanje natpisa,
    addWindowListener (new WindowAdapter () { // - obrada zatvaranja
        public void windowClosing (WindowEvent d) // prozora.
            { if (smeZatv) new Unisti ().start (); } } );
}
}

public static void main (String[] vpar) { // Glavna funkcija.
    for (int i=1; i<=Integer.parseInt (vpar[0]); i++)
        new SamoposlugaT (vpar, 50*i, 120*i).setVisible (true);
}
}

```

Nit za dugotrajnu radnju.

```
% java SamoposlugaT 3 3 5 1500 2000 2000
```

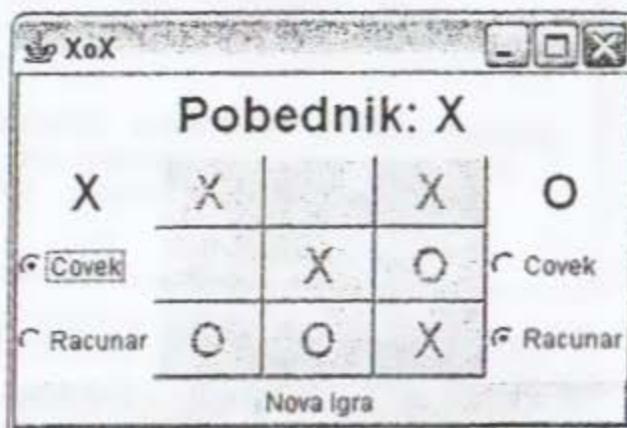


### Zadatak 8.15 Igra XoX (klase tabli, igrača, čoveka, računara i igara)

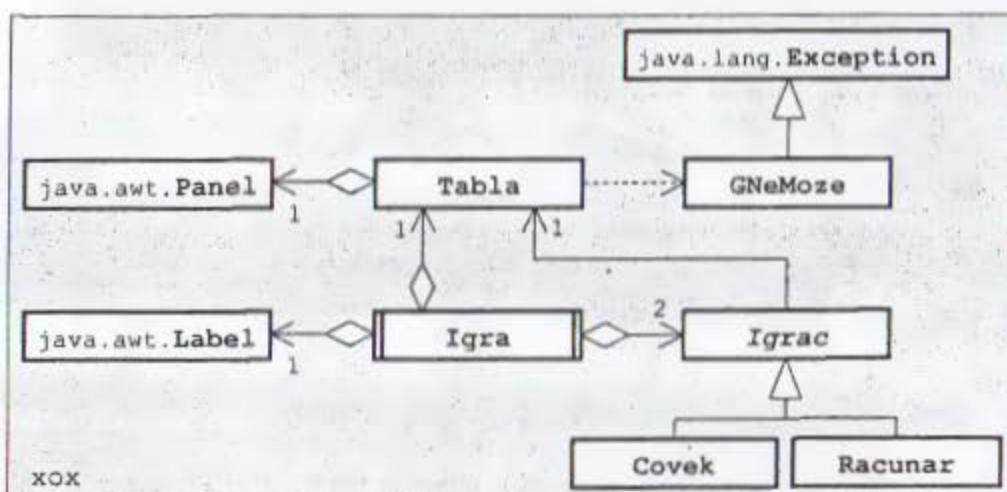
Napisati na jeziku Java sledeće klase za ostvarivanje igre XoX:

- *Tabla* za igru XoX sadrži grafičku ploču (*Panel*) na kojoj prikazuje  $3 \times 3$  polja. Polja mogu biti prazna ili mogu da sadrže oznaku igrača koji je zauzeo dato polje. Može da se postavi oznaka nekog od igrača na dato polje (greška je ako to polje nije prazno), da se dohvati oznaka na datom polju, da se isprazne oznake svih polja i da se zatraži indeks polja koje je pritisnuto na grafičkoj korisničkoj površi. Nijedno polje ne sme biti osetljivo na pritisak sve dok se ne zatraži dohvatanje indeksa pritisnutog polja, a kad se to zatraži, mogu da se pritisnu samo prazna polja. Tabla treba da obezbedi da onaj ko traži indeks pritisnutog polja čeka (bez trošenja procesorskog vremena) dok se neko polje ne pritisne.
- Apstraktan *igrač* ima jednoslovnu oznaku i poteze izvodi na zadatoj tabli. Izvođenje poteza se sastoji od izbora polja i pokušaja postavljanja svoje oznake na to polje, sve dok postavljanje oznake ne uspe.
- *Čovek* je igrač koji polja bira tako da od table zatraži indeks pritisnutog polja.
- *Računar* je igrač koji nasumice bira indeks polja.
- Aktivna *igra* upravlja igrom dva igrača na zadatoj tabli i prikazuje oznaku pobednika na zadatoj grafičkoj komponenti tipa *Label*. Igrači naizmenično stavljuju svoje oznake na slobodna polja. Pobednik je onaj koji prvi zauzme tri uzastopna polja u nekoj vrsti, koloni ili na nekoj dijagonali table. Igra je nerešena ako se tabla popuni a nijedan igrač nije zauzeo tri uzastopna polja.

Napisati na jeziku Java program s grafičkom korisničkom površi, prema priloženoj slici, kojim može da se igra XoX.



**Rešenje:**



```
// GNeMoze.java - Klasa za greške: Nedozvoljen potez.
```

```
package xox;

public class GNeMoze extends Exception {
    public String toString () {
        return "Nedozvoljen potez!";
    }
}
```

```

// Tabla.java - Klasa tabli za igru XoX.

package xox;
import java.awt.*;
import java.awt.event.*;

public class Tabla {
    private Panel ploca;                                // Komponenta za prikaz table.
    private Dugme[] dugmad = new Dugme [9];           // Polja table.
    private int pritisnuto;                            // Pritisnuto polje.

    private static class Dugme extends Button { // Polje s pridruženim
        private int indeks;                      // indeksom u tabli.
        public Dugme (int ind) { super (" "); indeks = ind; }
    }

    public Tabla (Panel plo) {                         // Inicijalizacija:
        (ploca=plo).setLayout(new GridLayout(3,3)); // - ploča za polja,
        ploca.setEnabled (false);
        Font font = new Font (null, Font.BOLD, 24); // - vrsta slova za natpise,
        DugmeAkcija osmatrac = new DugmeAkcija (); // - dugmad za polja.
        for (int i=0; i<9; i++) {
            ploca.add (dugmad[i] = new Dugme (i));
            dugmad[i].setFont (font);
            dugmad[i].addActionListener (osmatrac);
        }
    }

    public synchronized int uzmiPritisnuto () throws InterruptedException {
        ploca.setEnabled (true);                  // Dohvatanje indeksa pritisnutog
        wait ();                                // polja.
        ploca.setEnabled (false);
        return pritisnuto;
    }                                              // Obrada pritisaka polja.

    private class DugmeAkcija implements ActionListener {
        public void actionPerformed (ActionEvent d) {
            pritisnuto = ((Dugme)d.getSource()).indeks;
            dalje ();
        }
    }

    private synchronized void dalje () // Dojava da je polje pritisnuto.
    { notify (); }                                // Postavljanje oznake na polje.

    void postaviOznaku (int ind, String oznaka) throws GNeMoze {
        if (ind<0 || ind>=9 || !dugmad[ind].getLabel().equals(" "))
            throw new GNeMoze ();
        dugmad[ind].setLabel (oznaka); dugmad[ind].setEnabled (false);
    }

    void prazni () {                             // Pražnjenje table.
        for (int i=0; i<9; i++)
            { dugmad[i].setLabel (" "); dugmad[i].setEnabled (true); }
    }

    public String oznaka (int ind) { // Dohvatanje oznake polja.
        return dugmad[ind].getLabel ();
    }
}

```

```
// Igrac.java - Klasa apstraktnih igrača.

package xox;

public abstract class Igrac {
    private String oznaka; // Oznaka igrača.
    protected Tabla tabla; // Tabla na kojoj se igra.

    public Igrac (String ozn, Tabla tbl) { // Inicijalizacija.
        oznaka = ozn; tabla = tbl;
    }
        // Biranje polja.
    public abstract int birajPolje () throws InterruptedException;
        // Izvođenje poteza.
    public void odigrajPotez () throws InterruptedException {
        while (true) {
            try { tabla.postaviOznaku (birajPolje(), oznaka); return; }
            catch (GNeMoze g) {}
        }
    }
}
```

```
// Covek.java - Klasa ljudskih igrača.

package xox;

public class Covek extends Igrac { // Inicijalizacija.
    public Covek (String ozn, Tabla tbl) { super (ozn, tbl); }

    public int birajPolje () throws InterruptedException { // Biranje polja.
        return tabla.uzmiPritisnuto ();
    }
}
```

```
// Racunar.java - Klasa računarskih igrača.

package xox;

public class Racunar extends Igrac { // Inicijalizacija.
    public Racunar (String ozn, Tabla tbl) { super (ozn, tbl); }

    public int birajPolje () { // Biranje polja.
        return (int) (Math.random() * 9);
    }
}
```

```

// Igra.java - Klasa igara Xox-a.

package xox;
import java.awt.*;

public class Igra extends Thread {
    private Igrac[] igraci = new Igrac [2]; // Igrači.
    private int tekIgrac; // Igrač na potezu.
    private Tabla tabla; // Korišćena tabla.
    private Label oznStanje; // Stanje igre.
    private String inicijalizacija; // Inicijalizacija.

    public Igra (Tabla tbl, Igrac prvi, Igrac drugi, Label sta) {
        igraci[0] = prvi; igraci[1] = drugi;
        tabla = tbl;
        tabla.prazni ();
        tekIgrac = 0;
        oznStanje = sta;
        oznStanje.setText (" ");
        start ();
    }

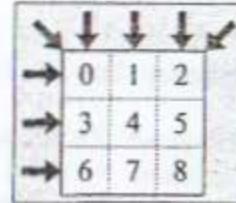
    private boolean isto (int a, int b, int c) { // Da li su tri neprazne
        return !tabla.oznaka(a).equals(" ") && // jednake oznake?
            tabla.oznaka(a).equals(tabla.oznaka(b)) &&
            tabla.oznaka(a).equals(tabla.oznaka(c));
    }

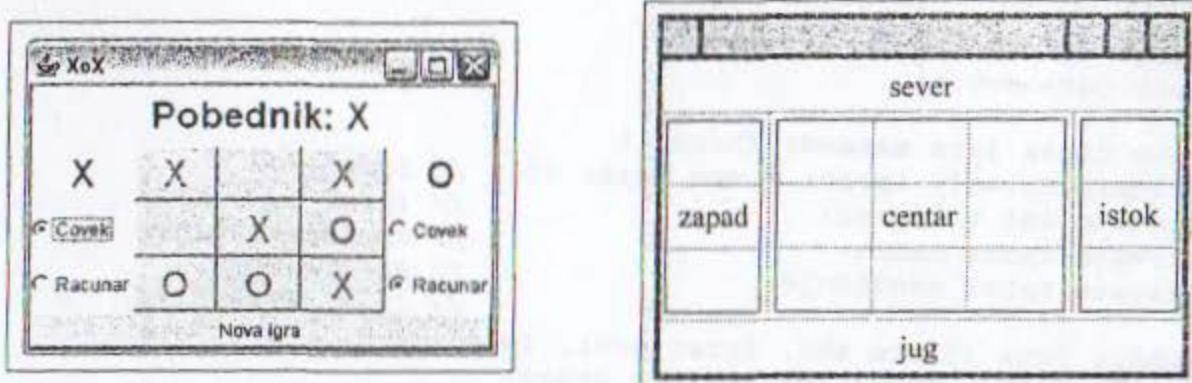
    public String stanje () { // Određivanje stanja
        if (isto (0, 1, 2)) return tabla.oznaka(0); // igre.
        if (isto (3, 4, 5)) return tabla.oznaka(3);
        if (isto (6, 7, 8)) return tabla.oznaka(6);
        if (isto (0, 3, 6)) return tabla.oznaka(0);
        if (isto (1, 4, 7)) return tabla.oznaka(1);
        if (isto (2, 5, 8)) return tabla.oznaka(2);
        if (isto (0, 4, 8)) return tabla.oznaka(0);
        if (isto (2, 4, 6)) return tabla.oznaka(2);
        for (int i=0; i<9; i++)
            if (tabla.oznaka(i).equals(" ")) return " ";
        return "?";
    }

    public void run () { // Telo niti.
        String stanje = " ";
        try {
            while (!interrupted() && stanje.equals(" "))
                igraci[tekIgrac].odigrajPotez ();
            tekIgrac = 1 - tekIgrac;
            stanje = stanje();
        }
        oznStanje.setText ("Pobednik: " + stanje);
    } catch (InterruptedException g) {}

    public void prekini () { interrupt (); } // Prekidanje igre.
}

```





// Xox.java - Organizovanje igre XoX.

```

import xox.*;
import java.awt.*;
import java.awt.event.*;

public class Xox extends Frame {

    private Font font = new Font (null, Font.BOLD, 24); // Font za natpise.

    private class PloIgrac extends Panel { // PLOČA ZA IZBOR IGRAČA:
        // Radio dugmad za izbor vrste igrača.
        private CheckboxGroup grupa = new CheckboxGroup ();
        private Checkbox rdgCovek = new Checkbox ("Covek", true, grupa),
                    rdgRacunar = new Checkbox ("Racunar", false, grupa);
        private String oznaka; // Oznaka igrača.
        private Igrac igrac; // Pridruženi igrač.

        PloIgrac (String ozn) {
            // Inicijalizacija:
            oznaka = ozn; // - oznaka igrača,
            Label lbl = new Label (ozn, Label.CENTER);
            lbl.setFont (font);
            ItemListener osmatrac = new ItemListener () { // - osmatrač za izbor
                public void itemStateChanged (ItemEvent d) { // vrste igrača,
                    if ((Checkbox)d.getSource() == rdgCovek)
                        igrac = new Covek (oznaka, tabla);
                    else
                        igrac = new Racunar (oznaka, tabla);
                }
            };
            rdgCovek.addItemListener (osmatrac);
            rdgRacunar.addItemListener (osmatrac);
            setLayout (new GridLayout (3,1)); // - popunjavanje ploče,
            add (lbl); add (rdgCovek); add (rdgRacunar);
            igrac = new Covek (oznaka, tabla); // - početni igrač.
        }
    } // class ploIgrac

    private Panel ploTabla = new Panel (); // Ploča za polja.
    private Tabla tabla = new Tabla (ploTabla); // Tabla za igru.
    private PloIgrac ploPrvi = new PloIgrac ("X"); // Ploče za izbor
    private PloIgrac ploDrugi = new PloIgrac ("O"); // vrste igrača.
    private Igra igra; // Igra koja se igra.
    private Label stanje = new Label ("", Label.CENTER); // Natpis stanja igre.
}

```

```

public void popuniProzor () {                                // Popunjavanje prozora:
    add (ploTabla, "Center");                                // - ploča za tablu,
    add (ploPrvi , "West" );                                 // - ploče za izbor
    add (ploDrugi, "East" );                                // vrste igrača,
    Button dugme = new Button ("Nova igra");                // - dugme za početak
    dugme.addActionListener (new ActionListener (){ // nove igre,
        public void actionPerformed (ActionEvent d) {
            if (igra != null) igra.prekini ();
            igra = new Igra (tabla, ploPrvi.igrac, ploDrugi.igrac, stanje);
        }
    });
    add (dugme, "South");
    stanje.setFont (font);                                    // - natpis stanja igre.
    add (stanje, "North");
}

public Xox () {                                            // Inicijalizacija:
    super ("XoX");
    setBounds (100, 100, 300, 200);                          // - popunjavanje prozora,
    popuniProzor ();                                         // - popunjavanje prozora,
    setVisible (true);

    addWindowListener (new WindowAdapter () {               // - obrada zatvaranja
        public void windowClosing (WindowEvent d) { // prozora.
            if (igra != null) igra.prekini (); dispose ();
        }
    });
} // konstruktor

public static void main (String[] varg) {                  // GLAVNA FUNKCIJA.
    new Xox ();
}

```

```

public void prikazi (double[] niz) {           // Prikazaivanje niza.
    if (niz != null) {
        this.niz = niz;
        FontMetrics fm = getGraphics ().getFontMetrics ();
        StringBuffer sb = new StringBuffer ();
        int p = 0;
        for (double br: niz) {
            String s = String.format (frm, br);
            int k = fm.stringWidth (s);
            if (p+k >= getWidth ()-30) { sb.append ('\n'); p = 0; }
            sb.append (s); p += k;
        }
        if (p > 0) sb.append ('\n');
        tks.setText (sb.toString ());
    }
}

public String toString () { return "prozor"; } // Naziv prikazivača.
}

```

```

// Histogram.java - Klasa prikazivača nizova u obliku histograma.

package prikazivaci;
import java.awt.*;
import java.awt.event.*;

public class Histogram extends Frame implements Prikazivac {

    private class Platno extends Canvas {           // Klasa platna:
        Platno () {                            // - konstruktor,
            addComponentListener (new ComponentAdapter () { // - obrada promene
                public void componentResized (ComponentEvent d) // veličine
                    { repaint (); } // platna,
            });
        }

        public void paint (Graphics g) {           // - crtanje po
            if (niz != null) {                   // platnu.
                g.setColor (Color.BLACK);
                int sir = getWidth (), vis = getHeight ();
                int n = niz.length;
                double max = niz[0];
                for (int i=1; i<n; i++) if (niz[i] > max) max = niz[i];
                double ds = (double) sir / n, dv = (double) vis / max;
                for (int i=0; i<n; i++) {
                    int x = (int) (i * ds), v = (int) (niz[i] * dv);
                    g.fillRect (x, vis-v, (int)ds-1, v);
                }
            }
        }
    } // class Platno

    private Platno platno = new Platno ();           // Platno po kome se crta.
    private double[] niz;                           // Niz za prikazivanje.
}

```

```

public Histogram (String naslov, int x, int y, // Sastavljanje prozora:
                  int sir, int vis) {
    super (naslov);
    setBounds (x, y, sir, vis);
    add (platno);
    addWindowListener (new WindowAdapter () { // - obrada zatvaranja
        public void windowClosing (WindowEvent d) // prozora.
            { dispose (); }
    });
}

public void prikazi (double[] niz) // Prikazivanje niza.
    { this.niz = niz; platno.repaint (); }

public String toString () { return "histogram"; } // Vrsta prikazivača.
}

```

// SkupPrikazivaca - Klasa skupa prikazivača.

```

package prikazivaci;

public class SkupPrikazivaca implements Prikazivac {

    private class Elem { // Element liste prikazivača.
        Prikazivac prik; Elem sled;
        Elem (Prikazivac p) {
            prik = p;
            if (prvi == null) prvi = this;
            else posl.sled = this;
            posl = this;
        }
    }

    private Elem prvi = null, posl = null; // Početak i kraj liste.

    public void dodaj (Prikazivac p) // Dodavanje prikazivača.
        ( new Elem (p); )

    public void ukloni (Prikazivac p) { // Uklanjanje prikazivača.
        Elek tek = prvi, pret = null;
        while (tek!=null && tek.prik!=p) { pret = tek; tek = tek.sled; }
        if (tek != null) {
            if (pret == null) prvi = tek.sled;
            else pret.sled = tek.sled;
            if (prvi == null) posl = null;
            if (posl == tek) posl = pret;
        }
    }

    public void prikazi (double[] niz) // Prikazivanje niza.
        for (Elem tek=prvi; tek!=null; tek=tek.sled)
            tek.prik.prikazi (niz);

    public String toString () { return "skup"; } // Naziv prikazivača.
}

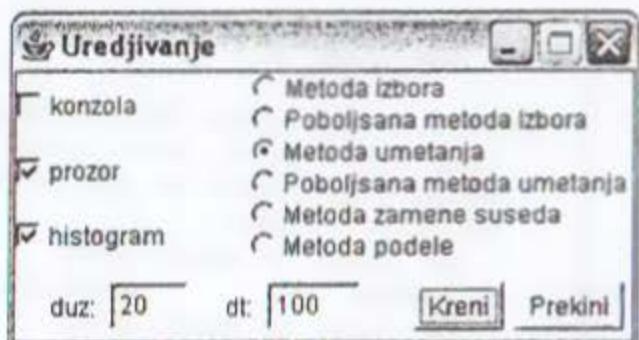
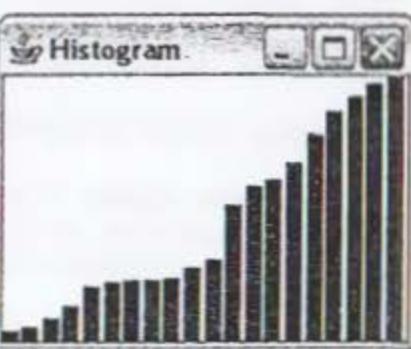
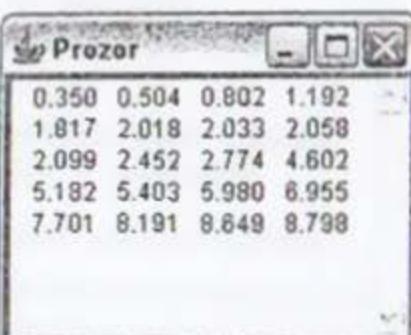
```

**Zadatak 8.16 Prikazivači, konzole, prozori, histogrami, skupovi prikazivača, razne vrste uređivača**

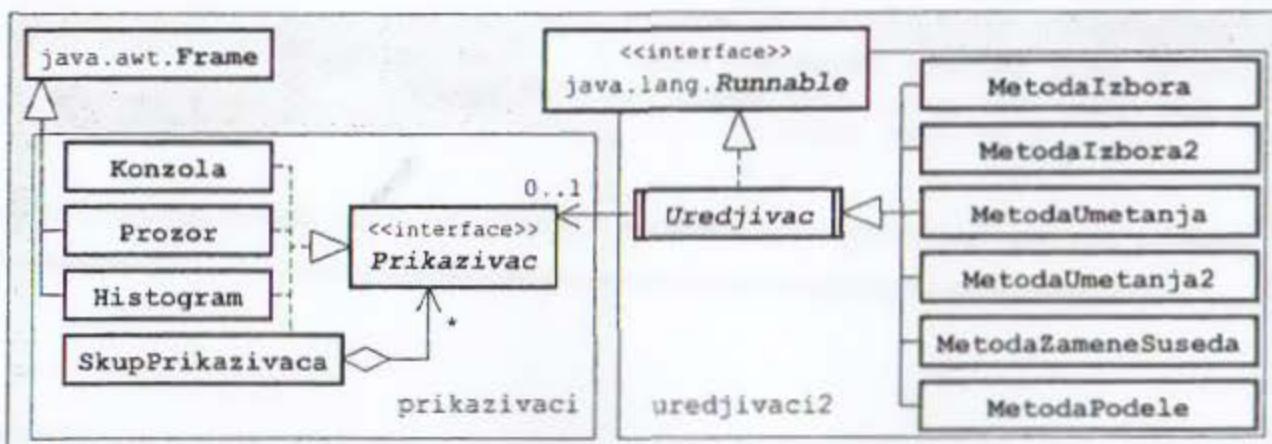
Napisati na jeziku Java sledeće tipove:

- Apstraktan *prikazivač* predviđa prikazivanje niza realnih brojeva. Tekstualni oblik objekta prikazivača predstavlja ime vrste prikazivača.
- *Konzola* je prikazivač koji sadržaj niza ispisuje na glavnom izlazu. U svakom redu ispisuje zadati broj elemenata niza po zadatom formatu.
- *Prozor* je prikazivač koji sadržaj niza ispisuje u prozoru sa zadatim naslovom, koordinatama, širinom, visinom i formatom prikazivanja elemenata niza. U svakom redu prikazuje onoliko elemenata koliko može da stane s obzirom na trenutnu širinu prozora.
- *Histogram* je prikazivač koji sadržaj niza prikazuje u obliku histograma u prozoru sa zadatim naslovom, koordinatama, širinom i visinom. Svaki element niza se prikazuje u obliku popunjeno pravougaonika čija je visina srazmerna vrednosti elementa. Visina pravougaonika za najveći element niza se poklapa s visinom raspoloživog prostora u prozoru. Širina pojedinih pravougaonika je takva da niz u celini stane po širini raspoloživog prostora u prozoru.
- *Skup prikazivača* je prikazivač koji može da sadrži proizvoljan broj prikazivača pomoću kojih zadati niz prikazuje na više načina. Stvara se prazan, posle čega prikazivači mogu da se dodaju i uklanjaju jedan po jedan.
- Apstraktan aktivan *uređivač* predviđa uređivanje niza realnih brojeva uz mogućnost prikazivanja sadržaja niza posle svakog koraka u postupku uređivanja pomoću nekog prikazivača. Korišćeni prikazivač može da se promeni. Ako se ne zada prikazivač sadržaj niza se ne prikazuje. Posle svakog koraka uređivanja prikazuje se trenutni sadržaj niza i napravi se pauza čije trajanje može da se promeni bilo kad u toku uređivanja. Tok uređivanja može da se prekine. Tekstualni oblik objekta uređivača predstavlja naziv primjenjenog algoritma.
- Konkretni uređivači ostvaruju sledeće algoritme uređivanja nizova: *metoda izbora*, *metoda umetanja*, *metoda zamene suseda* i *metoda podele*.

Napisati na jeziku Java program s grafičkom korisničkom površi prema priloženoj slici, koji uređuje nizove realnih brojeva zadatih dužina i pauza između dva koraka u postupku uređivanja. Mogu istovremeno više prikazivača da budu aktivna.



**Rešenje:**



```
// Prikazivac.java - Interfejs prikazivača nizova.

package prikazivaci;

public interface Prikazivac {
    void prikazi (double[] niz);           // Prikazivanje niza.
    String toString ();                   // Naziv vrste prikazivača.
}
```

```
// Konzola.java - Klasa prikazivača nizova na konzoli.

package prikazivaci;

public class Konzola implements Prikazivac {
    private int k;                         // Broj podataka po redu.
    private String frm;                    // Format prikazivanja.

    public Konzola (int k, String frm)     // Inicijalizacija.
        { this.k = k; this.frm = frm; }

    public void prikazi (double[] niz) {    // Prikazivanje niza.
        StringBuffer s = new StringBuffer ("\n");
        int n = niz.length;
        for (int i=0; i<n; i++) {
            s.append (String.format(frm, niz[i]));
            .append ((i%k==k-1 || k==n-1) ? '\n' : ' ');
        }
        System.out.println (s);
    }

    public String toString () { return "konzola"; } // Naziv prikazivača.
}
```

```
// Prozor.java - Klasa prikazivača nizova u prozoru.

package prikazivaci;
import java.awt.*;
import java.awt.event.*;

public class Prozor extends Frame implements Prikazivac {
    private TextArea tks = new TextArea (); // Komponenta za prikaz.
    private String frm;                  // Format prikazivanja.
    private double[] niz;                // Prikazivani niz.

    public Prozor (String naslov, int x, int y,      // Sastavljanje prozora:
                  int sir, int vis, String frm) {
        super (naslov); this.frm = frm;
        setBounds (x, y, sir, vis);
        add (tks);
        tks.addComponentListener (new ComponentAdapter () { // - obrada promene
            public void componentResized (ComponentEvent d) // veličine polja
                { prikazi (niz); }                      // za tekst,
        });
        addWindowListener (new WindowAdapter () {        // - obrada zatvaranja
            public void windowClosing (WindowEvent d) // prozora.
                { dispose (); }
        });
    }
}
```

```

// Uredjivac.java - Apstraktna klasa za algoritme uređivanja.

package uredjivaci2;
import prikazivaci.Prikazivac;

public abstract class Uredjivac implements Runnable {
    protected Prikazivac prik = null;      // Korišćeni prikazivač.
    protected double[] niz;                 // Niz koji se uređuje.
    private boolean prekini = false;        // Indikator prekidanja.
    private long dt = 10;                  // Trajanje pauze.
    private Thread nit = null;             // Nit koja radi.

    protected class GPrekinut           // Klasa za prijavu prekidanja postupka.
        extends Exception {}

    public void postaviPrikazivaca (Prikazivac prik) // Postavljanje
        { this.prik = prik; }                         // prikazivača.

    public void postaviDt (long dt)                // Postavljanje
        { this.dt = dt; }                           // kašnjenja.

    public synchronized void uredi (double[] niz) { // Zahtev za
        this.niz = niz;                          // uređivanje niza.
        (nit = new Thread (this)).start ();
    }

    public void run () {                         // Sadržaj niti.
        prekini = false;
        try { prikazi (); uredi (); } catch (GPrekinut g) {}
        nit = null;
    }

    protected abstract void uredi () throws GPrekinut; // Uređivanje niza.

    public void prekini () {                     // Zahtev za
        prekini = true;                        // prekidanje rada.
        if (nit != null) try { nit.join(); } catch (InterruptedException g) {}
    }

    protected void prikazi () throws GPrekinut { // Prikazivanje niza.
        if (prik != null) prik.prikazi (niz);
        try { Thread.sleep (dt); }
        catch (InterruptedException g) { prekini (); }
        if (prekini) throw new GPrekinut ();
    }

    public abstract String toString ();          // Naziv algoritma.
}

```

```

// MetodaIzbora.java - Klasa za uređivanje metodom izbora
// (Selection Sort).

package uredjivaci2;

public class MetodaIzbora extends Uredjivac {

```

```

public void uredi () throws GPrekinut { // Uredivanje niza.
    int n = niz.length;
    for (int i=0; i<n-1; i++)
        for (int j=i+1; j<n; j++) {
            if (niz[j] < niz[i]) {
                double p = niz[i]; niz[i] = niz[j]; niz[j] = p;
                prikazi ();
            }
        }
    }

    public String toString () // Naziv algoritma.
    { return "Metoda izbora"; }
}

```

```

// MetodaIzbora2.java - Klasa za uređivanje poboljšanom metodom izbora.

package uredjivaci2;

public class MetodaIzbora2 extends Uredjivac {

    public void uredi () throws GPrekinut { // Uredivanje niza.
        int n = niz.length;
        for (int i=0; i<n-1; i++) {
            int m = i;
            for (int j=i+1; j<n; j++)
                if (niz[j] < niz[m]) m = j;
            if (m != i) {
                double p = niz[i]; niz[i] = niz[m]; niz[m] = p;
                prikazi ();
            }
        }
    }

    public String toString () // Naziv algoritma.
    { return "Poboljsana metoda izbora"; }
}

```

```

// MetodaUmetanja.java - Klasa za uređivanje metodom umetanja
// (Insertion Sort).

package uredjivaci2;

public class MetodaUmetanja extends Uredjivac {

    public void uredi () throws GPrekinut { // Uredivanje niza.
        int n = niz.length;
        for (int i=1; i<n; i++) {
            int j = i - 1;
            while (j>=0 && niz[j+1] < niz[j]) {
                double p = niz[j+1]; niz[j+1] = niz[j]; niz[j--] = p;
                prikazi ();
            }
        }
    }

    public String toString () // Naziv algoritma.
    { return "Metoda umetanja"; }
}

```

```
// MetodaUmetanja2.java - Klasa za uređivanje poboljšanom metodom umetanja.

package uredjivaci2;

public class MetodaUmetanja2 extends Uredjivac {

    public void uredi () throws GPrekinut { // Uređivanje niza.
        int n = niz.length;
        for (int i=1; i<n; i++) {
            double p = niz[i];
            int j = i - 1;
            while (j>=0 && p < niz[j])
                {niz[j+1] = niz[j--]; prikazi (); }
            niz[j+1] = p; prikazi ();
        }
    }

    public String toString ()           // Naziv algoritma.
    { return "Poboljsana metoda umetanja"; }
}
```

```
// MetodaZameneSuseda.java - Klasa za uređivanje metodom zamene suseda
// (Bubble Sort).

package uredjivaci2;

public class MetodaZameneSuseda extends Uredjivac {

    public void uredi () throws GPrekinut { // Uređivanje niza.
        int n = niz.length;
        boolean dalje = true;
        for (int i=0; i<n-1 && dalje; i++) {
            dalje = false;
            for (int j=n-2; j>=i; j--)
                if (niz[j+1] < niz[j]) {
                    double p = niz[j+1]; niz[j+1] = niz[j]; niz[j] = p;
                    prikazi ();
                    dalje = true;
                }
        }
    }

    public String toString ()           // Naziv algoritma.
    { return "Metoda zamene suseda"; }
}
```

```
// MetodaPodele.java - Klasa za uređivanje metodom podele
// (Quick Sort).

package uredjivaci2;

public class MetodaPodele extends Uredjivac {

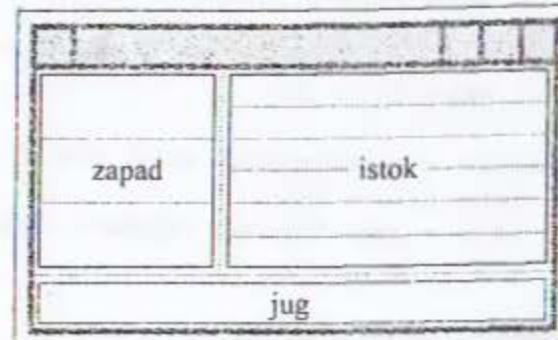
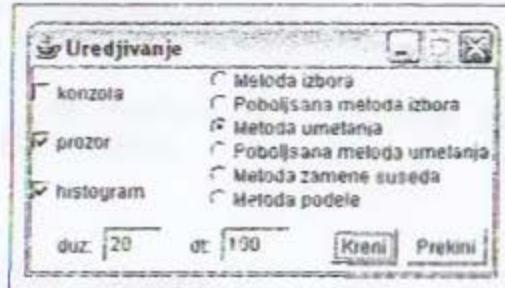
    public void uredi () throws GPrekinut // Uređivanje niza.
    { uredi (0, niz.length-1); }
```

```

private void uredi (int p, int q) throws GPrekinut {
    if (q > p) {
        int i = p-1, j = q;
        while (true) {
            do i++; while (niz[i] < niz[q]);
            do j--; while (j>=0 && niz[j] > niz[q]);
            if (i >= j) break;
            double b = niz[i]; niz[i] = niz[j]; niz[j] = b; prikazi ();
        }
        double b = niz[i]; niz[i] = niz[q]; niz[q] = b; prikazi ();
        uredi (p, i-1); uredi (i+1, q);
    }
}

public String toString () // Naziv algoritma.
{
    return "Metoda podele";
}

```



// PrikazivaciT.java - Ispitivanje klasa prikazivača i uređivača.

```

import prikazivaci.*;
import uredjivaci.*;
import java.awt.*;
import java.awt.event.*;

public class PrikazivaciT extends Frame {
    private Prikazivac[] prik = { // Raspoloživi prikazivači.
        new Konzola (8, "%8.3f"),
        new Prozor ("Prozor" , 400, 50, 200, 160, "%8.3f"),
        new Histogram ("Histogram", 650, 50, 200, 160)
    };
    private Uredjivac[] ured = { // Raspoloživi uređivači.
        new MetodaIzbora (), new MetodaIzbora2 (),
        new MetodaUmetanja(), new MetodaUmetanja2 (),
        new MetodaZameneSuseda (), new MetodaPodele ()
    };
    private TextField // Polja za tekst:
        tksDuz = new TextField ("20"), // - dužina niza,
        tksDt = new TextField ("100"); // - trajanje pauze.
    private SkupPrikazivaca prikazivac = // Skup aktivnih prikazivača.
        new SkupPrikazivaca ();
    private Uredjivac uredjivac = null; // Odabrani uređivac.
    private double[] niz; // Obradivani niz.
}

```

```

    // Klasa polja za potvrdu za izbor prikazivača:
private class Potvrda extends Checkbox {
    private Prikazivac prik;           // - pridruženi prikazivač.

    public Potvrda (Prikazivac p, boolean st) { // - inicijalizacija,
        super (p.toString(), st); prik = p;
        if (st) prikazivac.dodaj (p);
        if (p instanceof Frame) ((Frame)p).setVisible (st);
        addItemListener (new ItemListener () {           // - obrada promene
            public void itemStateChanged (ItemEvent d) { // stanje polja.
                boolean st = getState ();
                if (prik instanceof Frame) ((Frame)prik).setVisible (st);
                if (st) prikazivac.dodaj (prik);
                else prikazivac.ukloni (prik);
            }
        });
    }
} // class Potvrda

private CheckboxGroup grupa =           // Grupa radio dugmadi.
    new CheckboxGroup ();

                                // Klasa za radio dugmad za izbor uređivača:
private class Radio extends Checkbox {
    private Uredjivac ured;           // - pridruženi uređivač,

    public Radio (Uredjivac u, boolean st) { // - inicijalizacija.
        super (u.toString(), grupa, st); ured = u;
    }
} // class Radio

private void popuniProzor () {          // Popunjavanje prozora:
    Panel plo = new Panel (new GridLayout (0, 1)); // - ploča za izbor
    add (plo, "West");                      // prikazivača,
    for (Prikazivac p: prik)
        plo.add (new Potvrda (p, true));

    add (plo = new Panel (new GridLayout (0, 1)), // - ploča za izbor
         "East");                         // uređivača,
    boolean st = true;
    for (Uredjivac u: ured)
        plo.add (new Radio (u, st)); st = false;

    plo = new Panel (); add (plo, "South");      // - ploča za upravljanje
    plo.add (new Label ("duz:", Label.RIGHT));   // (dužina niza)
    plo.add (tksDuz);
    plo.add (new Label ("dt:", Label.RIGHT));    // (trajanje pauze)
    plo.add (tksDt);
    plo.add (new Label ());
    tksDt.addTextListener (new TextListener () { // (obrada promene
        public void textValueChanged (TextEvent d) { // trajanja pauze)
            try {
                uredjivac.postaviDt (Long.parseLong (tksDt.getText()));
            } catch (NumberFormatException g) {}
        }
    });
}

```

```

Button dgm = new Button ("Kreni");           // (dugme za početak
plo.add (dgm);                            // uređivanja)
dgm.addActionListener (new DugmeAkcija());

dgm = new Button ("Prekini"); plo.add (dgm); // (dugme za
dgm.addActionListener (new ActionListener () { // prekidanje).
    public void actionPerformed (ActionEvent d) {
        if (uredjivac != null) uredjivac.prekini ();
    }
});

private PrikazivaciT () {                   // Inicijalizacija:
    super ("Uredjivanje");
    setBounds (50, 50, 300, 160);
    setResizable (false);
    popuni prozor ();                      // - popunjavanje prozora
    addWindowListener (new WindowAdapter () { // - obrada zatvaranja
        public void windowClosing (WindowEvent d) { // prozora,
            if (uredjivac != null) uredjivac.prekini ();
            for (int i=0; i<prik.length; i++)
                if (prik[i] instanceof Frame)
                    ((Frame)prik[i]).dispose ();
            dispose ();
        }
    });
}

// Klasa rukovaoca dogadajima pri pokretanju uređivanja.
private class DugmeAkcija implements ActionListener {
    public void actionPerformed (ActionEvent d) {
        if (uredjivac != null) uredjivac.prekini ();
        uredjivac = ((Radio)grupa.getSelectedCheckbox()).ured;
        uredjivac.postaviPrikazivaca (prikazivac);
        uredjivac.postaviDt (Long.parseLong (tksDt.getText()));
        niz = new double [Integer.parseInt(tksDuz.getText())];
        for (int i=0; i<niz.length; niz[i++]=(Math.random()*10));
        uredjivac.uredi (niz);
    }
} // class DugmeAkcija

public static void main (String[] varg) {      // Glavna funkcija.
    new PrikazivaciT ().setVisible (true);
}
}

```

**Zadatak 8.17 Crtanje ping-pong loptice u apletu**

Napisati na jeziku Java aplet koji pravolinijski pomera lopticu datog poluprečnika unutar svoje grafičke površi. Sa ivica površi loptica se odbija elastično.

**Rešenje:**

```
// PingPong.java - Crtanje ping-pong loptice u apletu.

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

public class PingPong extends Applet implements Runnable {

    private Thread nit = new Thread (this); // Nit apleta.
    private boolean radi = false; // Da li nit treba da radi?
    private int sir, vis,
        r, // Poluprečnik loptice.
        x, y, // Koordinate centra loptice.
        dx, dy, // Korak duž x i y-ose.
        dt; // Vreme između dva koraka.
    private Color boja; // Boja loptice.

    public void init () { // Inicijalizacija apleta:
        sir = Integer.parseInt (getParameter ("width")); // - dohvatanje
        vis = Integer.parseInt (getParameter ("height")); // parametara,
        dx = Integer.parseInt (getParameter ("dx"));
        dy = Integer.parseInt (getParameter ("dy"));
        r = Integer.parseInt (getParameter ("r"));
        int b = Integer.parseInt (getParameter("boja"), 16);
        boja = new Color (b>>>16, (b>>>8)&255, b&255);
        dt = Integer.parseInt (getParameter ("dt"));
        nit.start (); // - pokretanje niti.
    }

    public synchronized void start () // Aktiviranje apleta.
    { radi = true; notify (); }

    public void stop () { radi = false; } // Deaktiviranje apleta.

    public void destroy () { nit.interrupt (); } // Uništavanje apleta.

    public void run () { // Telo niti:
        try {
            while (!nit.interrupted ()) {
                if (!radi)
                    synchronized (this) { if (!radi) wait (); } // - pauziranje,
                if (dx>0 && x+dx+2*r>=sir || // - promena smera duž x-ose,
                    dx<0 && x+dx<0 ) dx = -dx;
                if (dy>0 && y+dy+2*r>=vis || // - promena smera duž y-ose,
                    dy<0 && y+dy<0 ) dy = -dy;
                x += dx; // - nove koordinate,
                y += dy; // - ponovno iscrtavanje,
                repaint (); // - čekanje do narednog koraka.
                nit.sleep (dt);
            }
        } catch (InterruptedException g) {}
    }
}
```

```

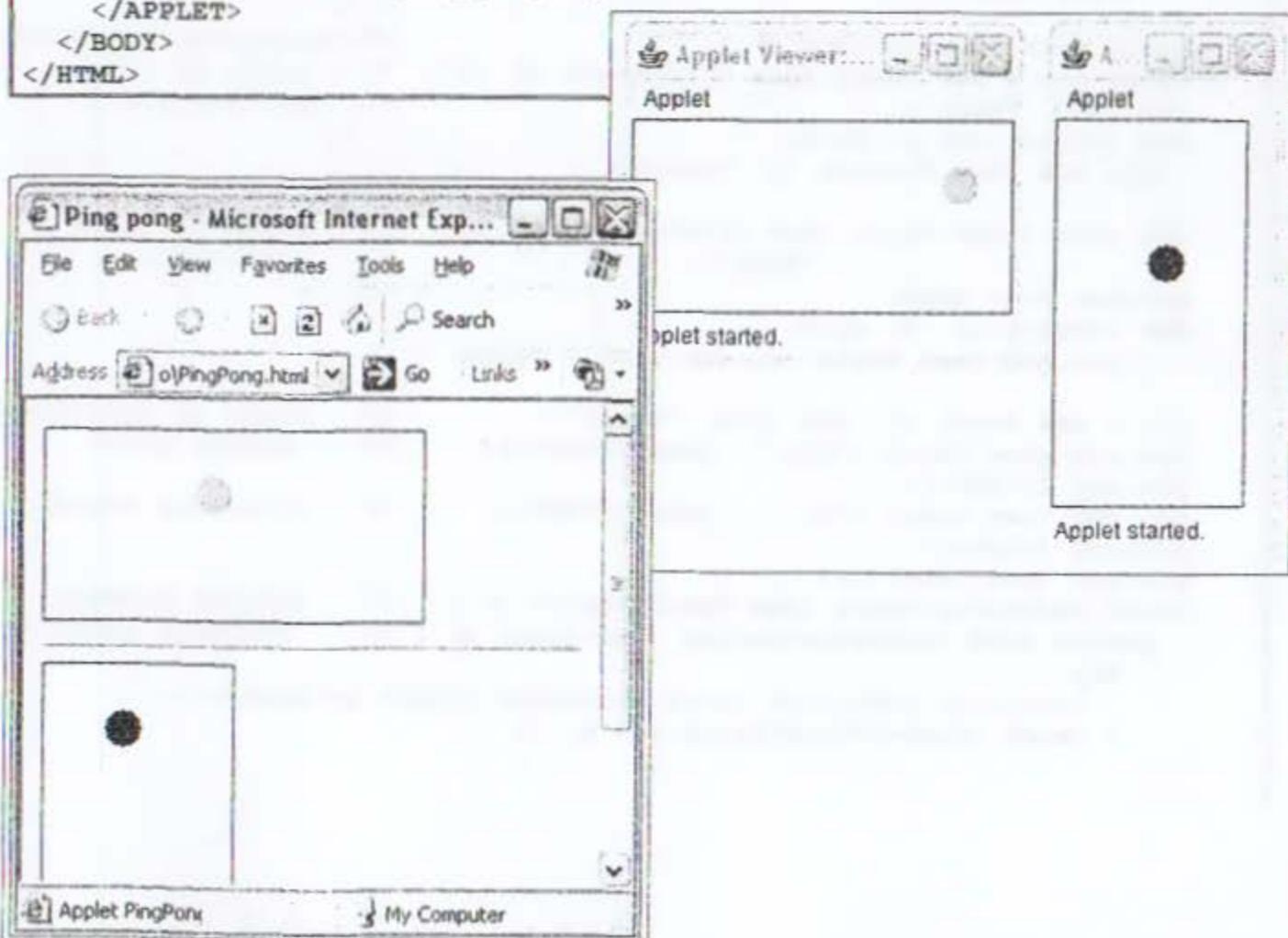
public void paint (Graphics g) {           // Isrtavanje apleta.
    g.setColor (Color.RED);
    g.drawRect (0, 0, sir-1, vis-1);
    g.setColor (boja);
    g.fillOval (x, y, 2*r, 2*r);
}
}

```

```

<HTML>
<HEAD><TITLE>Ping pong</TITLE></HEAD>
<BODY>
<OBJECT CODETYPE="application/java"
        CODE="PingPong.class" WIDTH=200 HEIGHT=100>
<PARAM NAME="dx"      VALUE="3">
<PARAM NAME="dy"      VALUE="2">
<PARAM NAME="r"       VALUE="10">
<PARAM NAME="boja"    VALUE="c0c0c0">
<PARAM NAME="dt"      VALUE="10">
    Ovaj aplet crta ping-pong lopticu.
</OBJECT>
<HR>
<!-- Zastareli način umetanja apleta. -->
<APPLET CODE="PingPong.class" WIDTH=100 HEIGHT=200>
<PARAM NAME="dx"      VALUE="4">
<PARAM NAME="dy"      VALUE="4">
<PARAM NAME="r"       VALUE="10">
<PARAM NAME="boja"    VALUE="ff">
<PARAM NAME="dt"      VALUE="20">
    Ovaj aplet crta ping-pong lopticu.
</APPLET>
</BODY>
</HTML>

```



```
// SrVred2.java - Srednja vrednost brojeva u zapisima tekstualne datoteke.

import usluge.Citaj;
import java.io.*;

public class SrVred2 {
    public static void main (String[] varg) {
        try {
            Citaj ul = new Citaj ("SrVred2.pod");
            PrintStream izl = new PrintStream ("SrVred2.rez");
            double z = 0;
            while (true) {
                int n = ul.IntS ();
                if (ul.endS ()) break;
                double [] a = new double [n];
                double s = 0;
                for (int i=0; i<n; i++) s += (a[i] = ul.DoubleS ());
                z += (s /= n);
                if (s > 0) {
                    izl.print (n);
                    for (int i=0; i<n; izl.print(" " + a[i++]));
                    izl.println ();
                }
            }
            System.out.println ("Zbir srednjih vrednosti= " + z);
        } catch (FileNotFoundException g) {
            System.out.println (g);
        }
    }
}
```

**SrVred2.pod**

```
6 1 2 3 4 5 6
8 -1 -2 -3 -4 -5 -6 -7 -8
5 4 -2 5 -7 1
7 1 -2 3 -4 5 -6 7
4 4 -5 6 -7
```

**# java SrVred2**

```
Zbir srednjih vrednosti= -0.7285714285714286
```

**SrVred2.rez**

```
6 1.0 2.0 3.0 4.0 5.0 6.0
5 4.0 -2.0 5.0 -7.0 1.0
7 1.0 -2.0 3.0 -4.0 5.0 -6.0 7.0
```

### Zadatak 9.2 Obrada rečenica u tekstualnoj datoteci

Napisati na jeziku Java funkciju za prepisivanje sadržaja sekvensijalne tekstualne datoteke u drugu sekvensijalnu tekstualnu datoteku uz pretvaranje prvih slova rečenica u velika, a svih ostalih slova u mala. Kraj rečenice se obeležava tačkom (.), znakom užvika (!) ili znakom pitanja (?).

Napisati na jeziku Java program koji čita parove imena datoteka i poziva prethodnu funkciju sve dok umesto imena jedne od datoteka ne pročita tri zvezdice (\*\*\*) .

**Rešenje:**

```
// Recenice.java - Sredivanje rečenica.

import java.io.*;
import usluge.Citaj;

public class Recenice {
    private static void recenice (Citaj ul, PrintStream izl) {
        boolean prvi = true;
        while (true) {
            char zn = ul.getChS ();
            if (ul.endS ()) break;
            if (Character.isUpperCase (zn))
                ( if (!prvi) zn = Character.toLowerCase (zn); else prvi = false; )
            else if (Character.isLowerCase (zn))
                ( if (prvi) { zn = Character.toUpperCase (zn); prvi = false; } )
            else if (zn=='.' || zn=='!' || zn=='?')
                prvi = true;
            izl.print (zn);
        }
    }

    public static void main (String[] varg) {
        while (true) {
            System.out.print ("Ime ulazne datoteke? ");
            String ulaz = Citaj.String ();
            if (ulaz.equals ("***")) break;
            System.out.print ("Ime izlazne datoteke? ");
            String izlaz = Citaj.String ();
            if (izlaz.equals ("***")) break;
            try {
                recenice (new Citaj (ulaz), new PrintStream (izlaz));
            } catch (FileNotFoundException g) {
                System.out.println ("*** Datoteka ne postoji!");
            }
        }
    }
}
```

*Recenice.pod*

```
Prva, lepa recenica. dRUGA
ruZNa ReCeNiCa. OVA RECENICA SE
ZAVRSAVA NA KRAJU REDA.
```

```
u ovoj recenici ima i suvisnih
razmaka. DANAS JE 24. FEBRUAR 2005.
```

**\* java Recenice**

```
Ime ulazne datoteke? Recenice.pod
Ime izlazne datoteke? Recenice.rez
Ime ulazne datoteke? abc.pod
Ime izlazne datoteke? abc.rez
*** Datoteka ne postoji!
Ime ulazne datoteke? ***
```

*Recenice.rez*

```
Prva, lepa recenica. Druga
ruzna recenica. Ova recenica se
zavrsava na kraju reda.
```

```
U ovoj recenici ima i suvisnih
razmaka. Danas je 24. Februar 2005.
```

### Zadatak 9.3 Obrada sekvencijalne binarne datoteke

Sekvencijalna binarna datoteka sadrži zapise promenljive dužine čiji je sadržaj jedan celo broj  $n$  i  $n$  realnih brojeva. Napisati na jeziku Java programe za:

- formiranje jedne takve datoteke čitajući podatke iz sekvencijalne tekstualne datoteke;
- ispisivanje sadržaja jedne takve datoteke na glavnom izlazu računara; i
- razvrstavanje zapisa jedne takve datoteke u dve, tako da u jednoj budu zapisi u kojima je srednja vrednost podataka veća od nule, a u drugoj zapisi sa negativnom srednjom vrednošću.

Imena datoteka za obradu programima se dostavljaju kao parametri glavne funkcije.

**Rešenje:**

```
// SekPravi.java - Formiranje sekvencijalne binarne datoteke.

import usluge.Citaj;
import java.io.*;

public class SekPravi {
    public static void main (String[] varg) {
        try {
            Citaj ulaz = new Citaj (varg[0]);
            DataOutputStream izlaz = new DataOutputStream (
                new FileOutputStream (varg[1]));
            while (true) {
                int n = ulaz.IntS ();
                if (ulaz.endS ()) break;
                izlaz.writeInt (n);
                for (int i=0; i<n; i++) {
                    double x = ulaz.DoubleS ();
                    izlaz.writeDouble (x);
                }
            }
        } catch (FileNotFoundException g) {
            System.out.println ("*** Greska pri otvaranju datoteke!");
        } catch (IOException g) {
            System.out.println ("*** Ulagno/izlazna greska!");
        }
    }
}
```

```
// SekPrik.java - Prikazivanje sadržaja sekvencijalne binarne datoteke.

import java.io.*;

public class SekPrik {
    public static void main (String[] varg) {
        try {
            DataInputStream ulaz = new DataInputStream (
                new FileInputStream (varg[0]));
            while (true) {
                int n = ulaz.readInt (); System.out.print (n);
                for (int i=0; i<n; i++)
                    { double x = ulaz.readDouble (); System.out.print (" " + x); }
                System.out.println ();
            }
        }
    }
}
```

```

    } catch (FileNotFoundException g) {
        System.out.println ("**** Greska pri otvaranju datoteke!");
    } catch (EOFException g) {
    } catch (IOException g) {
        System.out.println ("**** Ulazno/izlazna greska!");
    }
}
}

```

// SekRadi.java - Obrada sekvencijalne binarne datoteke.

```

import java.io.*;

public class SekRadi {
    public static void main (String[] varg) {
        try {
            DataInputStream ulaz = new DataInputStream (
                new FileInputStream (varg[0]));
            DataOutputStream poz = new DataOutputStream (
                new FileOutputStream (varg[1]));
            DataOutputStream neg = new DataOutputStream (
                new FileOutputStream (varg[2]));
            while (true) {
                int n = ulaz.readInt ();
                double[] x = new double [n];
                double s = 0;
                for (int i=0; i<n; s+=(x[i++]=ulaz.readDouble()));
                if (s != 0) {
                    (s>0 ? poz : neg).writeInt (n);
                    for (int i=0; i<n; i++)
                        (s>0 ? poz : neg).writeDouble (x[i]);
                }
            }
        } catch (FileNotFoundException g) {
            System.out.println ("**** Greska pri otvaranju datoteke!");
        } catch (EOFException g) {
        } catch (IOException g) {
            System.out.println ("**** Ulazno/izlazna greska!");
        }
    }
}

```

```

$ java SekPravi SekDat.pod SekDat.dat
$ java SekPrik SekDat.dat
3 1.0 2.0 3.0
4 -1.0 -2.0 -3.0 -4.0
5 0.0 0.0 0.0 0.0 0.0
10 1.0 -1.0 2.0 -2.0 3.0 -3.0 4.0 -4.0 5.0 -5.0
2 123.0 -456.0
2 -123.0 456.0
$ java SekRadi SekDat.dat SekDat.poz SekDat.neg
$ java SekPrik SekDat.poz
3 1.0 2.0 3.0
2 -123.0 456.0
$ java SekPrik SekDat.neg
4 -1.0 -2.0 -3.0 -4.0
2 123.0 -456.0

```

SekDat.pod

3 1 2 3
4 -1 -2 -3 -4
5 0 0 0 0 0
10 1 -1 2 -2 3 -3 4 -4 5 -5
2 123 -456
2 -123 456

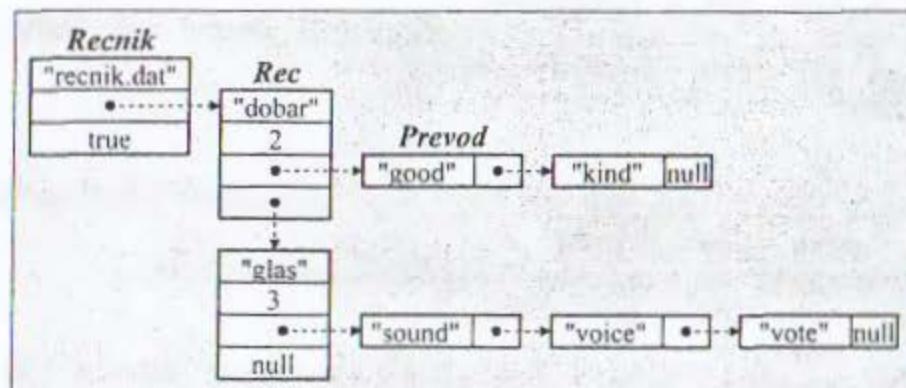
#### Zadatak 9.4 Klasa rečnika, snimanje objekata u datoteku

Napisati na jeziku Java klasu za rečnik koji za svaku reč može da sadrži više prevoda. Predviđeti:

- čitanje rečnika iz zadate datoteke (ako datoteka ne postoji, stvara se prazan rečnik),
- snimanje rečnika u datoteku s imenom koje mu je dodeljeno prilikom stvaranja,
- dodavanje prevoda za datu reč (reč već može da postoji u rečniku),
- brisanje svih prevoda i samo odabranog prevoda date reči,
- dohvatanje svih prevoda date reči, i
- sastavljanje tekstualnog oblika sadržaja rečnika, po jedan red za svaku reč.

Napisati na jeziku Java interaktivni program (s menijem) za rukovanje jednim rečnikom.

**Rešenje:**



```

// Recnik.java - Klasa rečnika.

import java.io.*;

public class Recnik implements Serializable {

    private static final long serialVersionUID = 1L; // Broj verzije klase.
    private String imeDat; // Ime datoteke za trajno uskladištanje.
    private Rec prva; // Prva reč u rečniku.
    private boolean menjano; // Da li je rečnik u memoriji menjan?

    private Recnik (String ime) { imeDat = ime; } // Nov prazan rečnik.

    public static Recnik otvori (String imeDat) { // Otvaranje rečnika.
        try {
            ObjectInputStream dat = new ObjectInputStream (
                new FileInputStream (imeDat));
            Recnik recnik = (Recnik)dat.readObject ();
            dat.close ();
            return recnik;
        } catch (FileNotFoundException g) { // - datoteka ne postoji,
            return new Recnik (imeDat); // stvara se prazan rečnik.
        } catch (IOException g) {
            System.out.println (g); System.exit (1); return null;
        } catch (ClassNotFoundException g) {
            System.out.println (g); System.exit (1); return null;
        }
    }
}
  
```

```

public Recnik snimi () { // Snimanje rečnika u datoteku.
    try {
        if (menjano) {
            ObjectOutputStream dat = new ObjectOutputStream (
                new FileOutputStream (imeDat));
            dat.writeObject (this); dat.close (); menjano = false;
        }
    } catch (IOException g) {
        System.out.println (g); System.exit (1);
    }
    return this;
}

public Recnik dodaj (String rec, String prevod) { // Dodavanje prevoda.
    Rec tek = prva, pret = null;
    while (tek!=null && rec.compareToIgnoreCase(tek.rec)>0)
        { pret = tek; tek = tek.sled; }
    if (tek!=null && rec.equalsIgnoreCase(tek.rec)) tek.dodaj (prevod);
    else new Rec (rec, prevod, pret);
    return this;
}

public Recnik brisi (String rec) { // Brisanje reči sa svim prevodima.
    Rec tek = prva, pret = null;
    while (tek!=null && !rec.equalsIgnoreCase(tek.rec))
        { pret = tek; tek = tek.sled; }
    if (tek != null) {
        if (pret == null) prva = tek.sled; else pret.sled = tek.sled;
        menjano = true;
    }
    return this;
}

public Recnik brisi (String rec, String prevod) { // Brisanje jednog
    Rec tek = prva, pret = null; // prevoda.
    while (tek!=null && !rec.equalsIgnoreCase(tek.rec))
        { pret = tek; tek = tek.sled; }
    if (tek != null) tek.brisi (prevod);
    return this;
}

public String[] prevodi (String rec) { // Dohvatanje svih prevoda reči.
    Rec tek = prva;
    while (tek!=null && !rec.equalsIgnoreCase(tek.rec)) tek = tek.sled;
    if (tek != null) return tek.prevodi (); else return null;
}

public String toString () { // Tekstualni oblik rečnika.
    StringBuffer s = new StringBuffer ();
    for (Rec tek=prva; tek!=null; tek=tek.sled)
        s.append (tek).append ('\n');
    return s.toString ();
}

private class Rec implements Serializable { // KLASA ZA JEDNU REČ.

    String rec; // Tekst reči.
    int brPrev; // Broj prevoda.
    Prevod prvi; // Prvi prevod.
    Rec sled; // Sledeća reč u rečniku.
}

```

```

Rec (String rec, String prevod, Rec iza) { // Stvaranje nove reči.
    this.rec = rec; new Prevod (prevod, prvi);
    if (iza == null) { sled = prva; prva = this; }
    else { sled = iza.sled; iza.sled = this; }
}

Rec dodaj (String prevod) { // Dodavanje novog prevoda.
    Prevod tek = prvi, pret = null;
    while (tek!= null && prevod.compareToIgnoreCase(tek.prevod)>0)
        { pret = tek; tek = tek.sled; }
    if (tek==null || !prevod.equalsIgnoreCase(tek.prevod))
        new Prevod (prevod, pret);
    return this;
}

Rec brisi (String prevod) { // Brisanje datog prevoda.
    Prevod tek = prvi, pret = null;
    while (tek!= null && prevod.compareToIgnoreCase(tek.prevod)>0)
        { pret = tek; tek = tek.sled; }
    if (tek!=null && prevod.equalsIgnoreCase(tek.prevod)) {
        if (pret == null) prvi = tek.sled; else pret.sled = tek.sled;
        brPrev--; menjano = true;
    }
    return this;
}

String[] prevodi () { // Dohvatanje svih prevoda.
    String[] rez = new String [brPrev];
    int i = 0;
    for (Prevod tek=prvi; tek!=null; tek=tek.sled) rez[i++] = tek.prevod;
    return rez;
}

public String toString () { // Tekstualni oblik reči s prevodima.
    StringBuffer s = new StringBuffer ();
    s.append (rec).append (':');
    for (Prevod tek=prvi; tek!=null; tek=tek.sled) {
        s.append (' ').append (tek.prevod)
            .append (tek.sled!=null ? ',' : '.');
    }
    return s.toString ();
}

private class Prevod implements Serializable { // KLASA ZA JEDAN PREVOD.

    String prevod; // Tekst prevoda.
    Prevod sled; // Sledeći prevod.

    Prevod (String p, Prevod iza) { // Stvaranje novog prevoda.
        prevod = p;
        if (iza == null) { sled = prvi; prvi = this; }
        else { sled = iza.sled; iza.sled = this; }
        brPrev++; menjano = true;
    }
} // class Prevod
} // class Rec
} // class Recnik

```

```

public Recnik snimi () { // Snimanje rečnika u datoteku.
    try {
        if (menjano) {
            ObjectOutputStream dat = new ObjectOutputStream (
                new FileOutputStream (imeDat));
            dat.writeObject (this); dat.close (); menjano = false;
        }
    } catch (IOException g) {
        System.out.println (g); System.exit (1);
    }
    return this;
}

public Recnik dodaj (String rec, String prevod) { // Dodavanje prevoda.
    Rec tek = prva, pret = null;
    while (tek!=null && rec.compareToIgnoreCase(tek.rec)>0)
        { pret = tek; tek = tek.sled; }
    if (tek!=null && rec.equalsIgnoreCase(tek.rec)) tek.dodaj (prevod);
    else new Rec (rec, prevod, pret);
    return this;
}

public Recnik brisi (String rec) { // Brisanje reči sa svim prevodima.
    Rec tek = prva, pret = null;
    while (tek!=null && !rec.equalsIgnoreCase(tek.rec))
        { pret = tek; tek = tek.sled; }
    if (tek != null) {
        if (pret == null) prva = tek.sled; else pret.sled = tek.sled;
        menjano = true;
    }
    return this;
}

public Recnik brisi (String rec, String prevod) { // Brisanje jednog
    Rec tek = prva, pret = null; // prevoda.
    while (tek!=null && !rec.equalsIgnoreCase(tek.rec))
        { pret = tek; tek = tek.sled; }
    if (tek != null) tek.brisi (prevod);
    return this;
}

public String[] prevodi (String rec) { // Dohvatanje svih prevoda reči.
    Rec tek = prva;
    while (tek!=null && !rec.equalsIgnoreCase(tek.rec)) tek = tek.sled;
    if (tek != null) return tek.prevodi (); else return null;
}

public String toString () { // Tekstualni oblik rečnika.
    StringBuffer s = new StringBuffer ();
    for (Rec tek=prva; tek!=null; tek=tek.sled)
        s.append (tek).append ('\n');
    return s.toString ();
}

private class Rec implements Serializable { // KLASA ZA JEDNU REČ.

    String rec; // Tekst reči.
    int brPrev; // Broj prevoda.
    Prevod prvi; // Prvi prevod.
    Rec sled; // Sledeća reč u rečniku.
}

```

```

Rec (String rec, String prevod, Rec iza) { // Stvaranje nove reči.
    this.rec = rec; new Prevod (prevod, prvi);
    if (iza == null) { sled = prva; prva = this; }
    else { sled = iza.sled; iza.sled = this; }
}

Rec dodaj (String prevod) { // Dodavanje novog prevoda.
    Prevod tek = prvi, pret = null;
    while (tek!= null && prevod.compareToIgnoreCase(tek.prevod)>0)
        { pret = tek; tek = tek.sled; }
    if (tek==null || !prevod.equalsIgnoreCase(tek.prevod))
        new Prevod (prevod, pret);
    return this;
}

Rec brisi (String prevod) { // Brisanje datog prevoda.
    Prevod tek = prvi, pret = null;
    while (tek!= null && prevod.compareToIgnoreCase(tek.prevod)>0)
        { pret = tek; tek = tek.sled; }
    if (tek!=null && prevod.equalsIgnoreCase(tek.prevod)) {
        if (pret == null) prvi = tek.sled; else pret.sled = tek.sled;
        brPrev--; menjano = true;
    }
    return this;
}

String[] prevodi () { // Dohvatanje svih prevoda.
    String[] rez = new String [brPrev];
    int i = 0;
    for (Prevod tek=prvi; tek!=null; tek=tek.sled) rez[i++] = tek.prevod;
    return rez;
}

public String toString () { // Tekstualni oblik reči s prevodima.
    StringBuffer s = new StringBuffer ();
    s.append (rec).append (':');
    for (Prevod tek=prvi; tek!=null; tek=tek.sled) {
        s.append (' ').append (tek.prevod)
        .append (tek.sled!=null ? ',' : '.');
    }
    return s.toString ();
}

private class Prevod implements Serializable { // KLASA ZA JEDAN PREVOD.

    String prevod; // Tekst prevoda.
    Prevod sled; // Sledeci prevod.

    Prevod (String p, Prevod iza) { // Stvaranje novog prevoda.
        prevod = p;
        if (iza == null) { sled = prvi; prvi = this; }
        else { sled = iza.sled; iza.sled = this; }
        brPrev++; menjano = true;
    }
} // class Prevod
} // class Rec
} // class Recnik

```

```

// RecnikT.java - Ispitivanje klase rečnika.

import usluge.Citaj;

public class RecnikT {
    public static void main (String[] varg) {
        Recnik r = Recnik.otvori ("RecnikT.dat");
        radi: while (true) {
            System.out.print ("\n1. Traženje prevoda\n" +
                "2. Dodavanje prevoda\n" +
                "3. Brisanje prevoda\n" +
                "4. Brisanje svih prevoda\n" +
                "5. Prikazivanje rečnika\n" +
                "0. Završetak rada\n\n");
            String rec = "", prevod = ""; String[] prevodi = null;
            switch (Citaj.Char ()) {
                case '1': // Traženje prevoda:
                    while (true) {
                        System.out.print ("Rec?      "); rec = Citaj.String ();
                        if (rec.equals ("?")) break;
                        System.out.print ("Prevodi:  ");
                        if ((prevodi = r.prevodi (rec)) != null)
                            for (int i=0; i<prevodi.length;
                                System.out.print (prevodi[i++]+ " "));
                        System.out.println ();
                    } break;
                case '2': // Dodavanje prevoda:
                    while (true) {
                        System.out.print ("Rec?      "); rec = Citaj.String ();
                        if (rec.equals ("?")) break;
                        while (true) {
                            System.out.print ("Prevod?   "); prevod = Citaj.String ();
                            if (prevod.equals ("?")) break;
                            r.dodaj (rec, prevod);
                        }
                    } break;
                case '3': // Brisanje prevoda:
                    while (true) {
                        System.out.print ("Rec?      "); rec = Citaj.String ();
                        if (rec.equals ("?")) break;
                        while (true) {
                            System.out.print ("Prevod?   "); prevod = Citaj.String ();
                            if (prevod.equals ("?")) break;
                            r.brisi (rec, prevod);
                        }
                    } break;
                case '4': // Brisanje svih prevoda:
                    while (true) {
                        System.out.print ("Rec?      "); rec = Citaj.String ();
                        if (rec.equals ("?")) break;
                        System.out.print ("Prevodi:  ");
                        if ((prevodi = r.prevodi (rec)) != null)
                            for (int i = 0; i<prevodi.length;
                                System.out.print (prevodi[i++]+ " "));
                        System.out.println ();
                        System.out.print ("Brisati?  ");
                        if (Citaj.String ().equalsIgnoreCase ("da")) r.brisi (rec);
                    } break;
            }
        }
    }
}

```



Izdavač

AKADEMSKA MISAO  
ELEKTROTEHNIČKI FAKULTET

Bul. kralja Aleksandra 73, Beograd  
tel./fax: (+381 11) 3218 354

[knjizara@akademiska-misao.co.yu](mailto:knjizara@akademiska-misao.co.yu)

[www.akademiska-misao.co.yu](http://www.akademiska-misao.co.yu)

---

CIP - Каталогизација у публикацији  
Народна библиотека Србије, Београд



004.438JAVA(075.8)(076)

КРАУС, Ласло Л.

Rešeni zadaci iz programskog jezika Java  
/Laslo Kraus. - [2. preradeno izdanje]. -  
Beograd : Akademска misao, 2007 (Beograd  
: Planeta print). - 265 str. ; ilustr. ; 24 cm

Tiraž 500. - Preporučena literatura str. 6

ISBN 978-86-7466-281-6

---

a) Програмски језик "JAVA" - Задаци  
COBISS.SR-ID 138080780