

Dejan Živković

# OSNOVE JAVA PROGRAMIRANJA

Zbirka pitanja i zadataka sa rešenjima



**UNIVERZITET SINGIDUNUM**

Dejan Živković

# **OSNOVE JAVA PROGRAMIRANJA**

Zbirka pitanja i zadataka sa rešenjima

Drugo izdanje

Beograd, 2010.

# **OSNOVE JAVA PROGRAMIRANJA**

## Zbirka pitanja i zadataka sa rešenjima

*Autor:*

Prof. dr Dejan Živković

*Recenzent:*

Prof. dr Dragan Cvetković

*Izdavač:*

UNIVERZITET SINGIDUNUM

Beograd, Danijelova 32

[www.singidunum.ac.rs](http://www.singidunum.ac.rs)

*Za izdavača:*

Prof. dr Milovan Stanišić

*Tehnička obrada:*

Dejan Živković

*Dizajn korica:*

Aleksandar Mihajlović

*Godina izdanja:*

2010.

*Tiraž:*

70 primeraka

*Štampa:*

Mladost Grup

Loznica

ISBN: 978-86-7912-238-4

# ***Sadržaj***

<b>Spisak programa</b>	<b>iv</b>
<b>Predgovor</b>	<b>vii</b>
<b>I Pitanja i zadaci</b>	<b>1</b>
<b>1 Uvod u Java programiranje</b>	<b>3</b>
Pitanja . . . . .	3
<b>2 Uvod u programski jezik Java</b>	<b>7</b>
Pitanja . . . . .	7
<b>3 Osnovni elementi jezika Java</b>	<b>11</b>
Pitanja . . . . .	11
Programski zadaci . . . . .	21
<b>4 Upravljačke naredbe</b>	<b>23</b>
Pitanja . . . . .	23
Programski zadaci . . . . .	36
<b>5 Metodi</b>	<b>39</b>
Pitanja . . . . .	39
Programski zadaci . . . . .	50
<b>6 Klase i objekti</b>	<b>53</b>
Pitanja . . . . .	53
Programski zadaci . . . . .	65
<b>7 Osnovne strukture podataka</b>	<b>67</b>
Pitanja . . . . .	67
Programski zadaci . . . . .	81

<b>8 Nasleđivanje klasa</b>	<b>85</b>
Pitanja . . . . .	85
Programski zadaci . . . . .	98
<b>9 Posebne klase i interfejsi</b>	<b>99</b>
Pitanja . . . . .	99
Programski zadaci . . . . .	106
<b>10 Grafičko programiranje</b>	<b>109</b>
Pitanja . . . . .	109
Programski zadaci . . . . .	125
 <b>II Rešenja</b>	 <b>127</b>
<b>1 Uvod u Java programiranje</b>	<b>129</b>
Odgovori na pitanja . . . . .	129
<b>2 Uvod u programski jezik Java</b>	<b>133</b>
Odgovori na pitanja . . . . .	133
<b>3 Osnovni elementi jezika Java</b>	<b>135</b>
Odgovori na pitanja . . . . .	135
Rešenja zadataka . . . . .	143
<b>4 Upravljačke naredbe</b>	<b>149</b>
Odgovori na pitanja . . . . .	149
Rešenja zadataka . . . . .	155
<b>5 Metodi</b>	<b>163</b>
Odgovori na pitanja . . . . .	163
Rešenja zadataka . . . . .	168
<b>6 Klase i objekti</b>	<b>173</b>
Odgovori na pitanja . . . . .	173
Rešenja zadataka . . . . .	179
<b>7 Osnovne strukture podataka</b>	<b>191</b>
Odgovori na pitanja . . . . .	191
Rešenja zadataka . . . . .	198

---

<b>8 Nasleđivanje klasa</b>	<b>213</b>
Odgovori na pitanja . . . . .	213
Rešenja zadataka . . . . .	219
<b>9 Posebne klase i interfejsi</b>	<b>231</b>
Odgovori na pitanja . . . . .	231
Rešenja zadataka . . . . .	235
<b>10 Grafičko programiranje</b>	<b>265</b>
Odgovori na pitanja . . . . .	265
Rešenja zadataka . . . . .	274



# *Spisak programa*

3.1	ImePrezime.java . . . . .	143
3.2	Inicijali.java . . . . .	143
3.3	A4.java . . . . .	143
3.4	CelFar.java . . . . .	144
3.5	Kamata.java . . . . .	144
3.6	PravougliTrougao.java . . . . .	145
3.7	Tren.java . . . . .	146
4.1	Uskrs1.java . . . . .	155
4.2	Uskrs2.java . . . . .	155
4.3	PismoGlava1.java . . . . .	156
4.4	ZmajskeOči.java . . . . .	157
4.5	Niz3n1.java . . . . .	157
4.6	NZD1.java . . . . .	158
4.7	NBD.java . . . . .	159
4.8	Romb.java . . . . .	159
4.9	SlovaCifre.java . . . . .	160
4.10	ListaReči.java . . . . .	161
5.1	NZD2.java . . . . .	168
5.2	Niz3n1m.java . . . . .	168
5.3	KapString.java . . . . .	169
5.4	HeksSistem.java . . . . .	170
5.5	DveKocke1.java . . . . .	171
6.1	MesečniKalendar.java . . . . .	179
6.2	TKK.java . . . . .	180
6.3	KompleksanBroj.java . . . . .	183
6.4	RimskiBroj.java . . . . .	185
6.5	DveKocke2.java . . . . .	187
6.6	PismoGlava.java . . . . .	189
7.1	Sito.java . . . . .	198
7.2	IgraŽivota.java . . . . .	199

---

7.3	KešMemorija.java . . . . .	201
7.4	ČesteReči.java . . . . .	203
7.5	DomZdravlja.java . . . . .	206
8.1	PovezanaLista.java . . . . .	219
8.2	PoliLinija.java . . . . .	221
8.3	PovezanaLista1.java . . . . .	225
9.1	Karta.java . . . . .	235
9.2	Konvertor.java . . . . .	236
9.3	KošarkaškaUtakmica.java . . . . .	238
9.4	GO.java . . . . .	244
9.5	GO1.java . . . . .	247
9.6	KućnaZabava.java . . . . .	250
9.7	PovezanaLista2.java . . . . .	255
9.8	IgraŽivota1.java . . . . .	257
9.9	Sortiranje.java . . . . .	260
10.1	Kamion.java . . . . .	274
10.2	Šah.java . . . . .	275
10.3	Ribice.java . . . . .	276
10.4	Kalkulator.java . . . . .	280
10.5	DigitalniSat.java . . . . .	283
10.6	ProricanjeSudbine.java . . . . .	284
10.7	PomeranjeKvadrata.java . . . . .	286
10.8	IgraKliznaTabla1.java . . . . .	290
10.9	IgraKliznaTabla2.java . . . . .	293
10.10	Editor.java . . . . .	297
10.11	Loptice.java . . . . .	301
10.12	AnalogniSat.java . . . . .	305



# Predgovor

Ova zbirka pitanja i zadataka može poslužiti čitaocima za proveru i utvrđivanje znanja iz Java programiranja. Zbirka je prilagođena nivou knjige *Osnove Java programiranja* i može poslužiti kao njen praktikum, jer potpuno prati logičku nît izlaganja materijala u toj knjizi.

Kod svakog pitanja za proveru znanja su dati potencijalni tačni odgovori. Pri tome,

- pitanje sa više ponuđenih odgovora označenih simbolom  $\circ$  ima samo jedan tačan odgovor;
- pitanje sa više ponuđenih odgovora označenih simbolom  $\square$  ima više tačnih odgovora i treba ih sve obeležiti.

Neka pitanja su data u obliku nepotpune rečenice na mesto označenom simbolom           . U takvom slučaju za to mesto treba izabrati i označiti ponuđeni odgovor kako bi s njim rečenica bila tačna.

Kod sastavljanja programskih zadataka je učinjen napor kako bi njihovo rešenje zahtevalo samo one elemente Jave koji su predstavljeni do odgovarajućeg poglavlja u knjizi *Osnove Java programiranja*. Ali, da bi zadaci ipak bili interesantniji i praktičniji, njihova rešenja se ponekad oslanjaju na klase Java platforme o kojima nije bilo reči u toj knjizi. Ovo se prvenstveno odnosi na grafičke programe u poslednjem poglavlju i čitaoci se upućuju na zvaničnu dokumentaciju Jave iz koje mogu saznati više detalja.

Kao i uvek, bio bih veoma zahvalan čitaocima na njihovom mišljenju o zbirci. Sve primedbe i pronađene greške se mogu poslati elektronskom poštom na adresu [dzivkovic@singidunum.ac.rs](mailto:dzivkovic@singidunum.ac.rs).

ĐEJAN ŽIVKOVIĆ  
Beograd, Srbija  
jun 2009.



Deo I

# Pitanja i zadaci



## *Glava*

# 1

## *Uvod u Java programiranje*

### Pitanja

1. Kako se nazivaju fizički delovi od kojih se sastoji računar (procesor, memorija, disk, tastatura, monitor i tako dalje)?
  - Softver
  - Hardver
  - Operativni sistem
  - Windows
2. Kako se nazivaju svi programi u računaru čijim izvršavanjem računar obavlja korisne poslove za ljude?
  - Softver
  - Hardver
  - Operativni sistem
  - Windows
3. Šta je „mozak“ računara?
  - Hardver
  - Procesor (CPU)
  - Memorija

- Disk
4. Od čega se sastoji računarski program (pojednostavljeno ali najtačnije rečeno)?
- naredbi
  - podataka
  - naredbi i podataka
  - teksta
5. Koliko bitova ima jedan bajt?
- 4
  - 8
  - 16
  - 32
6. Računar može izvršavati programe samo na \_\_\_\_\_.
- mašinskom jeziku
  - engleskom jeziku
  - prirodnom jeziku
  - jeziku visokog nivoa
7. Šta prevodi programe napisane na programskom jeziku visokog nivoa u programe na mašinskom jeziku?
- Operativni sistemi
  - Procesori
  - Ljudi
  - Prevodioci (kompajleri)
8. Kako se naziva program preveden na mašinski jezik Java virtuelne mašine?
- Java bajtkod
  - Java objektni kod

- Java aplet
- Java aplikacija

9. Šta mora biti ime datoteke u kojoj se nalazi ovaj Java program kojeg čini jedna klasa Test?

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

- Test.txt
- Test.class
- Test.java
- Main.java
- Main.txt

10. Ime datoteke u kojoj se nalazi preveden Java program (Java bajtkod) završava se sufiksom \_\_\_\_\_.

- .java
- .obj
- .class
- .exe

11. DOS komanda za prevođenje Java programa koji se nalazi u datoteci Test.java je \_\_\_\_\_.

- javac Test.java
- compile Test.java
- prevedi Test.java
- javap Test.java

12. DOS komanda za izvršavanje (interpretiranje) prevedenog Java programa u datoteci Test.class je \_\_\_\_\_.

- javac Test
  - izvrsti Test
  - java Test
  - java Test.class
13. Grafička okruženja za razvoj Java programa su \_\_\_\_\_.
- NetBeans
  - DOS
  - Windows
  - DrJava
  - Linux



*Glava*

# 2

## *Uvod u programski jezik Java*

### Pitanja

1. Klasa je opis \_\_\_\_\_ sa zajedničkim svojstvima.
  - promenljivih
  - procedura
  - programa
  - objekata
2. Objekat je primerak (instanca) neke \_\_\_\_\_.
  - promenljive
  - procedure
  - klase
  - mašine
3. Dve glavne karakteristike svakog softverskog objekta su \_\_\_\_\_.
  - obeležja (atributi) i mogućnosti (ponašanje)
  - sastavni delovi i izgled
  - broj i oblik
  - identifikacija i način upotrebe

4. Nasleđivanje klasa je način na koji se postojeća klasa može \_\_\_\_\_.  
 zameniti  
 proširiti  
 popraviti  
 ukloniti
5. Raspoložive klase Java platforme su organizovane po \_\_\_\_\_.  
 nazivima  
 paketima  
 modulima  
 folderima
6. Raspoložive klase Java platforme se dodaju programu navođenjem deklaracije \_\_\_\_\_.  
 import  
 export  
 module  
 package
7. Koje su od ovih rečenica o paketima tačne?  
 Po konvenciji, u Javi se imena paketa pišu svim malim slovima.  
 Deklaracija package nije obavezna.  
 Deklaracija import nije obavezna.  
 Klase u paketu java.lang se automatski dodaju u program.
8. U kojem od ovih slučajeva *nije* ispravno napisan komentar u Javi?  
 `/** tekst komentara */`  
 `// tekst komentara`  
 `-- tekst komentara`  
 `/* tekst komentara */`  
 `** tekst komentara **`

9. Dobar stil programiranja je važan, jer \_\_\_\_\_.

- program se neće prevesti zato što je napisan lošim stilom
- program će se brže izvršavati zato što je napisan dobrim stilom
- program će biti čitljiviji zato što je napisan dobrim stilom
- program će imati manji broj grešaka zato što je napisan dobrim stilom
- program će se lakše modifikovati zato što je napisan dobrim stilom

10. Svaka naredba u Javi se završava \_\_\_\_\_.

- tačkom (.)
- tačkom-zapetom (;)
- zapetom (,)
- zvezdicom (\*)



*Glava*

# 3

## *Osnovni elementi jezika Java*

### Pitanja

1. Koje su od ovih reči službene reči u Javi?  
 public  
 static  
 void  
 class  
 tačno
2. U kojem su od ovih slučajeva ispravno napisana imena (identifikatori) u Javi?  
 9x  
 ekran  
 brojStudenata  
 znak+ili-
3. U kojem su od ovih slučajeva ispravno napisana imena (identifikatori) u Javi?  
 3praseta  
 tri praseta

- prečnik
  - bzvz
4. Svi mogući tipovi podataka u Javi se dele na \_\_\_\_\_.
- specifične i generalne
  - celobrojne i realne
  - numeričke i tekstualne
  - primitivne i klasne
5. Koji od ovih celobrojnih tipova podataka zahteva najviše memorije za predstavljanje celih brojeva?
- long
  - int
  - short
  - byte
6. Koji od ovih tipova podataka u Javi služe za predstavljanje realnih brojeva?
- float
  - int
  - long
  - double
  - boolean
7. Jedan znak u Javi tipa char zauzima \_\_\_\_\_ u memoriji.
- jedan bajt
  - dva bajta
  - tri bajta
  - četiri bajta
8. Moguće logičke vrednosti tipa boolean u Javi su \_\_\_\_\_.
- tačno

- true
  - false
  - 0
  - 1
  - netačno
9. U kojem su od ovih slučajeva ispravno napisana imena promenljivih prema *konvenciji* u Javi za davanje imena promenljivim?
- kredit
  - Kredit
  - KREDIT
  - kamatnaStopa
  - KamatnaStopa
  - kamatna\_stopa
10. U kojem su od ovih slučajeva ispravno napisane naredbe za deklarisanje (definisanje) promenljivih u Javi?
- int dužina; int širina;
  - int dužina, širina;
  - int dužina; širina;
  - int dužina, int širina;
11. U kojem su od ovih slučajeva ispravno napisane naredbe za prikazivanje teksta Java je kul! na ekranu?
- System.out.println('Java je kul!');
  - System.println("Java je kul!");
  - System.out.writeln("Java je kul!");
  - System.out.println("Java je kul!");
  - System.out.print("Java je kul!");
  - System.out.printf("Java je kul!");
12. Kojom se od ovih naredbi ispravno dodeljuje vrednost 17 promenljivoj x?

- 17 = x;
  - x = 17;
  - x := 17;
  - x == 17;
13. Kojom se od ovih naredbi ispravno deklariše promenljiva x tipa int sa početnom vrednošću 17?
- int x = '17';
  - int x == 17;
  - int x = 17;
  - int x = 17.0;
14. Šta je rezultat izraza  $45 / 4$  u Javi?
- 10
  - 11
  - 11.25
  - 12
15. Koji od ovih izraza kao rezultat daju 0.5 u Javi?
- $1 / 2$
  - $1.0 / 2$
  - $1.0 / 2.0$
  - $(\text{double}) (1 / 2)$
  - $(\text{double}) 1 / 2$
  - $1 / 2.0$
16. Koji od ovih izraza kao rezultat daje 1 u Javi?
- $2 \% 1$
  - $15 \% 4$
  - $25 \% 5$
  - $37 \% 6$

17. Ako su  $a$  i  $b$  celobrojne promenljive tipa int, koji od ovih izraza u Javi kao rezultat daju tačan rezultat (realni broj) za matematički izraz  $\frac{a}{b^2}$ ? (Na primer, ako su  $a = 5$  i  $b = 2$ , onda je  $\frac{a}{b^2} = \frac{5}{4} = 1.25$  tačan rezultat.)

- $a / b * b$
- $a / (b * b)$
- $1.0 * a / b * b$
- $1.0 * a / (b * b)$
- $(double) a / (b * b)$

18. Koje od ovih naredbi dodele *nisu* ispravne?

- float  $f = -34;$
- int  $t = 23;$
- short  $s = 10;$
- int  $t = (\text{int})\text{false};$
- int  $t = 4.5;$

19. Koji se tekst dobija na ekranu izvršavanjem ovog programskog fragmenta?

```
double x = 5.5;  
int y = (int)x;  
System.out.println("x je " + x + " i y je " + y);
```

- x je 5 i y je 6
- x je 6.0 i y je 6.0
- x je 6 i y je 6
- x je 5.5 i y je 5
- x je 5.5 i y je 5.0

20. Koje od ovih naredbi ispravno dodaju 1 celobrojnoj promenljivi i tipa int?

- $i = i + (2 - 1);$
- $i = i + 1;$
- $i += 1;$

- i = 1 + i;
- i++;

21. Analizirajte sledeći program:

```
public class Test {  
    public static void main(String[] args) {  
        int mesec = 09;  
        System.out.println("Mesec je " + mesec);  
    }  
}
```

- Program prikazuje Mesec je 09 na ekranu.
  - Program prikazuje Mesec je 9 na ekranu.
  - Program prikazuje Mesec je 9.0 na ekranu.
  - Program ima grešku, jer 09 nije ispravno napisana oktalna vrednost.
22. U Java programu se izvršavanjem poziva metoda \_\_\_\_\_ odmah prekida izvršavanje tog programa.
- System.terminate(0)
  - System.halt(0)
  - System.exit(0)
  - System.stop(0)
23. Metod \_\_\_\_\_ u Javi izračunava broj x na stepen y.
- Math.power(x, y)
  - Math.exp(x, y)
  - Math.pow(x, y)
  - Math.pow(y, x)
24. Koja je od ovih naredbi deklarisanja promenljive ispravna u Javi?
- char c = 'A';
  - char c = '23';
  - char c = "A";
  - char c = "23";

25. Koje su od ovih naredbi deklarisanja promenljive ispravne u Javi?

- `string s = 'A';`
- `String s = '23';`
- `String s = "A";`
- `String s = "23";`

26. Šta se dodeljuje promenljivoj c kao rezultat izvršavanja ovog program-skog fragmenta?

```
String s = "Java";
char c = s.charAt(4);
```

- 'a'
- 'v'
- Ništa, jer dolazi do prekida izvršavanja programa usled greške.

27. Ako su s1 i s2 promenljive tipa String, koji su slučajevi ovih naredbi ili izraza pogrešni?

- `String s = "neki string";`
- `String s3 = s1 + s2;`
- `s1 >= s2`
- `int i = s1.length;`
- `s1.charAt(0) = '?';`

28. Ako su s1 i s2 promenljive tipa String, koji su slučajevi ovih naredbi ili izraza pogrešni?

- `String s3 = s1 - s2;`
- `s1 == s2`
- `boolean b = s1.compareTo(s2);`
- `char c = s1[0];`
- `char c = s1.charAt(s1.length());`
- `char c = s1.charAt(s1.length() - 1);`

29. Ako su s1 i s2 promenljive tipa String, šta je rezultat ovog izraza?

- `s1.equals(s2) == s2.equals(s1)`
- 0
  - 1
  - true
  - false
30. Šta je rezultat izraza "java" + "program" u Javi?
- javaprogram
  - Java program
  - Java\_program
  - Javaprogram
  - greška
31. Kojim se od ovih naredbi ispravno pretvara string s u celobrojnu vrednost promenljive i tipa int?
- `i = Integer.parseInt(s);`
  - `i = (new Integer(s)).intValue();`
  - `i = Integer.valueOf(s).intValue();`
  - `i = Integer.valueOf(s);`
  - `i = (int)(Double.parseDouble(s));`
32. Kojim se od ovih naredbi ispravno pretvara string s u realnu vrednost promenljive d tipa double?
- `d = Double.parseDouble(s);`
  - `d = (new Double(s)).doubleValue();`
  - `d = Double.valueOf(s).doubleValue();`
  - `d = (double)(Integer.parseInt(s));`
33. Kojim se od ovih naredbi ispravno pretvara realna vrednost promenljive d tipa double u string s?
- `s = d;`
  - `s = d.toString();`

- s = (new Double(d)).toString();
- s = (Double.valueOf(d)).toString();

34. Relacijski operator „manje ili jednako” u Javi je \_\_\_\_\_.

- <
- <=
- >=
- <=
- <<
- !=

35. Relacijski operator „jednako” u Javi je \_\_\_\_\_.

- <>
- !=
- ==
- =

36. Koji su od ovih logičkih izraza ispravno napisani u Javi?

- (true) && (3 => 4)
- !(x > 0) && (x > 0)
- (x > 0) || (x < 0)
- (x != 0) || (x = 0)
- (-10 < x < 0)

37. Koji od ovih logičkih izraza ispravno daje vrednost true ako je broj x između 1 i 100 ili je broj x negativan?

- 1 < x < 100 && x < 0
- ((x < 100) && (x > 1)) || (x < 0)
- ((x < 100) && (x > 1)) && (x < 0)
- (x != 0) || (x = 0)
- (1 > x > 100) || (x < 0)
- (1 < x < 100) || (x < 0)

38. Šta je vrednost promenljive  $x$  posle izračunavanja izraza

$(y > 10) \&& (x++ > 10)$

ako je pre izračunavanja tog izraza bilo  $x = 10$  i  $y = 10$ ?

- 9
- 10
- 11
- 12

39. Šta je vrednost promenljive  $x$  posle izračunavanja izraza

$(y > 10) \mid\mid (x++ > 10)$

ako je pre izračunavanja tog izraza bilo  $x = 10$  i  $y = 10$ ?

- 9
- 10
- 11
- 12

40. Koju vrednost ima promenljiva  $y$  posle izvršavanja ovog programskog fragmenta?

```
x = 0;  
y = (x > 0) ? 10 : -10;
```

- 10
- 0
- 10
- 20
- Programski fragment ima grešku.

## Programski zadaci

1. Napisati program koji u pravougaoniku na ekranu ispisuje vaše ime i prezime na sledeći način:

```
+-----+  
| |  
| Dejan Živković |  
| |  
+-----+
```

2. Napisati program koji na ekranu ispisuje vaše inicijale velikim slovima koja su „nacrtana” zvezdicama (znakom \*) u matrici dimenzije  $11 \times 12$ . Na primer, na ekranu se za inicijale DŽ dobija otprilike sledeća slika:

```
*****  
**   **  
**   **  
**   **  
**   **  
**   **  
**   **  
**   **  
*****  
*****
```

3. Napisati program koji format papira A4 ( $210 \times 297$  mm) prikazuje u inčima.
4. Napisati program koji Celzijusove stepene pretvara u Farenhajtove po formuli  $f = 9c/5 + 32$ .
5. Napisati program koji izračunava iznos kamate na depozit i uvećano stanje depozita nakon jedne godine. Ulazne veličine programa su početni depozit i godišnja kamatna stopa, a izlazne veličine su novčani iznos kamate i uvećani depozit nakon jedne godine.

6. Napisati program koji izračunava hipotenuzu pravouglog trougla ako su date dve njegove katete. Pored toga, program treba da prikaže vreme u sekundama koje je utrošeno za obavljanje tog zadatka.
  
7. Napisati program koji dati vremenski trenutak izražen pomoću broja godina, meseca, dana, sati i minuta „pakuje” u jednu promenljivu tipa int, zatim prikazuje njenu celobrojnu vrednost i na kraju „raspakuje” sadržaj te promenljive prikazujući originalne podatke o datom vremenskom trenutku.



# 4

## *Upravljačke naredbe*

### Pitanja

1. Kojim se od ovih naredbi ispravno prikazuje površina kruga ako je prečnik r pozitivan?
  - if (r != 0) System.out.println(r \* r \* Math.PI);
  - if (r >= 0) System.out.println(r \* r \* Math.PI);
  - if (r > 0) System.out.println(r \* r \* Math.PI);
  - if (r > 0) { System.out.println(r \* r \* Math.PI); }
  - if {r > 0} System.out.println(r \* r \* PI);
  - if (r <= 0) System.out.println(r \* r \* Math.PI);
  - if (r > 0) System.out.println(Math.pow(r, 2) \* Math.PI);
2. Ako je  $x = 1$ ,  $y = -1$  i  $z = 1$ , šta se prikazuje na ekranu izvršavanjem ovog programskog fragmenta? (Savet: najpre ispravno uparite if i else delove.)

```
if (x > 0)
    if (y > 0)
        System.out.println("x > 0 i y > 0");
    else if (z > 0)
        System.out.println("x < 0 i z > 0");
```

- x > 0 i y > 0

- x < 0 i z > 0
- x < 0 i z < 0
- Ništa se ne prikazuje.

3. Analizirajte sledeći programski fragment:

```
boolean tačno = false;
if (tačno = true)
    System.out.println("To je tačno!");
```

- Programski fragment ima grešku i ne može se izvršiti.
  - Programski fragment se normalno izvšava, ali se ništa ne prikazuje na ekranu.
  - Programski fragment se normalno izvšava i prikazuje se To je tačno! na ekranu.
4. Ako je broj celobrojna promenljiva tipa int, analizirajte sledeća dva ekvivalentna programska fragmenta A i B:

Fragment A:

```
boolean paranBroj;
if (broj % 2 == 0)
    paranBroj = true;
else
    paranBroj = false;
```

Fragment B:

```
boolean paranBroj = (broj % 2 == 0);
```

- Fragment A ima grešku.
  - Fragment B ima grešku.
  - Oba fragmenta imaju grešku.
  - Oba fragmenta su ispravna, ali je fragment B bolji.
5. Ako celobrojna promenljiva plata sadrži vrednost 4001, šta će biti prikazano na ekranu posle izvršavanja ovog programskog fragmenta?

```
if (plata > 3000)
    System.out.println("Plata je veća od 3000");
else if (plata > 4000)
    System.out.println("Plata je veća od 4000");
```

- Ništa se neće prikazati.
  - Plata je veća od 3000
  - Plata je veća od 4000
  - Plata je veća od 3000 u jednom redu i Plata je veća od 4000 u sledećem redu.
6. Šta se prikazuje na ekranu kao rezultat izvršavanja ovog programskog fragmenta?
- ```
double faktor = 1.5;
double cena = 0.0;
if ((faktor >= 0.0) && (faktor < 1.0))
    cena = 20 * faktor;
else if ((faktor >= 1.0) && (faktor < 1.5))
    cena = 15 * faktor;
else if (faktor >= 1.5)
    cena = 10 * faktor;
System.out.println("cena = " + cena);
```
- cena = 0.0
  - Ništa se neće prikazati.
  - Programski fragment ima grešku i neće se izvršiti.
  - cena = 15.0
7. Šta se prikazuje na ekranu kao rezultat izvršavanja ovog programa?

```
public class Test {
    public static void main(String[] args) {
        String s1 = "Java";
        String s2 = s1;

        if (s1 == s2)
            System.out.println(
                "s1 i s2 ukazuju na isti string");
        else
            System.out.println(
                "s1 i s2 ukazuju na različite stringove");
    }
}
```

- Ništa se ne prikazuje.

- s1 i s2 ukazuju na isti string
- s1 i s2 ukazuju na različite stringove

8. Šta se prikazuje na ekranu kao rezultat izvršavanja ovog programa?

```
public class Test {  
    public static void main(String[] args) {  
        String s1 = "Java";  
        String s2 = new String("Java");  
  
        if (s1 == s2)  
            System.out.println(  
                "s1 i s2 ukazuju na isti string");  
        else  
            System.out.println(  
                "s1 i s2 ukazuju na različite stringove");  
    }  
}
```

- Ništa se ne prikazuje.
- s1 i s2 ukazuju na isti string
- s1 i s2 ukazuju na različite stringove

9. Šta se prikazuje na ekranu kao rezultat izvršavanja ovog programa?

```
public class Test {  
    public static void main(String[] args) {  
        String s1 = "Java";  
        String s2 = s1;  
  
        if (s1.equals(s2))  
            System.out.println("s1 i s2 imaju isti sadržaj");  
        else  
            System.out.println("s1 i s2 imaju različit sadržaj");  
    }  
}
```

- Ništa se ne prikazuje.
- s1 i s2 imaju isti sadržaj
- s1 i s2 imaju različit sadržaj

10. Šta se prikazuje na ekranu kao rezultat izvršavanja ovog programa?

```
public class Test {  
    public static void main(String[] args) {  
        String s1 = "Java";  
        String s2 = "Java";  
  
        if (s1.equals(s2))  
            System.out.println("s1 i s2 imaju isti sadržaj");  
        else  
            System.out.println("s1 i s2 imaju različit sadržaj");  
    }  
}
```

- Ništa se ne prikazuje.
- s1 i s2 imaju isti sadržaj
- s1 i s2 imaju različit sadržaj

11. Šta se prikazuje na ekranu kao rezultat izvršavanja ovog programa?

```
public class Test {  
    public static void main(String[] args) {  
        String s1 = "Java";  
        String s2 = s1.toUpperCase();  
  
        if (s1 == s2)  
            System.out.println("s1 i s2 ukazuju na isti string");  
        else if (s1.equals(s2))  
            System.out.println("s1 i s2 imaju isti sadržaj");  
        else  
            System.out.println("s1 i s2 imaju različit sadržaj");  
    }  
}
```

- Ništa se ne prikazuje.
- s1 i s2 ukazuju na isti string
- s1 i s2 imaju isti sadržaj
- s1 i s2 imaju različit sadržaj

12. Koju vrednost ima promenljiva y posle izvršavanja ovog programskog fragmenta?

```
int x = 3, y = 3;  
switch (x + 3) {
```

```
case 6: y = 0;  
case 7: y = 1;  
default: y = y + 1;  
}
```

- 1
- 2
- 3
- 4

13. Šta se prikazuje na ekranu kao rezultat izvršavanja ovog programskog fragmenta?

```
char ch = 'a';  
switch (ch) {  
    case 'a':  
    case 'A':  
        System.out.print(ch); break;  
    case 'b':  
    case 'B':  
        System.out.print(ch); break;  
    case 'c':  
    case 'C':  
        System.out.print(ch); break;  
    case 'd':  
    case 'D':  
        System.out.print(ch);  
}
```

- abcd
- a
- aA
- A

14. Šta se prikazuje na ekranu kao rezultat izvršavanja ovog programskog fragmenta?

```
int ocena = 15;  
switch (ocena) {  
    case 0 :  
        System.out.println("ocena je 0");  
        break;
```

```
case 15 :  
    System.out.println("ocena je 15");  
case 30 :  
    System.out.println("ocena je 15 ili 30");  
    break;  
case 40 :  
    System.out.println("ocena je 40");  
default :  
    System.out.println("Pograšna ocena");  
}
```

- ocena je 15
- ocena je 15 u jednom redu i ocena je 15 ili 30 u sledećem redu.
- Ništa se neće prikazati.
- Pograšna ocena

15. Analizirajte sledeći programski fragment:

```
int x;  
double d = 1.5;  
switch (d) {  
    case 1.0: x = 1;  
    case 1.5: x = 2;  
    case 2.0: x = 3;  
}
```

- Programski fragment ima grešku, jer nedostaju potrebne naredbe break.
- Programski fragment ima grešku, jer nedostaje slučaj default u naredbi switch.
- Programski fragment ima grešku, jer kontrolna promenljiva d u naredbi switch ne može biti tipa double.
- Programski fragment nema grešaka.

16. Šta se prikazuje na ekranu kao rezultat izvršavanja ovog programskog fragmenta?

```
int k = 20;  
while (k > 0)  
    System.out.println(k);
```

- Programski fragment ima grešku i neće se izvršiti.
  - 20
  - Ništa se neće prikazati.
  - Stalno će se prikazivati 20 u beskonačnoj petlji.
17. Koliko puta se na ekranu prikazuje tekst Kako ste? kao rezultat izvršavanja ovog programskog fragmenta?

```
int brojač = 0;  
while (brojač < 10) {  
    System.out.println("Kako ste?");  
    brojač++;  
}
```

- 9
  - 10
  - 11
  - 0
18. Analizirajte sledeći programski fragment:

```
int brojač = 0;  
// Tačka A  
while (brojač < 10) {  
    System.out.println("Kako ste?");  
    brojač++;  
// Tačka B  
}  
// Tačka C
```

- Uslov brojač < 10 je uvek tačan u tački A.
  - Uslov brojač < 10 je uvek netačan u tački A.
  - Uslov brojač < 10 je uvek tačan u tački B.
  - Uslov brojač < 10 je uvek netačan u tački B.
  - Uslov brojač < 10 je uvek tačan u tački C.
  - Uslov brojač < 10 je uvek netačan u tački C.
19. Koliko puta se na ekranu prikazuje tekst Kako ste? kao rezultat izvršavanja ovog programskog fragmenta?

```
int brojač = 0;  
do {  
    System.out.println("Kako ste?");  
    brojač++;  
} while (brojač < 10);
```

- 9
- 10
- 11
- 0

20. Šta se prikazuje na ekranu kao rezultat izvršavanja ovog programskog fragmenta?

```
int i = 1;  
do {  
    i++;  
} while(i < 5);  
System.out.println("i = " + i);
```

- Programski fragment ima grešaku i neće se izvršiti.
- i = 5
- Ništa se neće prikazati.
- Stalno će se prikazivati i = 1 u beskonačnoj petlji.

21. Analizirajte sledeći programski fragment:

```
double suma = 0;  
for (double d = 0; d < 10;) {  
    d = d + 0.1;  
    suma = suma + d;  
}
```

- Programski fragment ima grešku, jer nedostaje treći deo (završnica) u zagradama for petlje.
- Programski fragment ima grešku, jer kontrolna promenljiva d u zagradama for petlje ne može biti tipa double.
- Pošto je uslov d < 10 uvek tačan, for petlja je beskonačna.
- Programski fragment nema grešaka i normalno se izvršava.

22. Šta se prikazuje na ekranu kao rezultat izvršavanja ovog programskog fragmenta?

```
int suma = 0;
for (int i = 0; i < 10; i++) {
    suma = suma + i;
}
System.out.println(suma);
```

- 10
- 11
- 12
- 13
- 45

23. Da li su ove dve for petlje ekvivalentne u smislu da daju istu vrednost promenljive suma nakon izvršavanja?

|                                                                     |                                                                     |
|---------------------------------------------------------------------|---------------------------------------------------------------------|
| <pre>for (int i = 0; i &lt; 10; ++i) {     suma = suma + i; }</pre> | <pre>for (int i = 0; i &lt; 10; i++) {     suma = suma + i; }</pre> |
|---------------------------------------------------------------------|---------------------------------------------------------------------|

- Da.
- Ne.

24. Da li je ova for petlja sintaksno ispravna?

```
for ( ; ; ) ;
```

- Da.
- Ne.

25. Analizirajte sledeći program:

```
public class Test {
    public static void main (String args[]) {
        int i = 0;
        for (i = 0; i < 10; i++);
            System.out.println(i + 4);
    }
}
```

- Program se neće izvršiti, jer se tačka-zapeta nalazi odmah iza zagrada for petlje.
- Program će se bez problema izvršiti i prikazaće se 4 na ekranu.
- Program će se bez problema izvršiti i prikazaće se 14 na ekranu.
- U programu je for petlja ekvivalentna sa ovom for petljom:

```
for (i = 0; i < 10; i++) { };
```

26. Da li će izvršavanje ovog programskog fragmenta biti beskonačno?

```
int stanje = 10;  
while (true) {  
    if (stanje < 9) break;  
    stanje = stanje - 9;  
}
```

- Da.
- Ne.

27. Koju vrednost ima promenljiva suma posle izvršavanja ovog programskog fragmenta?

```
int suma = 0;  
int i = 0;  
do {  
    i++;  
    suma = suma + i;  
    if (suma > 4) break;  
} while (i < 5);
```

- 5
- 6
- 7
- 8

28. Da li će izvršavanje ovog programskog fragmenta biti beskonačno?

```
int stanje = 10;  
while (true) {  
    if (stanje < 9) continue;  
    stanje = stanje - 9;  
}
```

- Da.
- Ne.

29. Šta se prikazuje na ekranu kao rezultat izvršavanja ovog programskog fragmenta?

```
int x = 5, y = 20;
while (y > 1) {
    y--;
    if (y % x != 0) continue;
    System.out.print(y + " ");
}
System.out.println();
```

- 20 19 18 17 16
- 20 15 10 5
- 15 10 5 0
- 15 10 5

30. Koja je sledeća naredba koja se izvršava nakon izvršavanja naredbe break spetlja u ovom programskom fragmentu?

```
spetlja:
for (int i = 1; i < 10; i++) {
    upetlja:
        for (int j = 1; j < 10; j++) {
            if (i * j > 50)
                break spetlja;
            System.out.println(i * j);
        }
    }
nastavak:
    . . .
```

- Naredba sa oznakom spetlja.
- Naredba sa oznakom upetlja.
- Naredba sa oznakom nastavak.
- Nijedna naredba, nego se program odmah završava.

31. Koja je sledeća naredba koja se izvršava nakon izvršavanja naredbe continue spetlja u ovom programskom fragmentu?

```
spetlja:  
    for (int i = 1; i < 10; i++) {  
        upetlja:  
            for (int j = 1; j < 10; j++) {  
                if (i * j > 50)  
                    continue spetlja;  
                System.out.println(i * j);  
                // Tačka A  
            }  
            // Tačka B  
        }  
    nastavak:  
    . . .
```

- Kontrola se prenosi u tačku B radi izvršavanja sledeće iteracije spoljašnje petlje sa oznakom spetlja.
- Kontrola se prenosi u tačku A radi izvršavanja sledeće iteracije unutrašnje petlje sa oznakom upetlja.
- Naredba sa oznakom nastavak.
- Nijedna naredba, nego se program odmah završava.

## Programski zadaci

1. Uskrs je pokretan crkveni praznik koji uvek pada u nedelju. Tačan datum katoličkog Uskrsa u godini između 1982. i 2048. godine dobija se na osnovu sledećeg postupka:

- 1)  $A$  je godina po modulu 19
- 2)  $B$  je godina po modulu 4
- 3)  $C$  je godina po modulu 7
- 4)  $D$  je  $(19A + 24)$  po modulu 30
- 5)  $E$  je  $(2B + 4C + 6D + 5)$  po modulu 7
- 6) Uskrs je  $(22 + D + E)$ . marta ili, ako je  $22 + D + E > 31$ , to je  $(22 + D + E - 31)$ . aprila

Napisati program koji prikazuje datum katoličkog Uskrsa za datu godinu između 1982. i 2048. godine.

2. Postupak za izračunavanje datuma katoličkog Uskrsa u prethodnom zadatku može se lako proširiti za sve godine između 1900. i 2099. godine. Naime, u slučaju četiri godine, 1954., 1981., 2049. i 2076., prethodna formula daje datum koji je 7 dana kasniji nego što treba da bude. Modifikujte program iz prethodnog zadatka tako da prikazuje datum katoličkog Uskrsa za datu godinu između 1900. i 2099. godine. (Napomena: u četiri posebne godine, oduzimanje 7 dana od datuma Uskrsa dobijenog po prvobitnoj formuli ne dovodi do promene meseca.)

3. Napisati program koji simulira bacanje novčića dati broj puta. Program treba da učita željeni broj bacanja novčića i da prikaže brojeve koliko puta je palo pismo i glava, kao i njihov količnik sa brojem bacanja. Program treba da ponavlja eksperiment sve dok se ne učita 0 za broj bacanja.

4. Napisati program koji simulira eksperiment bacanja dve kocke za igru i određuje koliko puta treba baciti par kocki dok na obe kocke ne padne istovremeno vrednost 1. (Inače, kockarski termin za ovaj ishod je „zmijske oči“.) Program treba da prikaže vrednosti kocki pri svakom bacanju i da na kraju prikaže ukupan broj bacanja kocki.

5. Niz brojeva za „ $3n + 1$  problem” počinje od datog celog broja  $n$ , dok se svaki sledeći broj niza dobija na sledeći način: ako je  $n$  paran broj, onda je to broj  $n$  podeljen sa 2; a ako je  $n$  neparan broj, onda je to broj  $n$  pomnožen sa 3 plus 1. Ovaj postupak za sledeći broj niza se ponavlja sve dok se ne dobije 1. Na primer, ako je početno dano  $n = 3$ , s obzirom da je 3 neparan broj, sledeći broj niza je  $3 \cdot 3 + 1 = 10$ . Kako je 10 paran broj, sledeći broj niza je  $10/2 = 5$ . Dalje, kako je 5 neparan broj, sledeći broj niza je  $3 \cdot 5 + 1 = 16$ . Nije teško proveriti da, nastavljajući ovaj postupak sve dok se ne dobije 1, kompletan niz za dati početni broj 3 jeste 3, 10, 5, 16, 8, 4, 2, 1.

Napisati program koji za dati početni broj  $n$  prikazuje niz brojeva za „ $3n + 1$  problem”.

6. Napisati program koji izračunava najveći zajednički delilac dva cela broja koristeći postupak traženja njihovog zajedničkog delioca počevši od manjeg od njih.

7. Napisati program koji određuje koji ceo broj između 1 i datog broja  $n$  ima najveći broj delioca i koliki je taj broj delioca. (Moguće je da više brojeva u ovom intervalu imaju isti, najveći broj delioca. Program treba da prikaže samo jedan od njih.)

8. Napisati program koji zvezdicama (znakom \*) crta geometrijsku figuru romb. Broj redova romba je ulazni podatak programa i ukoliko je to paran broj, treba ga zaokružiti na prvi veći neparan broj. Na primer, ako je broj redova romba 11, na ekranu se dobija sledeća slika:

```
*  
***  
*****  
*****  
*****  
*****  
*****  
*****  
*****  
*****  
***  
*
```

9. Napisati program koji učitava jedan red teksta i redom prikazuje njegove znakove brojeći pri tome broj slova i cifara koji se pojavljuju.
10. Napisati program koji učitava jedan red teksta i deli ga po rečima. Na primer, ako je ulazni red:

Profesor reče: "Student je položio ispit".

na ekranu se kao rezultat dobija:

Profesor  
reče  
Student  
je  
položio  
ispit

Jedna reč je niz susednih slova u ulaznom redu i treba da bude prikazana u posebnom redu. Svaki znak koji nije slovo u ulaznom redu se zanemaruje.

## *Glava*

# 5

## *Metodi*

### Pitanja

1. Ako metod ne vraća nijednu vrednost, koja se službena reč koristi za njegov tip rezultata u definiciji tog metoda?
  - void
  - return
  - public
  - static
2. Potpis metoda se sastoji od \_\_\_\_\_.
  - imena metoda
  - imena metoda i liste parametara
  - tipa rezultata, imena metoda i liste parametara
  - liste parametara
3. Analizirajte sledeći metod:

```
protected double nađi(int x, int y, boolean b) {  
    // . . .  
}
```

- Ime metoda je protected.
  - Ime metoda je nadī.
  - Tip rezultata metoda je int.
  - Tip rezultata metoda je boolean.
  - Tip rezultata metoda je double.
  - Broj parametara metoda je tri.
  - Broj parametara metoda je šest.
4. Svi Java programi moraju imati bar jedan metod \_\_\_\_\_.
- public static Main(String[] args)
  - public static Main(String args[])
  - public void main(String[] args)
  - public static void main(String[] args)
  - public static main(String[] args)
5. Argumenti pri pozivanju metoda se navode unutar \_\_\_\_\_.
- uglastih (srednjih) zagrada
  - običnih (malih) zagrada
  - vitičastih (velikih) zagrada
  - dvostrukih apostrofa
  - jednostrukih apostrofa
6. Vrednosti argumenata u pozivu metoda se prenose odgovarajućim parametrima metoda. Kako se naziva ovaj način prenošenja argumenata u Javi?
- pozivanje metoda.
  - prenošenje po vrednosti.
  - prenošenje po referenci.
  - prenošenje po imenu.
7. Koja se naredba koristi unutar tela nekog metoda za povratak iz tog metoda i vraćanje rezultata tog metoda?

- void
- return
- public
- static

8. Da li naredba return u ovom metodu dovodi do greške?

```
public static void main(String[] args) {  
    int max = 0;  
    if (max != 0)  
        System.out.println(max);  
    else  
        return;  
}
```

- Da.
- Ne.

9. Da li poziv metoda Math.pow() u ovom metodu dovodi do greške?

```
public static void main(String[] args) {  
    Math.pow(2, 4);  
}
```

- Da.
- Ne.

10. Šta je rezultat poziva nPrint("a", 4) ukoliko je metod nPrint() definišan na ovaj način?

```
void nPrint(String poruka, int n) {  
    while (n > 0) {  
        System.out.print(poruka);  
        n--;  
    }  
}
```

- Na ekranu se prikazuje aaaaa.
- Na ekranu se prikazuje aaaa.
- Na ekranu se prikazuje aaa.

- Na ekranu se prikazuje aa.
- Na ekranu se prikazuje a.

11. Ukoliko je metod nPrint() definisan na sledeći način:

```
void nPrint(String poruka, int n) {  
    while (n > 0) {  
        System.out.print(poruka);  
        n--;  
    }  
}
```

koju vrednost ima promenljiva k posle izvršavanja ovog programskog fragmenta?

```
int k = 2;  
nPrint("Java je kul!", k);
```

- 0
- 1
- 2
- 3

12. Analizirajte sledeći program:

```
public class Test {  
  
    public static void main(String[] args) {  
        System.out.println(xMetod(5));  
    }  
  
    public double xMetod(int n, double t) {  
        System.out.println("double");  
        return t;  
    }  
  
    public int xMetod(int n) {  
        System.out.println("int");  
        return n;  
    }  
}
```

- Program na ekranu prikazuje int i zatim 5.

- Program na ekranu prikazuje double i zatim 5.
- Program na ekranu prikazuje double.
- Program na ekranu prikazuje int.
- Program ima grešku, jer se ne može odrediti koju verziju preopterećenog metoda xMetod() treba pozvati.

13. Analizirajte sledeći program:

```
public class Test {  
  
    public static void main(String[] args) {  
        System.out.println(m(2));  
    }  
  
    public int m(int n) {  
        return n;  
    }  
  
    public void m(int n) {  
        System.out.println(n);  
    }  
}
```

- Program ima grešku, jer se ne može odrediti koju verziju preopterećenog metoda m() treba pozvati.
- Program ima grešku, jer je druga verzija preopterećenog metoda m() definisana ali se nigde ne poziva.
- Program se normalno izvršava i prikazuje 2 jedanput.
- Program se normalno izvršava i prikazuje 2 dvaput.

14. Kako se naziva promenljiva koja je deklarisana unutar nekog metoda?

- globalna promenljiva
- statička promenljiva
- blokovska promenljiva
- lokalna promenljiva

15. Koju vrednost ima promenljiva k posle izvršavanja ovog bloka?

```
{  
    int k = 2;  
    nPrint("Java je kul!", k);  
}
```

- 0
  - 1
  - 2
  - Promenljiva k nije definisana izvan tog bloka.
16. Koje su od ovih rečenica o rekurzivnim metodima tačne?
- Rekurzivni metodi su oni koji pozivaju sami sebe, bilo direktno ili indirektno.
  - Rekurzivni metodi se pozivaju drugačije od nerekurzivnih metoda.
  - Rekurzivni metodi rešavaju neki zadatak svođenjem polaznog problema na sličan prostiji problem (ili više njih).
  - Svaki rekurzivni metod mora imati *bazni slučaj* za najprostiji zadatak čije se rešenje ne dobija rekurzivnim pozivom.
17. Analizirajte sledeći rekurzivni metod:
- ```
public long faktorijel(int n) {  
    return n * faktorijel(n - 1);  
}
```
- Rezultat poziva faktorijel(3) je 2.
  - Rezultat poziva faktorijel(3) je 3.
  - Rezultat poziva faktorijel(3) je 6.
  - Poziv faktorijel(3) izaziva grešku pošto proizvodi beskonačan lanac poziva istog metoda faktorijel().
18. Analizirajte sledeći program:

```
public class Test {  
    public static void main(String[] args) {  
        int[] x = {1, 2, 3, 4, 5};  
        rMetod(x, 5);  
    }
```

```
public static void rMetod(int[] x, int n) {  
    System.out.print(x[n - 1] + " ");  
    rMetod(x, n - 1);  
}  
}
```

- Program na ekranu prikazuje 1 2 3 4 5.
- Program na ekranu prikazuje 1 2 3 4 5 i zatim grešku o prekoračenju granica indeksa niza x.
- Program na ekranu prikazuje 5 4 3 2 1.
- Program na ekranu prikazuje 5 4 3 2 1 i zatim grešku o prekoračenju granica indeksa niza x.

19. Analizirajte sledeći program:

```
public class Test {  
    public static void main(String[] args) {  
        rMetod(3);  
    }  
  
    public static void rMetod(int n) {  
        if (n > 1) {  
            System.out.print((n - 1) + " ");  
            rMetod(n - 1);  
        }  
    }  
}
```

- Program proizvodi beskonačan lanac poziva istog metoda rMetod().
- Program na ekranu prikazuje 1 2 3.
- Program na ekranu prikazuje 3 2 1.
- Program na ekranu prikazuje 1 2.
- Program na ekranu prikazuje 2 1.

20. Analizirajte sledeći program:

```
public class Test {  
    public static void main(String[] args) {  
        rMetod(2);  
    }  
}
```

```
public static void rMetod(int n) {  
    while (n > 1) {  
        System.out.print((n - 1) + " ");  
        rMetod(n - 1);  
    }  
}
```

- Program na ekranu ne prikazuje ništa.
- Program na ekranu prikazuje 1 2.
- Program na ekranu prikazuje 2 1.
- Program na ekranu beskonačno prikazuje 1 1 1 1 1 ....
- Program na ekranu beskonačno prikazuje 2 2 2 2 2 ....

21. Analizirajte sledeći rekurzivni metod:

```
public int rMetod(int n) {  
    if (n == 1)  
        return 1;  
    else  
        return n + rMetod(n - 1);  
}
```

- Pozivom rMetod(5) se isti metod rMetod() poziva još 3 puta.
- Pozivom rMetod(5) se isti metod rMetod() poziva još 4 puta.
- Pozivom rMetod(5) se isti metod rMetod() poziva još 5 puta.
- Pozivom rMetod(5) se isti metod rMetod() poziva još 6 puta.

22. Analizirajte sledeći rekurzivni metod:

```
public int rMetod(int n) {  
    if (n == 1)  
        return 1;  
    else  
        return n + rMetod(n - 1);  
}
```

- Rezultat poziva rMetod(5) je 5.
- Rezultat poziva rMetod(5) je 10.
- Rezultat poziva rMetod(5) je 15.

- Poziv rMetod(5) proizvodi beskonačan lanac poziva istog metoda rMetod().

23. Šta je bazni slučaj u ovom rekurzivnom metodu?

```
public int rMetod(int n) {  
    if (n == 1)  
        return 1;  
    else  
        return n + rMetod(n - 1);  
}
```

- $n$  je 1.
- $n$  je manje od 1.
- $n$  je veće od 1.
- Nema baznog slučaja.

24. Dopunite 4. red sledećeg rekurzivnog metoda za određivanje da li je neki string palindrom:

```
1 public static boolean palindrom(String s) {  
2     if (s.length() <= 1) // bazni slučaj  
3         return true;  
4     else if _____  
5         return false;  
6     else  
7         return palindrom(s.substring(1, s.length() - 1));  
8 }
```

- (`s.charAt(0) != s.charAt(s.length() - 1)`)
- (`s.charAt(0) != s.charAt(s.length())`)
- ((`s.charAt(1) != s.charAt(s.length() - 1)`)
- (`s.charAt(1) != s.charAt(s.length())`)

25. Dopunite 12. red sledećeg rekurzivnog metoda za određivanje da li je neki string palindrom:

```
1 public static boolean palindrom(String s) {  
2     return palindrom(s, 0, s.length() - 1);  
3 }
```

```
4
5  public static boolean palindrom(String s,
6                                int leviKraj, int desniKraj) {
7      if (desniKraj <= leviKraj) // bazni slučaj
8          return true;
9      else if (s.charAt(leviKraj) != s.charAt(desniKraj))
10         return false;
11     else
12         return _____ ;
13 }
```

- palindrom(s)
- palindrom(s, leviKraj, desniKraj)
- palindrom(s, leviKraj + 1, desniKraj)
- palindrom(s, leviKraj, desniKraj - 1)
- palindrom(s, leviKraj + 1, desniKraj - 1)

26. Dopunite 2. red sledećeg rekurzivnog metoda za sortiranje niza realnih brojeva u rastućem redosledu:

```
1  public static void sortiraj(double[] niz) {
2      _____ ;
3  }
4
5  public static void sortiraj(double[] niz, int n) {
6      if (n > 1) {
7          // Nalaženje najvećeg elementa niza i njegovog indeksa
8          int iMax = 0;
9          double max = niz[0];
10         for (int i = 1; i <= n; i++) {
11             if (niz[i] > max) {
12                 max = niz[i];
13                 iMax = i;
14             }
15
16             // Zamena najvećeg i poslednjeg elementa niza
17             niz[iMax] = niz[n];
18             niz[n] = max;
19
20             // Sortiranje prvog dela niza
21             sortiraj(niz, n - 1);
22         }
23     }
```

- sortiraj(niz)
- sortiraj(niz, niz.length)
- sortiraj(niz, niz.length - 1)
- sortiraj(niz, niz.length + 1)

27. Dopunite 15. red sledećeg rekurzivnog metoda za binarno pretraživanje sortiranog celobrojnog niza:

```
1  public static int nađiBroj(int[] niz, int broj) {  
2      return nađiBroj(niz, broj, 0, niz.length - 1);  
3  }  
4  
5  public static int nađiBroj(int[] niz, int broj,  
6                           int leviKraj, int desniKraj) {  
7      if (leviKraj > desniKraj) // broj nije nađen u nizu  
8          return -1;  
9  
10     // Pretraživanje prve ili druge polovine niza  
11     int sredina = (leviKraj + desniKraj) / 2;  
12     if (broj < niz[sredina])  
13         return nađiBroj(niz, broj, leviKraj, sredina - 1);  
14     else if (broj > niz[sredina])  
15         return _____;  
16     else  
17         return sredina;  
18 }
```

- nađiBroj(niz, broj, sredina + 1, leviKraj)
- nađiBroj(niz, broj, sredina - 1, leviKraj)
- nađiBroj(niz, broj, desniKraj, sredina + 1)
- nađiBroj(niz, broj, sredina + 1, desniKraj)

## Programski zadaci

1. Napisati metod `NZD()` koji izračunava najveći zajednički delilac dva cela broja koristeći Euklidov algoritam. Testirati taj metod tako što se u metodu `main()` učitavaju dva cela broja i prikazuje njihov najveći zajednički delilac pozivom metoda `NZD()`.
2. Napisati metod `triN1()` koji prikazuje niz brojeva za „ $3n + 1$  problem” (videti 5. zadatak iz glave 4). Testirati taj metod tako što se u metodu `main()` učitava početni broj niza i prikazuje ostatak tog niza pozivom metoda `triN1()`.
3. Napisati metod `kapitalizuj()` koji početno slovo svake reči datog stringa pretvara u veliko slovo. Testirati taj metod tako što se u metodu `main()` učitava jedan red teksta i prikazuje njegova „kapitalizovana” verzija pozivom metoda `kapitalizuj()`.
4. Heksadekadne „cifre” su dekadne cifre od 0 do 9 i slova  $A, B, C, D, E$  i  $F$ . U heksadekadnom sistemu ovi znakovi predstavljaju vrednosti od 0 redom do 15.

Napisati metod `heksVrednost()` koji daje heksadekadnu vrednost datog znaka. Ako parametar tog metoda nije jedan od dozvoljenih znakova, vraćena vrednost treba da bude  $-1$ .

Heksadekadni broj je niz heksadekadnih cifara kao što su  $34A7, FF, ABCD$  ili  $172300$ . Napisati program koji učitava heksadekadni broj i prikazuje dekadnu vrednost tog broja koristeći metod `heksVrednost()`. Ako svi znakovi heksadekadnog broja nisu dozvoljene heksadekadne cifre, program treba da prikaže odgovarajuću poruku o grešci.

5. Napisati metod `baciZaZbir()` koji simulira bacanje dve kocke za igru dok njihov zbir ne padne jednak datom mogućem broju od 2 do 12. Rezultat ovog metoda treba da bude broj bacanja koji je izvršen dok se nije desio željeni ishod.  
Napisati drugi metod `prosekZaZbir()` koji koristi metod `baciZaZbir()` za ponavljanje 100000 puta eksperimenta bacanja dve kocke dok se ne

dobije dati zbir. Parametar metoda je željeni zbir svakog bacanja, a rezultat metoda je prosečan broj bacanja koji se dobija za taj zbir u 100000 pokusa.

Na kraju, napisati metod main() koji poziva metod prosekZaZbir() za svaki mogući zbir od 2 do 12 i rezultate prikazuje u tabeli sledećeg oblika:

Zbir dve kocke	Prosečan broj bacanja
2	35.87
3	18.08
4	11.95
.	.
.	.
.	.



*Glava*

# 6

## *Klase i objekti*

### Pitanja

1. \_\_\_\_\_ je šablon za konstruisanje objekata istog tipa.
  - Objekat
  - Metod
  - Promenljiva
  - Klasa
  
2. Članovi klase (polja i metodi) mogu biti \_\_\_\_\_.
  - statički (klasni) i nestatički (objektni)
  - lokalni i globalni
  - proceduralni i neproceduralni
  - spoljašnji i unutrašnji
  
3. Za definisanje klase se koristi službena reč \_\_\_\_\_.
  - method
  - class
  - main
  - object

4. Kako se naziva specijalni metod neke klase koji se poziva prilikom konstruisanja svakog objekta te klase?

- Glavni metod
- Metod bez argumenata
- Konstruktor
- Rekursivni metod

5. Koje su od ovih rečenica o konstruktorima tačne?

- Podrazumevani konstruktor bez argumenata se automatski dodaje ukoliko u klasi nije eksplisitno definisan nijedan konstruktor.
- U klasi se mora eksplisitno definisati bar jedan konstruktor.
- Konstruktori nemaju tip rezultata, čak ni void.
- Konstruktori moraju imati isto ime kao klasa u kojoj se definišu.
- Konstruktori se pozivaju koristeći operator new kada se konstruiše objekat.

6. Analizirajte sledeći program koji se sastoji od dve klase u jednoj datoteci:

```
public class Test {  
    public static void main(String[] args) {  
        A a = new A();  
        a.prikaži();  
    }  
}  
  
class A {  
    String s;  
  
    public A(String s) {  
        this.s = s;  
    }  
    public void prikaži() {  
        System.out.println(s);  
    }  
}
```

- Program ima grešku, jer klasa A nije javna klasa.
- Program ima grešku, jer klasa A nema podrazumevani konstruktor.

- Program nema grešaka i normalno se izvršava ništa ne prikazujući na ekranu.
- Program ima grešku koja se može ispraviti ukoliko se naredba A a = new A(); u metodu main promeni u naredbu A a = new A("poruka");.

7. Analizirajte sledeći program koji se sastoji od dve klase u jednoj datoteci:

```
public class Test {  
    public static void main(String[] args) {  
        B c = new B(2);  
    }  
}  
  
class B {  
    int i;  
  
    public void b(int j) {  
        i = j;  
    }  
}
```

- Program ima grešku, jer klasa B nije javna klasa.
  - Program ima grešku, jer klasa B nema podrazumevani konstruktor.
  - Program ima grešku, jer klasa B nema konstruktor sa parametrom tipa int.
  - Program nema grešaka i normalno se izvršava ništa ne prikazujući na ekranu.
8. Ako je data deklaracija Krug k = new Krug(), koja je od ovih rečenica najjačnija?
- Promenljiva k sadrži celobrojnu vrednost.
  - Promenljivoj k se može dodeliti celobrojna vrednost.
  - Promenljiva k sadrži objekat klase Krug.
  - Promenljiva k sadrži referencu na objekat klase Krug.
9. Analizirajte sledeći program:

```
public class Test {  
  
    int x;  
  
    public Test(String s) {  
        System.out.println("Test");  
    }  
  
    public static void main(String[] args) {  
        Test t = null;  
        System.out.println(t.x);  
    }  
}
```

- Program ima grešku, jer promenljiva t nije inicijalizovana.
  - Program ima grešku, jer promenljiva x nije inicijalizovana.
  - Program ima grešku, jer klasa Test nema podrazumevani konstruktor.
  - Program ima grešku, jer se u nekoj klasi ne može deklarisati promenljiva tipa te iste klase kao što je to ovde slučaj sa promenljivom t.
  - Program ima grešku, jer promenljiva t ima vrednost null kada se prikazuje polje t.x.
  - Program nema grešaka i normalno se izvršava ništa ne prikazujući na ekranu.
10. Automatske početne vrednosti za polja logičkog, numeričkog i klasnog tipa svakog objekta su, redom, \_\_\_\_\_ .
- true, 1, null
  - false, 0, null
  - true, 0, null
  - false, 1, null
  - false, 0, void
11. Koje su od ovih rečenica o promenljivima tačne?
- Lokalne promenljive ne dobijaju automatski početne vrednosti.
  - Polja objekata dobijaju automatski početne vrednosti.

- Promenljiva nekog primitivnog tipa sadrži vrednost tog primitivnog tipa.
- Promenljiva nekog klasnog tipa ukazuje na memorijsku adresu u kojoj se nalazi objekat klase.
- Celobrojna vrednost koja predstavlja važeću memorijsku adresu može se dodeliti promenljivoj klasnog tipa.

12. Analizirajte sledeći program:

```
public class Test {  
  
    public static void main(String[] args) {  
        double prečnik;  
        final double PI= 3.15169;  
        double površina = prečnik * prečnik * PI;  
        System.out.println("Površina je " + površina);  
    }  
}
```

- Program ima grešku, jer promenljiva prečnik nije inicijalizovana.
- Program ima grešku, jer je konstanta PI definisana unutar metoda.
- Program ima grešku, jer konstanta PI ima previše decimala.
- Program ima grešku, jer konstanta PI ima premalo decimala.
- Program nema grešaka i normalno se izvršava.

13. Analizirajte sledeći program:

```
public class Test {  
  
    int x;  
  
    public Test(String s) {  
        System.out.println("Test");  
    }  
  
    public static void main(String[] args) {  
        Test t = new Test();  
        System.out.println(t.x);  
    }  
}
```

- Program ima grešku, jer se metod System.out.println() ne može koristiti u konstruktoru klase.
  - Program ima grešku, jer promenljiva x nije inicijalizovana.
  - Program ima grešku, jer klasa Test nema podrazumevani konstruktor.
  - Program ima grešku, jer se u nekoj klasi ne može konstruisati objekat te iste klase.
  - Program nema grešaka i normalno se izvršava prikazujući 0 na ekranu.
14. Koja je od ovih rečenica o objektima najtačnija?
- Objektna promenljiva sadrži neki objekat.
  - Promenljiva klasnog tipa sadrži neki objekat.
  - Neki objekat može sadržati druge objekte.
  - Neki objekat može sadržati reference na druge objekte.
15. Koja su polja zajednička i jedinstvena za sve objekte neke klase?
- Javna
  - Privatna
  - Objektna (instancna)
  - Statička (klasna)
16. Da li se statičko polje neke klase može koristiti bez konstruisanja i jednog objekta te klase?
- Da.
  - Ne.
17. U kojem redu treba zameniti znak ? službenom rečju static u definiciji ove klase?

```
1  public class Test {  
2  
3      private int broj;  
4  
5      public ? int kvadrat(int n) {  
6          return n * n;
```

```
7      }
8
9      public ? int getBroj() {
10         return broj;
11     }
12 }
```

- U redu 5.
- U redu 9.
- U oba reda 5 i 9.
- U nijednom redu.
18. Metod koji se pridružuje svakom pojedinačnom objektu neke klase naziva se \_\_\_\_\_.  
 statički metod  
 klasni metod  
 objektni metod  
 glavni metod
19. U kojem je od ovih slučajeva ispravno deklarisana konstanta MAX\_CENA kao članica klase?  
 final static MAX\_CENA = 99.98;  
 final static float MAX\_CENA = 99.98;  
 static double MAX\_CENA = 99.98;  
 final double MAX\_CENA = 99.98;  
 final static double MAX\_CENA = 99.98;
20. Analizirajte sledeći program:

```
public class Test {

    public static void main(String[] args) {
        int n = 2;
        xMetod(n);
        System.out.println("n je " + n);
    }

    void xMetod(int n) {
```

```
        n++;
    }
}
```

- Program ima grešku, jer metod xMetod() ne vraća nijednu vrednost.
  - Program ima grešku, jer metod xMetod() nije definisan da bude statički.
  - Program prikazuje n je 1 na ekranu.
  - Program prikazuje n je 2 na ekranu.
  - Program prikazuje n je 3 na ekranu.
21. Šta se prikazuje drugom naredbom println u metodu main prilikom izvršavanja ovog programa?
- ```
public class Test {  
  
    int i;  
    static int s;  
  
    public static void main(String[] args) {  
        Test t1 = new Test();  
        System.out.println("t1.i je " + t1.i + ", t1.s je " + t1.s);  
        Test t2 = new Test();  
        System.out.println("t2.i je " + t2.i + ", t2.s je " + t2.s);  
        Test t3 = new Test();  
        System.out.println("t3.i je " + t3.i + ", t3.s je " + t3.s);  
    }  
  
    public Test() {  
        i++;  
        s++;  
    }  
}
```
- t2.i je 1, t2.s je 1
  - t2.i je 1, t2.s je 2
  - t2.i je 2, t2.s je 2
  - t2.i je 2, t2.s je 1
22. Šta se prikazuje trećom naredbom println u metodu main prilikom izvršavanja ovog programa?

```
public class Test {  
  
    int i;  
    static int s;  
  
    public static void main(String[] args) {  
        Test t1 = new Test();  
        System.out.println("t1.i je " + t1.i + ", t1.s is " + t1.s);  
        Test t2 = new Test();  
        System.out.println("t2.i je " + t2.i + ", t2.s je " + t2.s);  
        Test t3 = new Test();  
        System.out.println("t3.i je " + t3.i + ", t3.s je " + t3.s);  
    }  
  
    public Test() {  
        i++;  
        s++;  
    }  
}
```

- t3.i je 1, t3.s je 1
- t3.i je 1, t3.s je 2
- t3.i je 1, t3.s je 3
- t3.i je 3, t3.s je 1
- t3.i je 3, t3.s je 3

23. Analizirajte sledeći program koji se sastoji od dve klase u jednoj datoteci:

```
public class Test {  
  
    public static void main(String[] args) {  
        A a = new A();  
        a.n++;  
    }  
}  
  
class A {  
    int n;  
    private A() {  
    }  
}
```

- Program ima grešku, jer klasa A ima privatni podrazumevani konstruktor.

- Program ima grešku, jer klasa A ima prazan podrazumevani konstruktor.
  - Program ima grešku, jer promenljiva n nije inicijalizovana.
  - Program nema grešaka i normalno se izvršava.
24. Koja se vrednost polja b.n prikazuje prvom naredbom println prilikom izvršavanja ovog programa?

```
public class Test {  
  
    public static void main(String[] args) {  
        int k = 0;  
        Brojač b = new Brojač();  
  
        for (int i = 0; i < 100; i++)  
            uvećaj(b, k);  
        System.out.println("b.n = " + b.n);  
        System.out.println("k = " + k);  
    }  
  
    public static void uvećaj(Brojač b, int k) {  
        b.n++;  
        k++;  
    }  
}  
  
class Brojač {  
    int n;  
  
    public Brojač(int n) {  
        this.n = n;  
    }  
  
    public Brojač() {  
        this.n = 1;  
    }  
}
```

- b.n = 101
- b.n = 100
- b.n = 99
- b.n = 98
- b.n = 0

25. Koja se vrednost promenljive k prikazuje drugom naredbom println prilikom izvršavanja ovog programa?

```
public class Test {  
  
    public static void main(String[] args) {  
        int k = 0;  
        Brojač b = new Brojač();  
  
        for (int i = 0; i < 100; i++)  
            uvećaj(b, k);  
        System.out.println("b.n = " + b.n);  
        System.out.println("k = " + k);  
    }  
  
    public static void uvećaj(Brojač b, int k) {  
        b.n++;  
        k++;  
    }  
}  
  
class Brojač {  
    int n;  
  
    public Brojač(int n) {  
        this.n = n;  
    }  
  
    public Brojač() {  
        this.n = 1;  
    }  
}
```

- k = 101
- k = 100
- k = 99
- k = 98
- k = 0

26. Analizirajte sledeću klasu Krug:

```
public class Krug {  
  
    private double prečnik;
```

```
public Krug(double prečnik) {  
    prečnik = prečnik;  
}  
}
```

- Klasa Krug ima grešku, jer nema metod main().
- Svaki konstruisani objekat klase Krug će imati prečnik 0. Na primer, naredbom Krug k = new Krug(2.35) dobija se krug k prečnika 0 iako se očekuje da njegov prečnik bude 2.35.
- Klasa Krug ima grešku, jer se ne može pisati naredba dodele prečnik = prečnik; u konstruktoru.
- Klasa Krug ima grešku, jer nema podrazumevani konstruktor.

27. Analizirajte sledeći program:

```
public class Test {  
  
    private double x;  
  
    public Test(double x) {  
        this.t();  
        this.x = x;  
    }  
  
    public Test() {  
        System.out.println("Podrazumevani konstruktor");  
        this(23);  
    }  
  
    public void t() {  
        System.out.println("Poziv metoda t()");  
    }  
}
```

- this.t() u konstruktoru Test(double x) može se pojednostaviti i zameniti samo sa t().
- this.x u konstruktoru Test(double x) može se pojednostaviti i zameniti samo sa x.
- this(23) u konstruktoru Test() mora se pozvati pre naredbe  
System.out.println("Podrazumevani konstruktor");.
- this(23) u konstruktoru Test() mora se zameniti sa this(23.0).

## Programski zadaci

1. Napisati program koji prikazuje mesečni kalendar ukoliko su dati mesec i godina. Za predstavljanje kalendara iskoristiti klase Calendar i GregorianCalendar iz paketa java.util.
2. Napisati klase Tačka, Kvadrat i Krug koje predstavljaju tačku, kvadrat i krug u koordinatnom sistemu ravni. Klasa Krug treba da sadrži metod koji daje opisan kvadrat datog kruga. Napisati jednostavan program koji testira napisane klase.
3. Napisati klasu KompleksanBroj koja predstavlja kompleksan broj sa realnim i imaginarnim delom. Klasa KompleksanBroj treba da sadrži metode kojima se realizuju uobičajene operacije nad kompleksnim brojevima. Napisati jednostavan glavni metod main() u kojem se testiraju napisani metodi.
4. Napisati klasu RimskiBroj koja predstavlja rimski broj. (Prepostavite da se rimski brojevi *ne* zapisuju u kondenzovanom obliku: na primer, broj 49 se piše XXXXVIIII umesto XLIX.) Klasa RimskiBroj treba da sadrži metode kojima se realizuju operacije sabiranja i množenja rimskih brojeva. Napisati jednostavan glavni metod main() u kojem se testiraju napisani metodi.
5. Ponovo uraditi 5. zadatak iz glave 5, ali tako da program sadrži posebnu klasu KockaZaIgru koja predstavlja kocku za igranje.
6. Napisati program koji simulira bacanje novčića dati broj puta i prikazuje koliko puta su pali „pismo“ i „glava“. Program treba da koristi posebnu klasu Brojač koja predstavlja opšti brojač za brojanje 0, 1, 2, 3, ... .



# 7

## *Osnovne strukture podataka*

### Pitanja

1. Koje je ime trećeg elementa u nizu pod nazivom a?
  - a[2]
  - a(2)
  - a[3]
  - a(3)
2. Koje su od ovih deklaracija niza a pogrešne?
  - int[] a = new int[2];
  - int[] a = new int(2);
  - int a = new int[2];
  - int a() = new int[2];
3. Ako je data deklaracija int i = 5, koji se od ovih izraza mogu koristiti za indekse elemenata niza double[] d = new double[100]?
  - i
  - (int)(Math.random() \* 100)
  - (int)(Math.random() \* 100 + 1)

- Math.random() \* 100
- i + 10
- i + 6.5

4. Analizirajte sledeći program:

```
public class Test {  
  
    public static void main(String[] args) {  
        int[] a = new int[3];  
        System.out.println("a[0] je " + a[0]);  
    }  
}
```

- Program ima grešku, jer je dužina niza a premala.
- Program ima grešku, jer elementi niza a nisu inicijalizovani.
- Program ima grešku, jer element a[0] nije definisan.
- Program nema grešaka i normalno se izvršava prikazujući a[0] je 0 na ekranu.

5. Koje su od ovih deklaracija nizova u Javi ispravne?

- int i = new int(30);
- double[] d = new double[30];
- int[] i = {3, 4, 3, 2};
- char[] c = new char();
- char[] c = new char{'a', 'b', 'c', 'd'};
- char[] c = {'a', 'b'};

6. Ako je data deklaracija int[] a = {1, 2, 3, 4}, koja je vrednost izraza a.length?

- 0
- 3
- 4
- 5

7. Analizirajte sledeći program:

```
public class Test {  
  
    public static void main(String[] args) {  
        int[] a = new int[5];  
        int i;  
        for (i = 0; i < a.length; i++)  
            a[i] = i;  
        System.out.print(a[i] + " ");  
    }  
}
```

- Program prikazuje 0 1 2 3 4 na ekranu.
- Program prikazuje 4 na ekranu.
- Program ima grešku, jer će se koristiti nepostojeći element a[5] u poslednjoj naredbi print u metodu main.
- Program ima grešku, jer promenljiva i u poslednjoj naredbi print u metodu main neće imati nijednu vrednost.

8. Šta je rezultat izvršavanja ovog programa?

```
public class Test {  
  
    public static void main(String[] args) {  
        int[] a = {120, 200, 016};  
        for (int i = 0; i < a.length; i++)  
            System.out.print(a[i] + " ");  
    }  
}
```

- Program prikazuje 120 200 16 na ekranu.
- Program prikazuje 120 200 14 na ekranu.
- Program prikazuje 120 200 22 na ekranu.
- Program ima grešku, jer umesto 016 treba pisati 16.

9. Šta se prikazuje na ekranu za vrednosti niza lista2 kao rezultat izvršavanja ovog programa?

```
public class Test {  
  
    public static void main(String[] args) {  
        int[] lista1 = {1, 2, 3};  
        int[] lista2 = {lista1[0], lista1[1], lista1[2]};  
        System.out.print(lista2[0] + " " + lista2[1] + " " + lista2[2]);  
    }  
}
```

```
int[] lista2 = {1, 2, 3};  
lista2 = lista1;  
lista1[0] = 0; lista1[1] = 1; lista2[2] = 2;  
  
for (int i = 0; i < lista2.length; i++)  
    System.out.print(lista2[i] + " ");  
}  
}
```

- 1 2 3
- 1 1 1
- 0 1 2
- 0 1 3

10. Šta se prikazuje na ekranu za vrednosti niza lista1 kao rezultat izvršavanja ovog programa?

```
public class Test {  
  
    public static void main(String[] args) {  
        int[] lista1 = {1, 2, 3};  
        int[] lista2 = {1, 2, 3};  
        lista2 = lista1;  
        lista1[0] = 0; lista1[1] = 1; lista2[2] = 2;  
  
        for (int i = 0; i < lista1.length; i++)  
            System.out.print(lista1[i] + " ");  
    }  
}
```

- 1 2 3
- 1 1 1
- 0 1 2
- 0 1 3

11. Šta se prikazuje na ekranu kao rezultat izvršavanja ovog programa?

```
public class Test {  
  
    public static void main(String[] args) {  
        int[] x = {1, 2, 3, 4};  
        int[] y = x;
```

```
x = new int[2];  
  
for (int i = 0; i < y.length; i++)  
    System.out.print(y[i] + " ");  
}  
}
```

- 1 2 3 4
- 0 0
- 0 0 3 4
- 0 0 0 0

12. Šta se prikazuje na ekranu kao rezultat izvršavanja ovog programa?

```
public class Test {  
  
    public static void main(String[] args) {  
        int[] x = {1, 2, 3, 4};  
        int[] y = x;  
  
        x = new int[2];  
  
        for (int i = 0; i < x.length; i++)  
            System.out.print(x[i] + " ");  
    }  
}
```

- 1 2 3 4
- 0 0
- 0 0 3 4
- 0 0 0 0

13. Analizirajte sledeći program:

```
public class Test {  
  
    public static void main(String[] args) {  
        final int[] x = {1, 2, 3, 4};  
        int[] y = x;  
  
        x = new int[2];
```

```
    for (int i = 0; i < y.length; i++)
        System.out.print(y[i] + " ");
    }
}
```

- Program prikazuje 1 2 3 4 na ekranu.
- Program prikazuje 0 0 na ekranu.
- Program ima grešku kod naredbe x = new int[2], jer je promenljiva x deklarisana da bude final i ne može se menjati.
- Elementi niza x se ne mogu menjati, jer je promenljiva x deklarisana da bude final.

14. Analizirajte sledeći programski fragment:

```
int[] lista = new int[5];
lista = new int[10];
```

- Programski fragment ima grešku, jer se promenljiva lista ne može menjati nakon što joj se dodeli vrednost.
- Programski fragment nema grešaka i normalno se izvršava. Drugom naredbom se novi niz dodeljuje promenljivoj lista.
- Programski fragment ima grešku, jer se promenljivoj lista dodeljuje novi niz.
- Programski fragment ima grešku, jer se promenljivoj lista dodeljuje novi niz različite dužine od prvog.

15. Analizirajte sledeći program:

```
public class Test {

    public static void main(String[] args) {
        int[] a = new int[4];
        a[1] = 1;

        a = new int[2];

        System.out.println("a[1] je " + a[1]);
    }
}
```

- Program ima grešku kod naredbe `a = new int[2]`, jer se novi niz dodeljuje promenljivoj `a`.
  - Program ima grešku kod naredbe `println`, jer `a[1]` nije inicijalizovan.
  - Program na ekranu prikazuje `a[1]` je 0.
  - Program na ekranu prikazuje `a[1]` je 1.
16. Naredbom \_\_\_\_\_ se kopija niza a dodeljuje nizu b.
- `b = Arrays.copyOf(a, a.length);`
  - `b = Arrays.copyOf(a);`
  - `Arrays.copyOf(b, a, a.length);`
  - `Arrays.copyOf(a, b);`
17. Kada se dati niz prenosi nekom metodu kao argument, tom metodu se zapravo prenosi \_\_\_\_\_.
- kopija datog niza
  - kopija prvog elementa datog niza
  - dužina datog niza
  - referenca na dati niz
18. Šta se prikazuje na ekranu kao rezultat izvršavanja ovog programa?

```
public class Test {  
  
    public static void main(String[] args) {  
        int[] x = {1, 2, 3, 4, 5};  
        uvećaj(x);  
  
        int[] y = {1, 2, 3, 4, 5};  
        uvećaj(y[0]);  
  
        System.out.println(x[0] + " " + y[0]);  
    }  
  
    public static void uvećaj(int[] a) {  
        for (int i = 0; i < a.length; i++)  
            a[i]++;  
    }  
}
```

```
public static void uvecaj(int n) {  
    n++;  
}  
}
```

- Poruka o greški.
- 1 1
- 2 2
- 2 1
- 1 2
19. Šta se prikazuje na ekranu za vrednosti niza lista kao rezultat izvršavanja ovog programa?

```
public class Test {  
  
    public static void main(String[] args) {  
        int[] lista = {1, 2, 3, 4, 5};  
  
        obrniNiz(lista);  
  
        for (int i = 0; i < lista.length; i++)  
            System.out.print(lista[i] + " ");  
    }  
  
    public void obrniNiz(int[] a) {  
        int[] b = new int[a.length];  
  
        for (int i = 0; i < a.length; i++)  
            b[i] = a[a.length - 1 - i];  
  
        a = b;  
    }  
}
```

- 1 2 3 4 5
- 5 4 3 2 1
- 5 4 1 2 3
- 1 2 5 4 3

20. Analizirajte sledeći program:

```
public class Test {  
  
    public static void main(String[] args) {  
        xMetod(new double[]{3, 3});  
        xMetod(new double[5]);  
        xMetod(new double[3]{1, 2, 3});  
    }  
  
    public void xMetod(double[] a) {  
        System.out.println(a.length);  
    }  
}
```

- Program ima grešku, jer je nepravilan argument new double[]{3, 3} u prvom pozivu metoda xMetod().
  - Program ima grešku, jer je nepravilan argument new double[5] u drugom pozivu metoda xMetod().
  - Program ima grešku, jer je nepravilan argument new double[3]{1, 2, 3} u trećem pozivu metoda xMetod().
  - Program ima grešku, jer će sve vrednosti niza a imati vrednost null prilikom izvršavanja drugog poziva metoda xMetod().
21. Kako se naziva deo memorije u kojem se smeštaju nizovi, kao i svi objekti? U tom delu se, radi efikasnosti, ne vodi mnogo računa o redu po kojem se zauzima slobodna i oslobađa zauzeta memorija.
- stek memorija
  - hip memorija
  - keš memorija
  - virtuelna memorija
22. Šta se tačno dobija kada se jedan niz vraća kao rezultat nekog metoda?
- kopija tog niza
  - kopija prvog elementa tog niza
  - dužina tog niza
  - referenca na taj niz
23. Ako je dato zaglavlje metoda `public static int[] xMetod()`, koja se od ovih naredbi `return` može koristiti u telu metoda `xMetod()`?

- return 1;
- return {1, 2, 3};
- return int[]{1, 2, 3};
- return new int[]{1, 2, 3};

24. Šta se prikazuje na ekranu za vrednosti niza lista kao rezultat izvršavanja ovog programa?

```
public class Test {

    public static void main(String[] args) {
        int[] lista = {1, 2, 3, 4, 5};

        lista = obrniNiz(lista);

        for (int i = 0; i < lista.length; i++)
            System.out.print(lista[i] + " ");
    }

    public int[] obrniNiz(int[] a) {
        int[] b = new int[a.length];

        for (int i = 0; i < a.length; i++)
            b[i] = a[a.length - 1 - i];

        return b;
    }
}
```

- 1 2 3 4 5
- 5 4 3 2 1
- 5 4 1 2 3
- 1 2 5 4 3

25. Šta se prikazuje na ekranu za vrednosti niza lista2 kao rezultat izvršavanja ovog programa?

```
public class Test {

    public static void main(String[] args) {
        int[] listal = {1, 2, 3, 4, 5};
```

```
int[] lista2 = obrniNiz(lista1);

for (int i = 0; i < lista2.length; i++)
    System.out.print(lista2[i] + " ");
}

public int[] obrniNiz(int[] a) {
    int[] b = new int[a.length];

    for (int i = 0; i < a.length; i++)
        b[i] = a[a.length - 1 - i];

    return b;
}
```

1 2 3 4 5

5 4 3 2 1

5 4 1 2 3

1 2 5 4 3

26. Ako je data deklaracija Krug[] k = new Krug[10], koja je od ovih rečenica najtačnija?

- Promenljiva k sadrži niz od 10 celobrojnih vrednosti.
- Promenljiva k sadrži niz od 10 objekata klase Krug.
- Promenljiva k sadrži referencu na niz od 10 promenljivih klasnog tipa Krug.
- Promenljiva k sadrži objekat klase Krug prečnika 10.

27. Ako je zaglavje glavnog metoda klase Test dano u standardnom obliku public static void main(String[] args), koji element niza args dobija vrednost stringa "abc" ukoliko je izvršavanje klase Test pokrenuto ovom DOS komandom?

```
java Test "+" 3 "abc" 2
```

args[0]

args[1]

args[2]

- args[3]

28. Koja su od ovih zaglavlja metoda prikaži() sa promenljivim brojem argumenata ispravne?

- public void prikaži(String... niska, double... broj)
- public void prikaži(double... broj, String ime)
- public void double... prikaži(double d1, double d2)
- public void prikaži(double... broj)
- public void prikaži(int n, double... broj)

29. Analizirajte sledeći program:

```
public class Test {  
  
    public static void main(String[] args) {  
  
        double[] d = {1.0, 2.0, 3.0};  
  
        System.out.println(prosek(d));  
        System.out.println(prosek(1, 2, 2, 1, 4));  
        System.out.println(prosek(new double[]{1, 2, 3}));  
        System.out.println(prosek(1.0, 2.0, 2.0, 1.0));  
    }  
  
    public static double prosek (double... brojevi) {  
        double zbir = 0;  
        for (double e : brojevi)  
            zbir = zbir + e;  
        return zbir / brojevi.length;  
    }  
}
```

- Program ima grešku, jer je nepravilan poziv prosek(d) u prvoj naredbi println.
- Program ima grešku, jer je nepravilan poziv prosek(1, 2, 2, 1, 4) u drugoj naredbi println.
- Program ima grešku, jer je nepravilan poziv prosek(new double[]{1, 2, 3}) u trećoj naredbi println.
- Program ima grešku, jer je nepravilan poziv prosek(1.0, 2.0, 2.0, 1.0) u četvrtoj naredbi println.

- Program se izvršava bez greške i prosek datih brojeva se tačno izračunava.
  - Program se izvršava bez greške, ali se prosek datih brojeva ne izračunava tačno.
30. Pozivom \_\_\_\_\_ sortira se niz lotoBrojevi tipa int[].
- Arrays(lotoBrojevi)
  - Arrays.sort(lotoBrojevi)
  - Arrays.sorts(lotoBrojevi)
  - Arrays.sortArray(lotoBrojevi)
31. Ako je data deklaracija int[] lotoBrojevi = {5, 8, 17, 23, 27, 33, 36}, šta je rezultat poziva Arrays.binarySearch(lotoBrojevi, 17)?
- 0
  - 1
  - 1
  - 2
  - 2
32. Koja je od ovih deklaracija ispravna?
- char[][] z = {'a', 'b'};
  - char[2][2] z = {{'a', 'b'}, {'c', 'd'}};
  - char[2][] z = {{'a', 'b'}, {'c', 'd'}};
  - char[][] z = {{'a', 'b'}, {'c', 'd'}};
33. Ako je data deklaracija double[][] d = new double[4][5], koje su vrednosti dužina d.length i d[2].length?
- 4 i 4
  - 4 i 5
  - 5 i 4
  - 5 i 5
34. Analizirajte sledeći program:

```
public class Test {  
  
    public static void main(String[] args) {  
  
        boolean[][] x = new boolean[3][];  
  
        x[0] = new boolean[1];  
        x[1] = new boolean[2];  
        x[2] = new boolean[3];  
  
        System.out.println("x[2][2] je " + x[2][2]);  
    }  
}
```

- Program ima grešku, jer je new boolean[3][] nepravilno.
- Program ima grešku, jer će x[2][2] imati vrednost null.
- Program se normalno izvršava i na ekranu se prikazuje x[2][2] je null.
- Program se normalno izvršava i na ekranu se prikazuje x[2][2] je false.

## Programski zadaci

1. Napisati program kojim se prikazuje niz svih prostih brojeva manjih od dateg broja  $m$  koristeći postupak Eratostenovog sita: redom isključiti proizvode svih prostih brojeva manjih od  $\sqrt{m}$ , a oni brojevi koji preostanu su prosti. Granicu niza prostih brojeva  $m$  preuzeti iz komandnog reda.
2. Napisati program koji simulira „igru života”. Ova igra se sastoji od kolonije organizama koji žive u sopstvenim ćelijama u konfiguraciji od dvodimenzionalne matrice ćelija. Konfiguracija organizama se menja u diskretnim vremenskim trenucima po generacijama, pri čemu je svaka ćelija matrice prazna ili zauzeta živim organizmom. Nova generacija organizama u ćelijama nastaje na osnovu stare generacije organizama zavisno od sadržaja osam susednih ćelija svake pojedine ćelije. (Ćelije na obodu matrice se podrazumevaju da na odgovarajuće strani uvek imaju prazne susedne ćelije.) Pravila za formiranje nove generacije organizama su:
  1. Živi organizam u ćeliji preživljava u sledećoj generaciji ukoliko je broj njegovih suseda dva ili tri.
  2. Živi organizam u ćeliji umire u sledećoj generaciji ukoliko je broj njegovih suseda manji od dva (zbog usamljenosti) ili veći od tri (zbog prenaseljenosti).
  3. U praznoj ćeliji se rađa novi organizam ukoliko se u tačno tri njene susedne ćelije nalazi živi organizam.

Drugim rečima, za ćelije u svakoj generaciji pravila prelaza su: puna ćelija ostaje puna ako ima dve ili tri pune susedne ćelije; puna ćelija postaje prazna ako ima manje od dve ili više od tri pune susedne ćelije; prazna ćelija postaje puna ako ima tačno tri pune susedne ćelije, a u suprotnom ostaje prazna.

Igra života počinje od zadate početne konfiguracije koja se učitava na ulazu. Zatim se u diskretnim trenucima redom formiraju sledeće konfiguracije organizama *istovremenom* primenom gornjih pravila na sve ćelije prethodne konfiguracije (tj. nova generacija se formira isključivo na osnovu prethodne generacije).

3. Napisati klasu kojom se realizuje keš-memorija. Keš-memorija (engl. *cache*) je struktura podataka koja se sastoji od niza elemenata fiksne dužine. U niz se mogu samo dodavati novi elementi i pritom se novi element uvek dodaje na prvo mesto niza. Ali, ukoliko se novi element već nalazi u nizu, elementi ispred njega se pomeraju za jedno mesto udesno. A ukoliko se novi element ne nalazi u nizu, postojeći elementi se takođe pomeraju za jedno mesto udesno, ali se poslednji element odbacuje ako je niz već popunjen.
  
4. Iskoristiti prethodni zadatak i napisati program u kojem se otkrivaju često ponavljane reči u tekstu. Jedna reč u nekom tekstu se često ponavlja ukoliko se pojavljuje više od jedanput unutar bilo koje sekvence susednih reči od, recimo, 30 jedinstvenih reči. Minimalna funkcionalnost koju program mora imati je:
  - a) Program treba svaku reč da konvertuje u mala slova kako se ne bi razlikovale iste reči sa različitom veličinom slova.
  - b) Ukoliko je neka reč skoro ponovljena u ulaznom tekstu, program treba da prikaže tu reč i broj puta koliko je ta reč bila ponovljena.
  
5. Napisati program kojim se upravlja čekaonicom jednog doma zdravlja. U domu zdravlja radi više doktora čiji se podaci unose na početku programa. Svaki doktor ima svoju specijalnost koja je predstavljena slovnim kodom: „o” za lekara opšte prakse, „k” za kardiologa, „g” za ginekologa i tako dalje. Kada pacijent dođe u dom zdravlja, prijavljuje se na recepciju i medicinska sestra na osnovu simptoma bolesti odlučuje o specijalnosti doktora koji treba da primi pacijenta. Kada jedan doktor odgovarajuće specijalnosti postane slobodan, pacijent koji najduže čeka za tu specijalnost ide kod takvog doktora na pregled.  
U domu zdravlja radi više doktora iste specijalnosti i oni naizmenično primaju pacijente. Ako doktor završi pregled (i bude slobodan), a nema pacijenata koji čekaju za njegovu specijalnost, doktor se odmara dok ne bude potreban. Kada je jedan doktor neke specijalnosti potreban, pacijenta prima doktor te specijalnosti koji se najduže odmarao.  
Prema tome, pacijenti se primaju kod doktora na fer način (oni koji duže čekaju će pre biti primljeni) i doktori se odmaraju na fer način (oni koji su se duže odmarali će pre primiti nove pacijente). Ali, moguće

je da pacijenti čekaju iako ima slobodnih doktora, pod uslovom da nijedan od slobodnih doktora nije potrebne specijalnosti.

Pored opcija za prijavljivanje novog pacijenta i registrovanje pregleđanog pacijenta, program treba da sadrži još tri dodatne opcije: za prikazivanje pacijenata u čekaonici po redu po kome će biti primljeni kod doktora; za prikazivanje slobodnih doktora po redu po kome će primiti pacijente; i za prikazivanje aktuelnih pregleda, odnosno parova zauzetih doktora i njihovih trenutnih pacijenata.





# 8

## ***Nasleđivanje klasa***

### **Pitanja**

1. Objektno orijentisano programiranje i Java omogućavaju da se nove klase prave na osnovu postojećih klasa. Kako se naziva takva mogućnost?
  - enkapsulacija
  - nasleđivanje
  - polimorfizam
  - apstrakcija
2. Koje su od ovih rečenica tačne?
  - Jedna klasa u Javi može *direktno* proširivati više klasa.
  - Proširena klasa sadrži dodatna polja i metode u odnosu na svoju nasleđenu klasu.
  - „Klasa A nasleđuje klasu B” znači da je A potklasa od B.
  - Ako klasa A nasleđuje klasu B, tada objekti klase A sadrže sva polja i sve metode klase B.
  - Ako klasa A nasleđuje klasu B, tada se svaki objekat klase A podrazumeva da je i objekat klase B.
3. Pronađite sve greške u sledećem programu:

```
1  public class Test {  
2  
3      public static void main(String[] args) {  
4          Vozilo v = new Vozilo(8);  
5          v.vazi();  
6          v.brojVrata = 2;  
7          Vozilo bmw = new Auto(2, 4);  
8          bmw.vazi();  
9          Auto audi = new Auto(4);  
10         audi.vazi();  
11     }  
12 }  
13  
14 class Vozilo {  
15  
16     public int brojTočkova;  
17     public Vozilo(int t) {  
18         brojTočkova = t;  
19     }  
20     public void vazi() {  
21         System.out.println("Vožnja vozila");  
22     }  
23 }  
24  
25 class Auto extends Vozilo {  
26  
27     public int brojVrata;  
28     public Auto(int v, int t) {  
29         super(t);  
30         brojVrata = v;  
31     }  
32     public void vazi() {  
33         System.out.println("Vožnja auta");  
34     }  
35 }
```

- Greške su u redovima 7, 10 i 14.
- Greške su u redovima 7, 9 i 25.
- Greške su u redovima 6, 7 i 10.
- Greške su u redovima 6, 9 i 10.
- Greške su u redovima 6, 9 i 29.
- Greške su u redovima 9, 10 i 29.

4. Analizirajte sledeći program:

```
public class Test extends A {  
  
    public static void main(String[] args) {  
        Test t = new Test();  
        t.prikaži();  
    }  
}  
  
class A {  
    String s;  
  
    public A(String s) {  
        this.s = s;  
    }  
  
    public void prikaži() {  
        System.out.println(s);  
    }  
}
```

- Program ima grešku, jer klasa Test nema podrazumevani konstruktor Test().
- Program ima grešku, jer klasa Test ima implicitni podrazumevani konstruktor bez parametara Test(), ali nasleđena klasa A nema takav konstruktor. Program bi radio bez greške ukoliko bi se uklonio konstruktor u klasi A.
- Program ima grešku, ali bi radio bez greške ukoliko bi se klasi A eksplicitno dodao konstruktor bez parametara A().

### 5. Šta se prikazuje na ekranu kao rezultat izvršavanja ovog programa?

```
public class Test extends A {  
    public static void main(String[] args) {  
        Test t = new Test();  
    }  
}  
  
class A extends B {  
    public A() {  
        System.out.println(  
            "Pozvan podrazumevani konstruktor klase A");  
    }  
}
```

```
class B {  
    public B() {  
        System.out.println(  
            "Pozvan podrazumevani konstruktor klase B");  
    }  
}
```

- Ništa.
  - Pozvan podrazumevani konstruktor klase A
  - Pozvan podrazumevani konstruktor klase B
  - Pozvan podrazumevani konstruktor klase A i u drugom redu Pozvan podrazumevani konstruktor klase B
  - Pozvan podrazumevani konstruktor klase B i u drugom redu Pozvan podrazumevani konstruktor klase A
6. Koja od ovih rečenica o službenoj reči super nije tačna?
- Službena reč super može poslužiti za pozivanje konstruktora nasleđene klase.
  - Službena reč super može poslužiti za pozivanje zaklonjenog metoda nasleđene klase.
  - Službena reč super ne može poslužiti za pozivanje zaklonjenog polja nasleđene klase.
7. Analizirajte sledeći program:

```
public class Test {  
    public static void main(String[] args) {  
        B b = new B();  
        b.m(5);  
        System.out.println("b.i je " + b.i);  
    }  
}  
  
class A {  
    int i;  
  
    public void m(int i) {  
        this.i = i;  
    }  
}
```

```
class B extends A {  
    public void m(String s) {  
        System.out.println(s);  
    }  
}
```

- Program ima grešku, jer je metod `m()` nadjačan sa različitim potpisom u klasi B.
  - Program ima grešku, jer se `b.m(5)` ne može pozvati pošto je metod `m(int)` zaklonjen u klasi B.
  - Program ima grešku kod `b.i`, jer je polje i nepristupačno iz klase B.
  - Metod `m()` nije nadjačan u klasi B. Klasa B nasleđuje metod `m(int)` od klase A i u B se definiše preopterećen metod `m(String)`.
8. Koje su od ovih rečenica tačne?
- Metod se može preopteretiti u istoj klasi.
  - Metod se može nadjačati u istoj klasi.
  - Ako su metodi preopterećeni, njihovi potpisi moraju biti isti.
  - Ako jedan metod nadjačava drugi metod, njihovi potpisi moraju biti isti.
9. Koje su od ovih rečenica tačne?
- Jedna forma polimorfizma u Javi je princip podtipa po kojem promenljiva klasnog tipa može sadržati reference na objekte svog deklarisanog tipa i svakog njegovog podtipa.
  - Objekat tipa B se može preneti kao argument metodu na mesto paramatera tipa A ukoliko je B klasa naslednica od A.
  - Za razrešavanje poziva preopterećenih metoda se primenjuje statičko vezivanje u fazi prevođenja programa.
  - Za razrešavanje poziva nadjačanih metoda se primenjuje dinamičko vezivanje u fazi izvršavanja programa.
10. Analizirajte sledeći program:

```
public class Test {  
  
    public static void main(String[] args) {  
        Tim partizan = new Tim("Partizan", "Beograd");  
        Tim zvezda = new Tim("Crvena zvezda", "Beograd");  
        KošarkaškaUtakmica finale;  
        finale = new KošarkaškaUtakmica(partizan, zvezda);  
        finale.domaćinPoentirao(3);  
        finale.gostPoentirao(2);  
        finale.pobednikUtakmice();  
    }  
}  
  
class Tim {  
  
    private String ime;  
    private String mesto;  
  
    public Tim(String ime, String mesto) {  
        this.ime = ime;  
        this.mesto = mesto;  
    }  
    public String toString() {  
        return ime + " (" + mesto + ")";  
    }  
}  
  
class KošarkaškaUtakmica {  
  
    public Tim domaćin, gost;  
    public int brojPoenaDomaćina, brojPoenaGosta;  
  
    public KošarkaškaUtakmica(Tim d, Tim g) {  
        domaćin = d;  
        gost = g;  
    }  
  
    public void domaćinPoentirao(int brojPoena) {  
        brojPoenaDomaćina += brojPoena;  
    }  
  
    public void gostPoentirao(int brojPoena) {  
        brojPoenaGosta += brojPoena;  
    }  
  
    public void pobednikUtakmice() {  
        if (brojPoenaDomaćina > brojPoenaGosta)
```

```
        System.out.println("Pobednik je " + domaćin);
    else if (brojPoenaDomaćina < brojPoenaGosta)
        System.out.println("Pobednik je " + gost);
    else
        System.out.println("Nerešeno");
}
}
```

- Izvršavanjem programa se na ekranu ništa ne prikazuje.
- Izvršavanjem programa se na ekranu prikazuje tekst: Pobednik je Crvena zvezda.
- Izvršavanjem programa se na ekranu prikazuje tekst: Pobednik je Crvena zvezda (Beograd).
- Izvršavanjem programa se na ekranu prikazuje tekst: Pobednik je Partizan.
- Izvršavanjem programa se na ekranu prikazuje tekst: Pobednik je Partizan (Beograd).
- Izvršavanjem program se na ekranu prikazuje tekst: Nerešeno.

11. Analizirajte sledeći program:

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

        Object a = new A();
        Object o = new Object();
        System.out.println(a);
        System.out.println(o);
    }
}

class A {
    int x;

    public String toString() {
        return "x u A je " + x;
    }
}
```

- Program ima sintaksnu grešku, jer naredbu System.out.println(a) treba zameniti naredbom System.out.println(a.toString()).
- Program nema greške i poziva se metod toString() u klasi Object prilikom izvršavanja naredbe System.out.println(a).

- Program nema greške i poziva se metod `toString()` u klasi A prilikom izvršavanja naredbe `System.out.println(a)`.
  - Program nema greške i poziva se metod `toString()` u klasi Object prilikom izvršavanja naredbe `System.out.println(o)`.
12. Metod `equals()` za proveru jednakosti dva objekta je definisan u klasi Object. Ukoliko bi se taj metod nadjačao u klasi String, koje bi od ovih zaglavlja tog nadjačanog metoda bilo ispravno?
- `public boolean equals(String s)`
  - `public boolean equals(Object o)`
  - `public static boolean equals(Object o)`
  - `public boolean equals(String s1, String s2)`
13. Analizirajte sledeći program:
- ```
public class Test {  
    public static void main(String[] args) {  
        Object a1 = new A();  
        Object a2 = new A();  
        System.out.println(a1.equals(a2));  
    }  
}  
  
class A {  
    int x;  
  
    public boolean equals(Object o) {  
        A a = (A)o;  
        return this.x == a.x;  
    }  
}
```
- Program ima grešku, jer se izrazom `a1.equals(a2)` proverava jednakost objekata tipa različitog od Object.
  - Program se izvršava bez greške i prikazuje se true na ekranu.
  - Program se izvršava bez greške i prikazuje se false na ekranu.
14. Analizirajte sledeći program:

```
public class Test {  
    public static void main(String[] args) {  
        Object a1 = new A();  
        Object a2 = new A();  
        System.out.println(a1.equals(a2));  
    }  
}  
  
class A {  
    int x;  
  
    public boolean equals(A a) {  
        return this.x == a.x;  
    }  
}
```

- Program ima grešku, jer se izrazom a1.equals(a2) proverava jednakost objekata tipa različitog od Object.
- Program se izvršava bez greške i prikazuje se true na ekranu.
- Program se izvršava bez greške i prikazuje se false na ekranu.

15. Analizirajte sledeći program:

```
public class Test {  
    public static void main(String[] args) {  
        A a1 = new A();  
        A a2 = new A();  
        System.out.println(a1.equals(a2));  
    }  
}  
  
class A {  
    int x;  
  
    public boolean equals(A a) {  
        return this.x == a.x;  
    }  
}
```

- Program ima grešku, jer se izrazom a1.equals(a2) proverava jednakost objekata tipa različitog od Object.
- Program se izvršava bez greške i prikazuje se true na ekranu.
- Program se izvršava bez greške i prikazuje se false na ekranu.

16. Pronađite greške u sledećem programu:

```
public class Test {  
    public static void main(String[] args) {  
        m(new MasterStudent());  
        m(new Student());  
        m(new Osoba());  
        m(new Object());  
    }  
  
    public static void m(Student x) {  
        System.out.println(x.toString());  
    }  
}  
  
class MasterStudent extends Student {}  
  
class Student extends Osoba {  
    public String toString() {  
        return "Student";  
    }  
}  
  
class Osoba extends Object {  
    public String toString() {  
        return "Osoba";  
    }  
}
```

- Greška je u pozivu m(new MasterStudent()).
- Greška je u pozivu m(new Student()).
- Greška je u pozivu m(new Osoba()).
- Greška je u pozivu m(new Object()).

17. Analizirajte sledeći programski fragment:

```
class C1 {}  
class C2 extends C1 {}  
class C3 extends C2 {}  
class C4 extends C1 {}
```

```
C1 c1 = new C1();  
C2 c2 = new C2();  
C3 c3 = new C3();  
C4 c4 = new C4();
```

- Rezultat izraza c1 instanceof C1 je true.
- Rezultat izraza c2 instanceof C1 je true.
- Rezultat izraza c3 instanceof C1 je true.
- Rezultat izraza c4 instanceof C2 je true.

18. Analizirajte sledeći program:

```
public class Test {  
    public static void main(String[] args) {  
        String s = "Java";  
        Object o = s;  
        String t = (String)o;  
    }  
}
```

- Kada se promenljiva s dodeljuje promenljivoj o u naredbi Object o = s, konstruiše se novi objekat.
- Kada se konvertuje tip promenljiva o i njena vrednost dodeljuje promenljivoj t u naredbi String t = (String)o, konstruiše se novi objekat.
- Kada se konvertuje tip promenljiva o i njena vrednost dodeljuje promenljivoj t u naredbi String t = (String)o, sadržaj promenljive o se menja.
- Promenljive s, o i t ukazuju na isti objekat tipa String.

19. Ako je data deklaracija niza Object[] x, koje su od ovih naredbi ispravne?

- x = new char[100];
- x = new int[100];
- x = new double[100];
- x = new String[100];
- x = new java.util.Date[100];

20. Kojim se od ovih izraza ispravno konstruiše dinamički niz tipa ArrayList?

- new ArrayList[]
- new ArrayList[100]
- new ArrayList()

ArrayList()

21. Ako je x dinamički niz tipa `ArrayList<String>` čiji su elementi dva stringa [Java, C++], tim redom, kojom naredbom će se njegov sadržaj smanjiti na [Java]?

- `x.remove("C++")`
- `x.remove(0)`
- `x.remove(1)`
- `x.remove(2)`

22. Ako je x dinamički niz tipa `ArrayList<String>` čiji su elementi dva stringa [Java, C++], tim redom, šta će biti njegov novi sadržaj nakon izvršavanja naredbe `x.add("Pascal")`?

- [Java, Pascal]
- [Java, C++, Pascal]
- [Pascal, Java, C++]
- [Java, Pascal, C++]

23. Kojom se naredbom dobija prvi element niza x tipa `ArrayList`?

- `x.first()`
- `x.get(0)`
- `x.get(1)`
- `x.get()`

24. Kojom se naredbom dobija aktuelna dužina niza x tipa `ArrayList`?

- `x.getSize()`
- `x.getLength(0)`
- `x.length(1)`
- `x.size()`

25. Koja se od ovih klasa ne može nasleđivati?

- `class A { }`

- class A { private A(){ } }
- final class A { }
- class A { protected A(){ } }

## Programski zadaci

1. Napisati klasu PovezanaLista koja predstavlja jednostruko povezanu listu objekata. Jedan element liste treba da bude predstavljen posebnom klasom ElementListe, a klasa PovezanaLista treba da omogućava samo osnovne operacije za rad sa listom: dodavanja novog objekta na kraj liste i proveravanje da li se dati objekat nalazi u listi ili ne. (Prepostavite da objekti u elementima liste implementiraju metod equals() kojim se ispituje njihova jednakost.)
2. Koristeći klasu PovezanaLista napisati klasu PoliLinija kojom se poligonalna linija predstavlja u obliku liste njenih tačaka temena.
3. Proširiti klasu PovezanaLista iz 1. zadatka tako da se omogućava pristup jednom elementu povezane liste kojim se određuje tekuća pozicija u listi. Proširenoj klasi dodati i metode kojima se dodaje novi objekat u tekućoj poziciji i uklanja element na tekućoj pozici.

# 9

## ***Posebne klase i interfejsi***

### **Pitanja**

1. Koje su od ovih rečenica o nabrojivim tipovima tačne?
  - Definicija nabrojivog tipa može stajati unutar nekog metoda.
  - Nabrojni tip mora biti član neke klase.
  - Vrednosti nabrojivog tipa su kodirani običnim celim brojevima 0, 1, 2 i tako dalje.
  - Za vrednosti nabrojivog tipa koristi se konvencija o imenima za obične konstante.
  - Nabrojni tip je specijalna vrsta ugnježđene klase.
2. Ako u programu treba koristiti tri osnovne boje, koja od ovih deklaracija predstavlja verovatno najbolji pristup za takav problem?
  - public enum osnovnaBoja {crvena, žuta, plava}
  - public int[] osnovnaBoja = {1, 2, 3}
  - public enum OsnovnaBoja {CRVENA, ŽUTA, PLAVA}
  - public String[] osnovnaBoja = {"crvena", "žuta", "plava"}
  - public String boja1 = "crvena", boja2 = "žuta", boja3 = "plava"
3. Šta se prikazuje na ekranu kao rezultat izvršavanja ovog programa?

```
public class Test {  
  
    public enum Dan {  
        PONEDELJAK, UTORAK, SREDA, ČETVRTAK, PETAK, SUBOTA, NEDELJA};  
  
    public static void main(String[] args) {  
        Dan d = Dan.SREDA;  
        System.out.print(d + " je ");  
        System.out.println(d.ordinal() + ". dan u nedelji");  
    }  
}
```

- SREDA je 3. dan u nedelji
- Sreda je 3. dan u nedelji
- Dan.SREDA je 3. dan u nedelji
- SREDA je 2. dan u nedelji
- Sreda je 2. dan u nedelji
- Dan.SREDA je 2. dan u nedelji

4. Šta se prikazuje na ekranu kao rezultat izvršavanja ovog programa?

```
public class Test {  
  
    public enum SvetloSemafora {  
        CRVENO ("Stani"), ZELENO ("Kreni"), ŽUTO ("Pazi");  
        private String opis;  
        private SvetloSemafora(String opis) {  
            this.opis = opis;  
        }  
        public String getOpis() {  
            return opis;  
        }  
    }  
  
    public static void main(String[] args) {  
        SvetloSemafora svetlo = SvetloSemafora.ZELENO;  
        System.out.println(svetlo.getOpis());  
    }  
}
```

- SvetloSemafora.ZELENO
- ZELENO
- zeleno

- Kreni
- Pazi

5. Koja je od ovih definicija apstraktne klase A ispravna?
- class A { abstract void aMetod() { } }
  - class A { abstract void aMetod(); }
  - abstract class A { abstract void aMetod(); }
  - public class abstract A { abstract void aMetod(); }
6. U kojem je od ovih slučajeva ispravno definisan apstraktni metod aMetod() u nekoj apstraktnoj klasi?
- public abstract aMetod();
  - public abstract void aMetod();
  - public void abstract aMetod();
  - public void aMetod() { }
  - public abstract void aMetod() { }
7. Koje su od ovih rečenica o apstraktnim klasama tačne?
- Moguće je konstruisati objekte apstraktne klase.
  - Apstraktna klasa može biti tip neke objektne promenljive.
  - Klasa koja nasleđuje apstraktnu klasu može biti apstraktna.
  - Klasa koja nasleđuje konkretnu klasu može biti apstraktna.
8. Koje su od ovih rečenica o apstraktnim klasama tačne?
- Apstraktne klase ne mogu imati konstruktore.
  - Klasa koja sadrži apstraktni metod mora biti apstraktna.
  - Moguće je definisati apstraktnu klasu koja ne sadrži nijedan apstraktni metod.
  - Apstraktne klase mogu imati konkretna polja i metode.
9. Ako je A apstraktna klasa, B je njena konkretna klasa naslednica i obe klase imaju podrazumevane konstruktore bez parametara, koje su od ovih deklaracija ispravne?

- A a = new A();
- A a = new B();
- B b = new A();
- B b = new B();

10. Koja je od ovih definicija interfejsa A ispravna?

- interface A { void print() { } }
- abstract interface A { print(); }
- abstract interface A { abstract void print() { } }
- interface A { void print(); }

11. Šta se prikazuje na ekranu kao rezultat izvršavanja ovog programa?

```
public class Test {

    public static void main(String[] args) {
        B b = new B();
        if (b instanceof A)
            System.out.println("b je instanca A");
        if (b instanceof C)
            System.out.println("b je instanca C");
    }
}

interface A {

    class B extends D implements A {
    }

    class C {
    }

    class D extends C {
    }
}
```

- Ništa.
- b je instanca A
- b je instanca C
- b je instanca A i u drugom redu b je instanca C

12. Ako je A interfejs i B je konkretna klasa sa podrazumevanim konstruktorom koja implementira A, koje su od ovih deklaracija ispravne?

- A a = new A();
- A a = new B();
- B b = new A();
- B b = new B();

13. Analizirajte sledeći program:

```
public class Test {  
    public static void main(String[] args) {  
        Voćka[] voće = {new Voćka(2), new Voćka(3), new Voćka(1)};  
        java.util.Arrays.sort(voće);  
    }  
}  
  
class Voćka {  
    private double težina;  
  
    public Voćka(double težina) {  
        this.težina = težina;  
    }  
}
```

- Program ima grešku, jer klasa Voćka nema podrazumevani konstruktor.
- Program ima grešku kod dodeljivanja početnih vrednosti nizu voće.
- Program ima grešku kod sortiranja niza voće, jer klasa Voćka ne implementira interfejs Comparable i zato objekti klase Voćka nisu uporedivi.
- Program nema greške, ali se ništa ne prikazuje na ekranu.

14. Analizirajte sledeću klasu Radnik:

```
public class Radnik implements Comparable {  
  
    double plata;  
  
    public int compareTo(Object o) {  
  
        Radnik r = (Radnik) o;
```

```
        if (this.plata < r.plata) return -1;
        if (this.plata > r.plata) return +1;
        return 0;
    }

    public static Radnik max(Radnik r1, Radnik r2) {
        if (r1.compareTo(r2) >= 0)
            return r1;
        else
            return r2;
    }
}
```

- Klasa Radnik ima grešku, jer nema nijedan konstruktor.
  - Klasa Radnik ima grešku, jer je metod max() definisan da bude static.
  - Klasa Radnik ima grešku, jer tip vraćene vrednosti metoda max() treba da bude int.
  - Klasa Radnik nema grešaka.
15. Koje su od ovih rečenica o ugnježđenim klasama tačne?
- Ugnježđene klase se mogu definisati kao članovi druge klase ili lokalno unutar metoda druge klase.
  - Statičke i objektne ugnježđene klase se, kao i obične klase, mogu definisati samo sa modifikatorom public ili bez njega.
  - Statička ugnježđena klasa ima potpun pristup statickim članovima obuhvatajuće klase, čak i ukoliko su oni deklarisani kao privatni.
  - Objektna ugnježđena klasa je analogna objektnom polju ili objektnom metodu. Zbog toga je objektna ugnježđena klasa vezana za objekat obuhvatajuće klase.

16. Ako je A statička ili objektna klasa ugnježđena unutar klase Test, koje ime će imati datoteka u kojoj se nalazi bajtkod klase A?

- A.class
- Test\$A.class
- A\$Test.class
- Test&A.class

17. Koja je od ovih rečenica o objektnim ugnježđenim klasama tačna?
- Objektne ugnježđene klase moraju implementirati neki interfejs.
  - Objektne ugnježđene klase se mogu instancirati samo unutar obuhvatajuće klase.
  - Objektne ugnježđene klase imaju pristup svim objektnim članovima obuhvatajuće klase, čak i ukoliko su oni deklarisani kao privatni.
  - Objektne ugnježđene klase se definišu kao članovi obuhvatajuće klase sa modifikatorom object.
18. Koje su od ovih rečenica o anonimnim klasama tačne?
- Anonimne klase su lokalne klase bez imena.
  - Anonimne klase uvek nasleđuju neku klasu ili implementiraju neki interfejs, ali ne sadrže eksplisitne reči extends ili implements.
  - Anonimne klase moraju implementirati sve apstraktne metode u nasleđenoj klasi ili interfejsu.
  - Anonimne klase uvek koriste podrazumevani konstruktor bez argumenata nasleđene klase radi konstruisanja svojih instanci.
19. Ako je jedna anonimna klasa ugnježđena unutar klase Test, koje će imati datoteka u kojoj se nalazi bajtkod te anonimne klase?
- A.class
  - Test\$A.class
  - A\$Test.class
  - TestA.class
  - Test\$1.class

## Programski zadaci

1. Napisati klasu Karta koja predstavlja kartu za igranje, a za predstavljanje boje i vrednosti karte koristiti nabrojive tipove. Radi testiranja napisati metod main() u kome se prikazuje ceo špil karata.
2. Napisati interfejs Konverzije i konkretnu klasu Konvertor koja ga implementira radi međusobnog pretvaranja veličina izraženih u anglosaksonskim i internacionalnim jedinicama (inči i centimetri, funte i kilogrami, ...).
3. Napisati interfejs PrenosnikUtakmice koji sadrži metode koji su bitni za prenos toka košarkaške utakmice sa gledišta onoga ko prenosi utakmicu. Napisati klasu Utakmica koja predstavlja košarkašku utakmicu sa listom prenosnika koji je prenose. Napisati i klase Zapisnik, Semafor i MobilniTelefon koje implementiraju interfejs PrenosnikUtakmice radi prenosa toka košarkaške utakmice. Na kraju, radi testiranja napisati klasu koja simulira jednu košarkašku utakmicu.
4. Napisati apstraktну klasu GeometrijskiOblik koja predstavlja geometrijski oblik u ravni (krug, pravougaonik, ...). Objekti ove klase treba da poseduju referentnu tačku pomoću koje se mogu translirati u ravni. Pored apstraktnog metoda za transliranje geometrijskog oblika, klasa treba da ima i apstraktne metode za obim i površinu geometrijskog oblika. Konkretizovati apstraktну klasu za geometrijske oblike kruga i pravougaonika i napisati posebnu klasu radi testiranja.
5. Ponoviti 4. zadatak uz dodatnu funkcionalnost da se geometrijski oblici mogu upoređivati na osnovu njihove površine. Radi testiranja, u posebnoj klasi na slučajan način generisati niz od 20 geometrijskih oblika i prikazati ih u rastućem redosledu.
6. Napisati apstraktnu klasu Uredaj koja predstavlja uređaj za kućnu zabavu, zatim interfejs DaljinskiUpravljač koji predstavlja daljinski upravljač za njihovo upravljanje, kao i dve klase TV i DVDPlayer koje

proširuju apstraktnu klasu Uredaj i interpretiraju interfejs Daljinski-Upravljač. Napisati i posebnu klasu Kućna Zabava radi testiranja metoda svih klasa i polimorfizma objekata.

7. Ponovo napisati klasu PovezanaLista iz 1. zadatka u glavi 8, ali na drugi način tako da klasa ElementListe bude ugnježđena.
8. Ponovo napisati program koji simulira „igru života” iz 2. zadatka u glavi 7, ali na drugi način tako da se koristi ugnježđena klasa za ćelije kolonije organizama i nabrojivi tip koji opisuje sadržaj ćelija (zauzeto/prazno).
9. Napisati klasu Sortirač u kojoj je definisan statički metod za sortiranje niza objekata u rastućem redosledu. Relativan redosled objekata određuje se jednim parametrom tog metoda tipa Upoređivač. Pritom, Upoređivač je ugnježđeni interfejs u kojem je definisan metod za poređenje dva objekta. Radi testiranja, u posebnoj klasi na slučajan način generisati niz od 10 kompleksnih brojeva (iskoristiti rešenje 3. zadatka iz glave 6) i sortirati ga na dva načina: prema modulima kompleksnih brojeva, kao i prema njihovim realnim a zatim imaginarnim delovima (ako su realni delovi jednaki).



# 10

## *Grafičko programiranje*

### Pitanja

1. Grafički programi u Javi su \_\_\_\_\_.
  - vođeni događajima
  - sporiji od konzolnih programa
  - zasnovani na grafičkom korisničkom interfejsu
  - objektno orijentisani
2. Koje se od ovih klasa nalaze u paketu `java.awt`?
  - Color
  - Font
  - Component
  - JFrame
  - JComponent
3. Koje se od ovih klasa nalaze u paketu `java.swing`?
  - Color
  - Font
  - Component

- JFrame
- JComponent

4. Analizirajte sledeći program:

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

public class Test {
    public static void main(String[] args) {
        JButton dugmeOK = new JButton("OK");
        JFrame okvir = new JFrame("Moj okvir");
        okvir.add(dugmeOK);
        okvir.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
        okvir.setVisible(true);
    }
}
```

- Program ima grešku, jer dugmeOK treba da bude tipa Button.
- Program ima grešku, jer se dugmeOK ne može direktno dodati okviru tipa JFrame.
- Program ima grešku, jer se naredbom new JFrame("Moj okvir") ne može konstruisati okvir.
- Program nema grešaka i normalno se izvršava.

5. Analizirajte sledeći program:

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

public class Test {
    public static void main(String[] args) {
        JFrame okvir = new JFrame("Moj okvir");
        okvir.add(new JButton("OK"));
        okvir.add(new JButton("Cancel"));
        okvir.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
        okvir.setSize(200, 200);
        okvir.setVisible(true);
    }
}
```

- Prikazuje se samo dugme OK.

- Prikazuje se samo dugme Cancel.
- Prikazuju se oba dugmeta OK i Cancel, pri čemu se dugme OK nalazi levo od dugmeta Cancel.
- Prikazuju se oba dugmeta OK i Cancel, pri čemu se dugme OK nalazi desno od dugmeta Cancel.

6. Koliko se okvira prikazuje ovim programom?

```
import javax.swing.*;  
  
public class Test {  
    public static void main(String[] args) {  
        JFrame okvir1 = new JFrame();  
        JFrame okvir2 = okvir1;  
        JFrame okvir3 = okvir2;  
        okvir1.setVisible(true);  
        okvir2.setVisible(true);  
        okvir3.setVisible(true);  
    }  
}
```

- 1
- 2
- 3
- 0

7. Koliko se okvira prikazuje ovim programom?

```
import javax.swing.*;  
  
public class Test extends JFrame {  
    public static void main(String[] args) {  
        JFrame okvir1 = new Test();  
        JFrame okvir2 = new Test();  
        JFrame okvir3 = new Test();  
        okvir1.setVisible(true);  
        okvir2.setVisible(true);  
        okvir3.setVisible(true);  
    }  
}
```

- 1

- 2
- 3
- 0

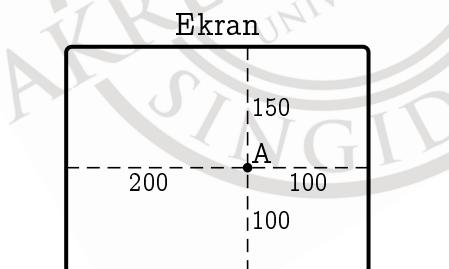
8. Kako se naziva najmanja tačka za crtanje na ekranu?

- rezolucija
- grid
- piksel
- dot

9. Koja je od ovih rečenica o koordinatnom početku ekrana tačna?

- Koordinatni početak (0, 0) se nalazi u donjem levom uglu ekrana.
- Koordinatni početak (0, 0) se nalazi u gornjem levom uglu ekrana.
- Koordinatni početak (0, 0) se nalazi u centru ekrana.
- Koordinatni početak (0, 0) se može programski promeniti.

10. Koje koordinate u Javi ima tačke A na ovoj slici?



- (200, 150)
- (200, 100)
- (100, 100)
- (100, 150)

11. Kako će izgledati okvir posle izvršavanja ove naredbe?

```
okvir.setBounds(100, 150, 300, 200);
```

- Gornji levi ugao okvira se nalazi u tački sa koordinatama (100, 150).  
 Gornji levi ugao okvira se nalazi u tački sa koordinatama (300, 200).  
 Širina okvira je 300 piksela i visina okvira je 200 piksela.  
 Širina okvira je 100 piksela i visina okvira je 150 piksela.
12. Koje su *relativne* koordinate gornjeg levog ugla svake grafičke komponente?
- (0, 0)  
 (10, 10)  
 (100, 100)  
 neodređene
13. Koji je podrazumevani način razmeštanja komponenti u panelu?
- FlowLayout.  
 GridLayout.  
 BorderLayout.  
 Nijedan.
14. Koji postupak razmeštanja komponenti unutar panela treba primeniti ukoliko dugme koje se nalazi u panelu ne treba da menja veličinu usled promene veličine panela?
- FlowLayout.  
 GridLayout.  
 BorderLayout, ali dugme treba dodati centralnom polju panela.  
 BorderLayout, ali dugme treba dodati istočnom ili zapadnom polju panela.  
 BorderLayout, ali dugme treba dodati severnom ili južnom polju panela.
15. Kojim se od ovih naredbi postiže da razmeštanje komponenti unutar panela p tipa JPanel bude izvedeno postupkom BorderLayout?
- p.setLayout(new BorderLayout());  
 p.setLayout(BorderLayout());

- `p.setLayout(new BorderLayout(5, 10));`
  - `p.setLayout(new BorderLayout(BorderLayout.CENTER));`
  - `p.setLayout(BorderLayout(BorderLayout.CENTER));`
16. Kojom se od ovih naredbi ispravno dodaje komponenta k panelu p tipa JPanel?
- `k.add(p);`
  - `p.add(k);`
  - `p.insert(k);`
  - `p.append(k);`
  - `k.addContentPane(p);`
17. Koji metod treba nadjačati radi crtanja u svakoj Swing komponenti?
- `repaint()`
  - `update()`
  - `paintComponent()`
  - `init()`
18. Koje su od ovih rečenica u vezi sa crtanjem u Javi tačne?
- Objekat tipa Graphics se može konstruisati pomoću `new Graphics()`.
  - Svaki put kada se komponenta prikaže na ekranu, automatski se konstruiše njen objekat tipa Graphics.
  - Metod `paintComponent()` ne treba nikad pozivati direktno, jer se on automatski poziva od strane JVM.
  - Pozivanje metoda `paintComponent()` od strane JVM se može implicitno zahtevati metodom `repaint()`.
19. Za proizvoljno crtanje u Javi, najbolje je definisati posebnu klasu koja proširuje klasu \_\_\_\_\_ i nadjačati njen metod `paintComponent()`.
- JLabel
  - JButton
  - JFrame
  - JComponent

20. Analizirajte sledeći program:

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

public class Test extends JFrame {

    public Test() {
        add(new MojaKomponenta("Zdravo narode!"));
    }

    public static void main(String[] args) {
        JFrame okvir = new JFrame();
        okvir.setSize(300, 300);
        okvir.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
        okvir.setVisible(true);
    }
}

class MojaKomponenta extends JComponent {

    private String poruka;

    public MojaKomponenta(String poruka) {
        this.poruka = poruka;
    }

    public void paintComponent(Graphics g) {
        g.drawString(poruka, 20, 20);
    }
}
```

- Program ima grašku, jer se u konstruktoru klase Test ne sme dodavati komponenta.
- Program se normalno izvršava i prikazuje poruku Zdravo narode! u okviru.
- Program se normalno izvršava, ali se ništa ne prikazuje u okviru.
- Program bi prikazao poruku Zdravo narode! u okviru ukoliko bi se u glavnom metodu zamenio izraz new JFrame() sa new Test().
- Program bi prikazao poruku Zdravo narode! u okviru ukoliko bi se u glavnom metodu zamenio izraz new JFrame() sa new Test("Zdravo narode!").

21. Analizirajte sledeći program:

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

public class Test extends JFrame {

    public Test() {
        add(new MojaKomponenta());
    }

    public static void main(String[] args) {
        JFrame okvir = new Test();
        okvir.setSize(300, 300);
        okvir.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
        okvir.setVisible(true);
    }
}

class MojaKomponenta extends JComponent {

    String poruka;

    public void setPoruka(String poruka) {
        this.poruka = poruka;
    }

    public void paintComponent(Graphics g) {
        g.drawString(poruka, 20, 20);
    }
}
```

- Program ima grašku, jer se u konstruktoru klase Test ne sme dodavati komponenta.
- Program ima grašku, jer poruka ima vrednost null kada se izvršava naredba `g.drawString(poruka, 20, 20)` u metodu `paintComponent()`.
- Program se normalno izvršava, ali se ništa ne prikazuje u okviru.
- Program bi prikazao poruku Zdravo narode! u okviru ukoliko bi se u glavnom metodu izraz `new Test()` zamenio sa `new Test("Zdravo narode!")`.
- Program bi prikazao poruku Zdravo narode! u okviru ukoliko bi se izraz `new MojaKomponenta()` u konstruktoru klase Test zamenio sa:  
`(new MojaKomponenta()).setPoruka("Zdravo narode!")`.

22. Ako je dat grafički kontekst `g` tipa `Graphics` neke komponente, kojom

- se od ovih naredbi crta pravougaonik čija su širina 20 i visina 50 i čiji se gornji levi ugao nalazi u tački (20, 20) te komponente?
- g.drawRect(20, 50, 20, 20);
  - g.drawRectFill(20, 20, 20, 50);
  - g.drawRect(20, 20, 20, 50);
  - g.drawRectFill(20, 50, 20, 20);
23. Ako je dat grafički kontekst g2 tipa Graphics2D neke komponente, kojim se od ovih naredbi crta pravougaonik čija su širina 20 i visina 50 i čiji se gornji levi ugao nalazi u tački (20, 20) te komponente?
- g2.draw(Rectangle2D(20, 20, 20, 50));
  - g2.draw(Rectangle2D.Double(20, 20, 20, 50));
  - g2.draw(new Rectangle2D.Double(20, 20, 20, 50));
  - g2.draw(new Rectangle2D.Float(20, 20, 20, 50));
  - g2.draw(new Rectangle2D.Double(20.0, 20.0, 20.0, 50.0));
  - g2.draw(new Rectangle2D.Float(20.0, 20.0, 20.0, 50.0));
  - g2.draw(new Rectangle2D.Float(20.0F, 20.0F, 20.0F, 50.0F));
  - g2.draw(new Rectangle2D(20, 20, 20, 50));
24. Ako je dat grafički kontekst g2 tipa Graphics2D neke komponente, kojim se od ovih naredbi konstruiše tačka t koja se nalazi u donjem desnom uglu te komponente?
- Point2D t = new Point2D(getWidth(), getHeight());
  - Point2D t = new Point2D.Double(getWidth(), getHeight());
  - Point2D.Double t = new Point2D.Double(getWidth(), getHeight());
  - Point2D t = new Point2D.Double(0, 0);
  - Point2D t = new Point2D.Double(0, getHeight());
  - Point2D t = new Point2D.Double(getWidth(), 0);
  - Point2D t = new Point2D.Double(getHeight(), getWidth());
25. Kojim se od ovih izraza ispravno konstruiše objekat boje u Javi?
- new Color(0, 0, 0)
  - new Color(0, 266, 0)

- new Color(255, 255, 255)
- new Color(1, 2, 3)

26. Kojim se od ovih naredbi ispravno određuje boja pozadine komponente k?

- k.setBackground(Color.PINK);
- k.setBackground(new Color(0, 128, 128));
- k.setBackground(Color(0, 128, 128));
- setBackground(Color.YELLOW)
- k.setForeground(Color.RED);

27. Kojim se od ovih izraza ispravno konstruiše objekat fonta u Javi?

- new Font("SansSerif", Font.BOLD, 36)
- new Font("SansSerif", Font.CAPS, 20)
- new Font("SansSerif", Font.BOLD, 10.5)
- new Font("Serif", Font.BOLD + Font.ITALIC, 12)
- new Font("Dialog", Font.PLAIN, 10)
- new Font("Cyrillic", Font.PLAIN, 12)
- new Font(Serif, Font.PLAIN, 12)

28. Kojeg tipa se proizvodi događaj kada se tasterom miša pritisne na neko dugme?

- ItemEvent
- MouseEvent
- MouseMotionEvent
- ActionEvent
- WindowEvent

29. Kojeg tipa se proizvodi događaj kada se pritisne jedan od tastera miša?

- ItemEvent
- MouseEvent
- MouseMotionEvent

- ActionEvent
- WindowEvent

30. Koje su od ovih rečenica o događajima tačne?

- Komponenta u kojoj se desio neki događaj se naziva izvor događaja.
- Ako komponenta može generisati neki događaj, svaka naslednica te komponente može generisati isti tip događaja.
- Sve klase događaja su naslednice klase EventObject.
- Klase događaja i interfejsi rukovalaca događaja se nalaze u paketu java.events.
- Ime klase događaja u Javi je standardnog oblika <Ime>Event, a interfejsa rukovaoca odgovarajućeg događaja je oblika <Ime>Listener.

31. Svaki objekat događaja ima metod \_\_\_\_\_.

- getSource()
- getActionCommand()
- getTimeStamp()
- getWhen()

32. Odgovarajući metod za obradu događaja (npr. actionPerformed()) na-  
lazi se u \_\_\_\_\_.

- objektu izvora događaja
- objektu rukovaoca događaja
- objektima izvora i rukovaoca događaja
- klasi Object
- klasi EventObject

33. Koje su od ovih rečenica o rukovaocima događaja tačne?

- Svaka klasa događaja u Javi ima odgovarajući interfejs rukovaoca događaja.
- Klasa rukovaoca događaja mora implementirati interfejs rukovaoca odgovarajućeg događaja.

- Jednom izvoru događaja može biti pridruženo više rukovaoca događaja.
  - Jedan rukovalac događaja može obrađivati događaje iz više izvora.
  - Rukovalac događaja se automatski pridružuje izvoru događaja prema tipu događaja koje obrađuje.
34. Koja od ovih naredbi objektu dugme pridružuje objekat rukovaoca akcijskog događaja rukovalacDugmeta?
- dugme.addListener(rukovalacDugmeta);
  - dugme.addActionListener(rukovalacDugmeta);
  - dugme.addActionEventListener(rukovalacDugmeta);
  - dugme.addEventlistener(rukovalacDugmeta);
35. Koji interfejs treba implementirati radi obrade akcijskog događaja pritiska tasterom miša na dugme?
- ButtonListener
  - ButtonPressedListener
  - MouseListener
  - ActionListener
36. Rukovalac akcijskog događaja mora biti objekat tipa \_\_\_\_\_. .
- ActionEvent
  - ActionListener
  - EventObject
  - WindowListener
37. Analizirajte sledeći program:

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

public class Test extends JFrame {

    public Test() {
```

```
    JButton dugmeOK = new JButton("OK");
    add(dugmeOK);
    dugmeOK.addActionListener(new ActionListener() {
        public void actionPerformed(ActionEvent e) {
            System.out.println("Dugme OK je pritisnuto");
        }
    });
}

public static void main(String[] args) {
    JFrame okvir = new Test();
    okvir.setSize(300, 300);
    okvir.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
    okvir.setVisible(true);
}
```

- Program ima logičku grešku, jer nijedan rukovalac akcijskog događaja dugmeta nije pridružen dugmetu OK.
- Program se normalno izvršava i svaki put kada se pritisne na dugme OK u okviru prikazuje se poruka Dugme OK je pritisnuto.
- Program se normalno izvršava, ali se ništa ne prikazuje u okviru.
- Program se normalno izvršava prikazujući dugme OK u okviru, ali ne i poruku Dugme OK je pritisnuto kada se pritisne na dugme OK.

38. Analizirajte sledeći program:

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

public class Test extends JFrame implements ActionListener {

    public Test() {
        JButton dugmeOK = new JButton("OK");
        add(dugmeOK);
    }

    public void actionPerformed(ActionEvent e) {
        System.out.println("Dugme OK je pritisnuto");
    }

    public static void main(String[] args) {
        JFrame okvir = new Test();
```

```
okvir.setSize(300, 300);
okvir.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
okvir.setVisible(true);
}
}
```

- Deklaracija `import java.awt.event.*` je suvišna, jer je njen efekat obuhvaćen deklaracijom `import java.awt.*`.
- Program ima grešku, jer klasa `Test` ne može istovremeno da proširuje klasu `JFrame` i implementira interfejs `ActionListener`.
- Program se normalno izvršava i svaki put kada se pritisne na dugme `OK` u okviru prikazuje se poruka Dugme `OK` je pritisnuto.
- Program se normalno izvršava prikazujući dugme `OK` u okviru, ali ne i poruku Dugme `OK` je pritisnuto kada se pritisne na dugme `OK`.

39. Analizirajte sledeći program:

```
1 import java.awt.*;
2 import java.awt.event.*;
3 import javax.swing.*;
4
5 public class Test extends JFrame {
6
7     public Test() {
8         JButton dugmeOK = new JButton("OK");
9         JButton dugmeNOK = new JButton("Nije OK");
10        add(dugmeOK);
11        add(dugmeNOK);
12        dugmeOK.addActionListener(this);
13        dugmeNOK.addActionListener(this);
14    }
15
16    public void actionPerformed(ActionEvent e) {
17        System.out.println("Jedno od dva dugmeta je pritisnuto");
18    }
19
20    public static void main(String[] args) {
21        JFrame okvir = new Test();
22        okvir.setSize(300, 300);
23        okvir.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
24        okvir.setVisible(true);
25    }
26 }
```

- Program ima grešku u redovima 12 i 13, jer dve različite komponente ne mogu imati isti rukovalac njihovim događajima.
- Program ima grešku u redovima 12 i 13, jer klasa Test ne implementira interfejs ActionListener.
- Program ima grešku u redu 16, jer metod actionPerformed() ima pogrešan potpis.
- Program ima grešku u redu 21, jer se objekat tipa Test dobijen izrazom new Test() dodeljuje promenljivoj okvir tipa JFrame.
- Program se normalno izvršava prikazujući dva dugmeta u okviru, ali ne i poruku Jedno od dva dugmeta je pritisnuto kada se pritisne na bilo koje od ova dva dugmeta u okviru.

40. Analizirajte sledeći program:

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

public class Test extends OkvirRukovalac {

    JButton dugmeOK = new JButton("OK");

    public Test() {
        add(dugmeOK);
        dugmeOK.addActionListener(this);
    }

    public void actionPerformed(ActionEvent e) {
        super.actionPerformed(e);
        if (e.getSource() == dugmeOK)
            System.out.println("Pritisnuto dugme OK");
    }

    public static void main(String[] args) {
        OkvirRukovalac okvir = new Test();
        okvir.setSize(200, 100);
        okvir.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
        okvir.setVisible(true);
    }
}

class OkvirRukovalac extends JFrame implements ActionListener {
```

```
 JButton dugmeNOK = new JButton("Nije OK");

public OkvirRukovalac() {
    setLayout(new FlowLayout());
    add(dugmeNOK);
    dugmeNOK.addActionListener(this);
}

public void actionPerformed(ActionEvent e) {
    if (e.getSource() == dugmeNOK)
        System.out.println("Pritisnuto dugme NOK");
}
```

- Program u okviru prikazuje dugme NOK levo od dugmeta OK.
  - Program prikazuje poruku Pritisnuto dugme OK kada se u okviru tasterom miša pritisne dugme OK.
  - Program prikazuje poruku Pritisnuto dugme NOK kada se u okviru tasterom miša pritisne dugme NOK.
  - Ukoliko se izostavi prva naredba super.actionPerformed(e) u metodu actionPerformed() klase Test, ne prikazuje se poruka Pritisnuto dugme NOK kada se u okviru tasterom miša pritisne dugme NOK.
41. Koja od ovih reči *nije* ime adapterske klase?
- ActionAdapter
  - MouseAdapter
  - WindowAdapter

## Programski zadaci

1. Napisati program kojim se prikazuje slika kamiona.
2. Napisati program kojim se prikazuje slika šahovske table.
3. Napisati program kojim se prikazuje slika akvarijuma sa ribicama. Ribice treba da imaju slučajnu boju i da budu okrenute levo ili desno na slučajan način.
4. Napisati program kojim se simulira jednostavan kalkulator sa operacijama sabiranja, oduzimanja, množenja i deljenja.
5. Napisati program kojim se prikazuje digitalni sat i tačno vreme na njemu.
6. Napisati program kojim se predskazuje sudbina tako što se prikazuje jedna slučajno odabrana iz niza sADBINA.
7. Napisati program kojim se prikazuje kvadrat čija se boja može menjati i koji se može pomerati tasterima na tastaturi.
8. Napisati program kojim se simulira igranje na tabli sa kliznim pločicama. Pri tome, igrač svoj sledeći potez zadaje preko tastature, a efekat njegovog poteza se prikazuje grafički pomeranjem željenog broja na prazno mesto.
9. Napisati grafički program kojim se simulira igranje na tabli sa kliznim pločicama. Igrač svoj sledeći potez zadaje pritiskom tastera miša na pločicu koju želi da pomeri i ta pločica se zatim pomera na prazno mesto.
10. Napisati program kojim se realizuje jednostavan grafički editor za uređivanje teksta.

11. Napisati program kojim se realizuje animacija većeg broja loptiča-skočica koje se na slučajan način kreću i odbijaju unutar jednog pravougaonika. Pored toga, ukoliko se tasterom miša pritisne u tom pravougaoniku, sve loptice treba da krenu prema mestu pritiska.
12. Napisati program kojim se prikazuje analogni sat i tačno vreme na njemu.



**Deo II**

**Rešenja**





*Glava*

# 1

## ***Uvod u Java programiranje***

### Odgovori na pitanja

1.  Softver  
 Hardver  
 Operativni sistem  
 Windows
  
2.  Softver  
 Hardver  
 Operativni sistem  
 Windows
  
3.  Hardver  
 Procesor (CPU)  
 Memorija  
 Disk
  
4.  naredbi  
 podataka  
 naredbi i podataka

- teksta
5. ○ 4  
☒ 8  
○ 16  
○ 32
6. ☒ mašinskom jeziku  
○ engleskom jeziku  
○ prirodnom jeziku  
○ jeziku visokog nivoa
7. ○ Operativni sistemi  
○ Procesori  
○ Ljudi  
☒ Prevodioci (kompajlери)
8. ☒ Java bajtkod  
○ Java objektni kod  
○ Java aplet  
○ Java aplikacija
9. ○ Test.txt  
○ Test.class  
☒ Test.java  
○ Main.java  
○ Main.txt
10. ○ .java  
○ .obj  
☒ .class  
○ .exe

11.  javac Test.java  
 compile Test.java  
 prevedi Test.java  
 javap Test.java
  
12.  javac Test  
 izvrsti Test  
 java Test  
 java Test.class
  
13.  NetBeans  
 DOS  
 Windows  
 DrJava  
 Linux



# 2

## ***Uvod u programski jezik Java***

### **Odgovori na pitanja**

1.  promenljivih  
 procedura  
 programa  
 objekata
  
2.  promenljive  
 procedure  
 klase  
 mašine
  
3.  obeležja (atributi) i mogućnosti (ponašanje)  
 sastavni delovi i izgled  
 broj i oblik  
 identifikacija i način upotrebe
  
4.  zameniti  
 proširiti  
 popraviti

- ukloniti
5.  nazivima  
 paketima  
 modulima  
 folderima
6.  import  
 export  
 module  
 package
7.  Po konvenciji, u Javi se imena paketa pišu svim malim slovima.  
 Deklaracija package nije obavezna.  
 Deklaracija import nije obavezna.  
 Klase u paketu java.lang se automatski dodaju u program.
8.  `/**` tekst komentara `*/`  
 `//` tekst komentara  
 `--` tekst komentara  
 `/*` tekst komentara `*/`  
 `**` tekst komentara `**`
9.  program se neće prevesti zato što je napisan lošim stilom  
 program će se brže izvršavati zato što je napisan dobrim stilom  
 program će biti čitljiviji zato što je napisan dobrim stilom  
 program će imati manji broj grešaka zato što je napisan dobrim stilom  
 program će se lakše modifikovati zato što je napisan dobrim stilom
10.  tačkom `(.)`  
 tačkom-zapetom `(;)`  
 zapetom `(,)`  
 zvezdicom `(*)`

# 3

## ***Osnovni elementi jezika Java***

### **Odgovori na pitanja**

1.  public  
 static  
 void  
 class  
 tačno
  
2.  9x  
 ekran  
 brojStudenata  
 znak+ili-
  
3.  3praseta  
 tri praseta  
 prečnik  
 bzvz
  
4.  specifične i generalne  
 celobrojne i realne

- numeričke i tekstualne
- primitivne i klasne

5.  long

- int
- short
- byte

6.  float

- int
- long
- double
- boolean

7.  jedan bajt

- dva bajta
- tri bajta
- četiri bajta

8.  tačno

- true
- false
- 0
- 1
- netačno

9.  kredit

- Kredit
- KREDIT
- kamatnaStopa
- KamatnaStopa
- kamatna\_stopa

10.  int dužina; int širina;  
 int dužina, širina;  
 int dužina; širina;  
 int dužina, int širina;
11.  System.out.println('Java je kul!');  
 System.println("Java je kul!");  
 System.out.writeln("Java je kul!");  
 System.out.println("Java je kul!");  
 System.out.print("Java je kul!");  
 System.out.printf("Java je kul!");
12.  17 = x;  
 x = 17;  
 x := 17;  
 x == 17;
13.  int x = '17';  
 int x == 17;  
 int x = 17;  
 int x = 17.0;
14.  10  
 11  
 11.25  
 12
15.  1/2  
 1.0/2  
 1.0/2.0  
 (double) (1/2)  
 (double) 1/2  
 1/2.0

16.  2%1  
 15%4  
 25%5  
 37%6

17.  a / b \* b  
 a / (b \* b)  
 1.0 \* a / b \* b  
 1.0 \* a / (b \* b)  
 (double) a / (b \* b)

18.  float f = -34;  
 int t = 23;  
 short s = 10;  
 int t = (int)false;  
 int t = 4.5;

19.  x je 5 i y je 6  
 x je 6.0 i y je 6.0  
 x je 6 i y je 6  
 x je 5.5 i y je 5  
 x je 5.5 i y je 5.0

20.  i = i + (2 - 1);  
 i = i + 1;  
 i += 1;  
 i = 1 + i;  
 i++;

21.  Program prikazuje Mesec je 09 na ekranu.  
 Program prikazuje Mesec je 9 na ekranu.  
 Program prikazuje Mesec je 9.0 na ekranu.  
 Program ima grešku, jer 09 nije ispravno napisana oktalna vrednost.

22.  System.terminate(0)  
 System.halt(0)  
 System.exit(0)  
 System.stop(0)
23.  Math.power(x, y)  
 Math.exp(x, y)  
 Math.pow(x, y)  
 Math.pow(y, x)
24.  char c = 'A';  
 char c = '23';  
 char c = "A";  
 char c = "23";
25.  string s = 'A';  
 String s = '23';  
 String s = "A";  
 String s = "23";
26.  'a'  
 'v'  
 Ništa, jer dolazi do prekida izvršavanja programa usled greške.
27.  String s = "neki string";  
 String s3 = s1 + s2;  
 s1 >= s2  
 int i = s1.length;  
 s1.charAt(0) = '?';
28.  String s3 = s1 - s2;  
 s1 == s2  
 boolean b = s1.compareTo(s2);

- char c = s1[0];  
 char c = s1.charAt(s1.length());  
 char c = s1.charAt(s1.length() - 1);
29.  0  
 1  
 true  
 false
30.  javaprogram  
 Java program  
 Java\_program  
 Javaprogram  
 greška
31.  i = Integer.parseInt(s);  
 i = (new Integer(s)).intValue();  
 i = Integer.valueOf(s).intValue();  
 i = Integer.valueOf(s);  
 i = (int)(Double.parseDouble(s));
32.  d = Double.parseDouble(s);  
 d = (new Double(s)).doubleValue();  
 d = Double.valueOf(s).doubleValue();  
 d = (double)(Integer.parseInt(s));
33.  s = d;  
 s = d.toString();  
 s = (new Double(d)).toString();  
 s = (Double.valueOf(d)).toString();
34.  <  
 =<

- $\geq$
- $\leq$
- $<<$
- $\neq$

35.   $\diamond$
- $\neq$
  - $\diamond \diamond$
  - $=$

36.   $(\text{true}) \ \&\& \ (3 \Rightarrow 4)$
- $!(x > 0) \ \&\& \ (x > 0)$
  - $(x > 0) \ || \ (x < 0)$
  - $(x \neq 0) \ || \ (x = 0)$
  - $(-10 < x < 0)$

37.   $1 < x < 100 \ \&\& \ x < 0$
- $((x < 100) \ \&\& \ (x > 1)) \ || \ (x < 0)$
  - $((x < 100) \ \&\& \ (x > 1)) \ \&\& \ (x < 0)$
  - $(x \neq 0) \ || \ (x = 0)$
  - $(1 > x > 100) \ || \ (x < 0)$
  - $(1 < x < 100) \ || \ (x < 0)$

38.  9
- 10
  - 11
  - 12

39.  9
- 10
  - 11
  - 12

40. Ⓛ – 10

- 0
- 10
- 20
- Programska fragment ima grešku.

## Rešenja zadataka

1.

LISTING 3.1: ImePrezime.java

```
public class ImePrezime {  
  
    public static void main(String[] args) {  
  
        System.out.println(" +-----+");  
        System.out.println(" |           |");  
        System.out.println(" | Dejan Živković |");  
        System.out.println(" |           |");  
        System.out.println(" +-----+");  
    }  
}
```

2.

LISTING 3.2: Inicijali.java

```
public class Inicijali {  
  
    public static void main(String[] args) {  
  
        System.out.println("           ** **");  
        System.out.println("           **");  
        System.out.println(" *****      *****");  
        System.out.println(" **   **      **");  
        System.out.println(" *****      *****");  
    }  
}
```

3.

LISTING 3.3: A4.java

```
public class A4 {  
  
    public static void main(String[] args) {  
  
        final double CM_PO_INČU = 2.54;  
        double širinaPapira = 21.0;
```

```
    double visinaPapira = 29.7;

    System.out.print("Format A4 (210 x 297mm) u inčima: ");
    System.out.printf("%5.2f x %5.2f\n",
                      širinaPapira/CM_PO_INČU, visinaPapira/CM_PO_INČU);
}
```

4.

LISTING 3.4: CelFar.java

```
import java.util.*;

public class CelFar {

    public static void main(String[] args) {

        int f; // broj stepeni Farenhajta
        int c; // broj stepeni Celzijusa

        // Ulaz programa se dobija preko tastature
        Scanner tastatura = new Scanner(System.in);

        // Učitavanje stepena Celzijusa od korisnika
        System.out.print("Unesite stepene Celzijusa: ");
        c = tastatura.nextInt();

        // Izračunavanje stepena Farenhajta po formuli
        f = 9*c/5 + 32;

        // Prikazivanje rezultata na ekranu
        System.out.print(c + " stepeni Celzijusa = ");
        System.out.println(f + " stepeni Farenhajta");
    }
}
```

5.

LISTING 3.5: Kamata.java

```
import java.util.*;

public class Kamata {

    public static void main(String[] args) {

        double depozit;      // početni depozit
        double kamatnaStopa; // godišnja kamatna stopa
    }
}
```

```
    double iznosKamate; // novčani iznos kamate

    Scanner tastatura = new Scanner(System.in);
    System.out.print("Unesite početni depozit: ");
    depozit = tastatura.nextDouble();
    System.out.print("Unesite godišnju kamatnu stopu: ");
    kamatnaStopa = tastatura.nextDouble();

    iznosKamate = depozit * kamatnaStopa;
    depozit = depozit + iznosKamate;

    System.out.println();
    System.out.print("Novčani iznos godišnje kamate: ");
    System.out.println(iznosKamate);
    System.out.print("Uvećan depozit nakon jedne godine: ");
    System.out.println(depozit);
}
}
```

## 6.

LISTING 3.6: PravougliTrougao.java

```
import java.util.*;

public class PravougliTrougao {

    public static void main(String[] args) {

        double kateta1, kateta2, hipotenuza; // strane trougla

        long početnoVreme; // početak izvršavanja, u milisekundama
        long završnoVreme; // kraj izvršavanja, u milisekundama
        double protekloVreme; // razlika vremena, u sekundama

        početnoVreme = System.currentTimeMillis();

        Scanner tastatura = new Scanner(System.in);

        System.out.print("Program računa hipotenuzu ");
        System.out.println("pravouglog trougla za date katete.");

        System.out.print("Unesite dužinu jedne katete: ");
        kateta1 = tastatura.nextDouble();
        System.out.print("Unesite dužinu druge katete: ");
        kateta2 = tastatura.nextDouble();

        hipotenuza = Math.sqrt(kateta1*kateta1 + kateta2*kateta2);
```

```
        System.out.println();
        System.out.print("Hipotenuza pravouglog trougla sa ");
        System.out.print("katetama " + kateta1);
        System.out.print(" i " + kateta2 + " je: ");
        System.out.printf("%8.2f\n", hipotenuza);

        završnoVreme = System.currentTimeMillis();
        protekloVreme = (završnoVreme - početnoVreme)/1000.0;

        System.out.println();
        System.out.print("Vreme izvršavanja u sekundama je: ");
        System.out.println(protekloVreme);
    }
}
```

7.

LISTING 3.7: Tren.java

```
import java.util.*;

public class Tren {

    public static void main(String[] args) {

        int pakovanoVreme; // pakovani podaci vremenskog trenutka
        int godina, mesec, dan, sat, minut;

        // Ulaz programa se dobija preko tastature
        Scanner tastatura = new Scanner(System.in);

        // Učitavanje podataka vremenskog trenutka
        System.out.print("Unesite dan, mesec, godinu: ");
        dan = tastatura.nextInt();
        mesec = tastatura.nextInt();
        godina = tastatura.nextInt();

        System.out.print("Unesite sat, minut: ");
        sat = tastatura.nextInt();
        minut = tastatura.nextInt();

        // Pomeranje uлево bitova vremenskih podataka
        godina = godina << 20;
        mesec = mesec << 16;
        dan = dan << 11;
        sat = sat << 6;
```

```
// Pakovanje podataka vremenskog trenutka  
pakovanovreme = godina | mesec | dan | sat | minut;  
  
// Prikazivanje pakovanog sadržaja na ekranu  
System.out.println("Pakovano vreme: " + pakovanovreme);  
  
// Raspakivanje podataka vremenskog trenutka  
godina = pakovanovreme >>> 20;  
mesec = (pakovanovreme >>> 16) & 0x0000000f;  
dan = (pakovanovreme >>> 11) & 0x0000001f;  
sat = (pakovanovreme >>> 6) & 0x0000001f;  
minut = pakovanovreme & 0x0000003f;  
  
// Prikazivanje raspakovanog sadržaja na ekranu  
System.out.print("Raspakovano vreme: ");  
System.out.print(dan + "." + mesec + "." + godina);  
System.out.println(" " + sat + ":" + minut);  
}  
}
```





# 4

## ***Upravljačke naredbe***

### **Odgovori na pitanja**

1.  if ( $r != 0$ ) System.out.println( $r * r * \text{Math.PI}$ );  
 if ( $r >= 0$ ) System.out.println( $r * r * \text{Math.PI}$ );  
 if ( $r > 0$ ) System.out.println( $r * r * \text{Math.PI}$ );  
 if ( $r > 0$ ) { System.out.println( $r * r * \text{Math.PI}$ ); }  
 if { $r > 0$ } System.out.println( $r * r * \text{PI}$ );  
 if ( $r <= 0$ ) System.out.println( $r * r * \text{Math.PI}$ );  
 if ( $r > 0$ ) System.out.println(Math.pow( $r$ , 2) \* Math.PI);
  
2.   $x > 0$  i  $y > 0$   
  $x < 0$  i  $z > 0$   
  $x < 0$  i  $z < 0$   
 Ništa se ne prikazuje.
  
3.  Programski fragment ima grešku i ne može se izvršiti.  
 Programski fragment se normalno izvšava, ali se ništa ne prikazuje na ekranu.  
 Programski fragment se normalno izvšava i prikazuje se To je tačno! na ekranu.

4.  Fragment A ima grešku.  
 Fragment B ima grešku.  
 Oba fragmenta imaju grešku.  
 Oba fragmenta su ispravna, ali je fragment B bolji.
5.  Ništa se neće prikazati.  
 Plata je veća od 3000  
 Plata je veća od 4000  
 Plata je veća od 3000 u jednom redu i Plata je veća od 4000 u sledećem redu.
6.  cena = 0.0  
 Ništa se neće prikazati.  
 Programski fragment ima grešku i neće se izvršiti.  
 cena = 15.0
7.  Ništa se ne prikazuje.  
 s1 i s2 ukazuju na isti string  
 s1 i s2 ukazuju na različite stringove
8.  Ništa se ne prikazuje.  
 s1 i s2 ukazuju na isti string  
 s1 i s2 ukazuju na različite stringove
9.  Ništa se ne prikazuje.  
 s1 i s2 imaju isti sadržaj  
 s1 i s2 imaju različit sadržaj
10.  Ništa se ne prikazuje.  
 s1 i s2 imaju isti sadržaj  
 s1 i s2 imaju različit sadržaj
11.  Ništa se ne prikazuje.

- s1 i s2 ukazuju na isti string
  - s1 i s2 imaju isti sadržaj
  - s1 i s2 imaju različit sadržaj
12.  1  
 2  
 3  
 4
13.  abcd  
 a  
 aA  
 A
14.  ocena je 15  
 ocena je 15 u jednom redu i ocena je 15 ili 30 u sledećem redu.  
 Ništa se neće prikazati.  
 Pograšna ocena
15.  Programski fragment ima grešku, jer nedostaju potrebne naredbe break.  
 Programski fragment ima grešku, jer nedostaje slučaj default u naredbi switch.  
 Programski fragment ima grešku, jer kontrolna promenljiva d u naredbi switch ne može biti tipa double.  
 Programski fragment nema grešaka.
16.  Programski fragment ima grešku i neće se izvršiti.  
 20  
 Ništa se neće prikazati.  
 Stalno će se prikazivati 20 u beskonačnoj petlji.
17.  9

- 10
- 11
- 0

18.  Uslov brojač  $< 10$  je uvek tačan u tački A.
- Uslov brojač  $< 10$  je uvek netačan u tački A.
  - Uslov brojač  $< 10$  je uvek tačan u tački B.
  - Uslov brojač  $< 10$  je uvek netačan u tački B.
  - Uslov brojač  $< 10$  je uvek tačan u tački C.
  - Uslov brojač  $< 10$  je uvek netačan u tački C.

19.  9
- 10
  - 11
  - 0

20.  Programski fragment ima grešaku i neće se izvršiti.
- $i = 5$
  - Ništa se neće prikazati.
  - Stalno će se prikazivati  $i = 1$  u beskonačnoj petlji.
21.  Programski fragment ima grešku, jer nedostaje treći deo (završnica) u zagradama for petlje.
- Programski fragment ima grešku, jer kontrolna promenljiva d u zagradama for petlje ne može biti tipa double.
  - Pošto je uslov  $d < 10$  uvek tačan, for petlja je beskonačna.
  - Programski fragment nema grešaka i normalno se izvršava.

22.  10
- 11
  - 12
  - 13
  - 45

23.  Da.

Ne.

24.  Da.

Ne.

25.  Program se neće izvršiti, jer se tačka-zapeta nalazi odmah iza zagrada for petlje.

Program će se bez problema izvršiti i prikazaće se 4 na ekranu.

Program će se bez problema izvršiti i prikazaće se 14 na ekranu.

U programu je for petlja ekvivalentna petlji for ( $i = 0; i < 10; i++$ ) { };

26.  Da.

Ne.

27.  5

6

7

8

28.  Da.

Ne.

29.  20 19 18 17 16

20 15 10 5

15 10 5 0

15 10 5

30.  Naredba sa oznakom spetlja.

Naredba sa oznakom upetlja.

Naredba sa oznakom nastavak.

Nijedna naredba, nego se program odmah završava.

31. Ⓛ Kontrola se prenosi u tačku B radi izvršavanja sledeće iteracije spoljašnje petlje sa oznakom spetlja.
- Kontrola se prenosi u tačku A radi izvršavanja sledeće iteracije unutrašnje petlje sa oznakom upetlja.
- Naredba sa oznakom nastavak.
- Nijedna naredba, nego se program odmah završava.



## Rešenja zadataka

1.

LISTING 4.1: Uskrs1.java

```
import java.util.*;  
  
public class Uskrs1 {  
  
    public static void main(String[] args) {  
  
        Scanner tastatura = new Scanner(System.in);  
  
        System.out.print("Unesite godinu za datum Uskrsa: ");  
        int g = tastatura.nextInt();  
  
        int a = g % 19;  
        int b = g % 4;  
        int c = g % 7;  
        int d = (19 * a + 24) % 30;  
        int e = (2 * b + 4 * c + 6 * d + 5) % 7;  
  
        System.out.print("Katolički Uskrs je u nedelju, ");  
        int f = 22 + d + e;  
        if (f > 31)  
            System.out.printf("%d. aprila %d.\n", f - 31, g);  
        else  
            System.out.printf("%d. marta %d.\n", f, g);  
    }  
}
```

2.

LISTING 4.2: Uskrs2.java

```
import java.util.*;  
  
public class Uskrs2 {  
  
    public static void main(String[] args) {  
  
        Scanner tastatura = new Scanner(System.in);  
  
        System.out.print("Unesite godinu za datum Uskrsa: ");  
        int g = tastatura.nextInt();  
  
        int a = g % 19;  
        int b = g % 4;
```

```
int c = g % 7;
int d = (19 * a + 24) % 30;
int e = (2 * b + 4 * c + 6 * d + 5) % 7;

int f = 22 + d + e;
int m = 3;
if (f > 31) {
    f = f - 31;
    m = 4;
}
if (g == 1954 || g == 1981 || g == 2049 || g == 2076)
    f = f - 7;
System.out.print("Katolički Uskrs je u nedelju, ");
System.out.printf("%d. %d. %d.\n", f, m, g);
}
```

3.

LISTING 4.3: PismoGlava1.java

```
import java.util.*;

public class PismoGlava1 {

    public static void main(String[] args) {

        final int PISMO = 0;
        int brojBacanja, ishodBacanja;
        int brojPisma, brojGlava;
        Scanner tastatura = new Scanner(System.in);

        while(true) {
            System.out.print("Unesite broj bacanja novčića: ");
            brojBacanja = tastatura.nextInt();
            if (brojBacanja == 0) break;
            brojPisma = 0;
            brojGlava = 0;
            for (int i = 0; i < brojBacanja; i++) {
                ishodBacanja = (int)(Math.random() + 0.5);
                if (ishodBacanja == PISMO)
                    brojPisma++;
                else
                    brojGlava++;
            }
            System.out.print("Broj pisma    Broj glava    ");
            System.out.print("Broj pisma/Broj bacanja    ");
            System.out.println("Broj glava/Broj bacanja    ");
        }
    }
}
```

```
        System.out.printf("%8d %12d %17.2f %25.2f\n",
                           brojPisma, brojGlava,
                           (double)brojPisma/brojBacanja,
                           (double)brojGlava/brojBacanja);
    }
}
```

4.

LISTING 4.4: ZmajskeOči.java

```
public class ZmajskeOči {

    public static void main(String[] args) {

        int brojBacanja = 0; // brojač bacanja dve kocke
        int kocka1;          // broj koji je pao na prvoj kocki
        int kocka2;          // broj koji je pao na drugoj kocki

        do {
            kocka1 = (int)(Math.random()*6) + 1; // baci prvu kocku
            kocka2 = (int)(Math.random()*6) + 1; // baci drugu kocku
            brojBacanja++;
            System.out.printf("%4d. bacanje: kocka1 = %d, kocka2 = %d\n",
                              brojBacanja, kocka1, kocka2);
        } while ((kocka1 != 1) || (kocka2 != 1));
    }
}
```

5.

LISTING 4.5: Niz3n1.java

```
import java.util.*;

public class Niz3n1 {

    public static void main(String[] args) {

        int n; // elementi niza

        Scanner tastatura = new Scanner(System.in);

        System.out.print("Unesite početni broj niza: ");
        n = tastatura.nextInt();
        while(n <= 0) {
            System.out.println();
            System.out.print("Početni broj niza ");
        }
    }
}
```

```
System.out.println("mora biti pozitivan!");
System.out.print("Unesite početni broj niza: ");
n = tastatura.nextInt();
}

System.out.println();
System.out.println(n);
while(n != 1) {
    if (n % 2 == 0)
        n = n / 2;
    else
        n = 3 * n + 1;
    System.out.println(n);
}
}
```

## 6.

LISTING 4.6: NZD1.java

```
import java.util.*;

public class NZD1 {

    public static void main(String[] args) {

        int x, y; // dati brojevi
        int d; // (najveći) zajednički delilac

        Scanner tastatura = new Scanner(System.in);

        System.out.print("Unesite dva pozitivna cela broja: ");
        x = tastatura.nextInt();
        y = tastatura.nextInt();

        d = x < y ? x : y; // početni delilac

        while((x % d != 0) || (y % d != 0))
            d = d - 1;

        System.out.println("NZD(" + x + "," + y + ") = " + d);
    }
}
```

7.

LISTING 4.7: NBD.java

```
import java.util.*;  
  
/** Program prikazuje broj sa najvećim brojem delioca u  
 * intervalu od 1 do n */  
public class NBD {  
  
    public static void main(String[] args) {  
  
        int brojSaNBD = 1;      // broj sa najvećim brojem delioca  
        int maxBrojDelioca = 1; // njegov najveći broj delioca  
        int brojDelioca;       // broj delioca aktuelnog broja  
  
        Scanner tastatura = new Scanner(System.in);  
  
        System.out.println(  
            "Unesite gornju granicu intervala brojeva (veću od 1)");  
        System.out.print(  
            "u kojem se traži broj sa najvećim brojem delioca: ");  
        int n = tastatura.nextInt();  
  
        for (int k = 1; k <= n; k++) {  
            brojDelioca = 0;  
            for (int j = 1; j <= k; j++)  
                if (k % j == 0)  
                    brojDelioca++;  
            if (maxBrojDelioca < brojDelioca) {  
                maxBrojDelioca = brojDelioca;  
                brojSaNBD = k;  
            }  
        }  
  
        System.out.println(  
            "Broj sa najvećim brojem delioca je " + brojSaNBD);  
        System.out.println(  
            "Broj delioca tog broja je " + maxBrojDelioca);  
    }  
}
```

8.

LISTING 4.8: Romb.java

```
import java.util.*;  
  
public class Romb {
```

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

    int brojRedova;    // broj redova romba, tj. broj
                       // zvezdica u srednjem redu romba
    int brojBlankova; // broj vodećih blankova u redu
    int brojZvezdica; // broj zvezdica u redu

    Scanner tastatura = new Scanner(System.in);

    System.out.print("Unesite broj redova romba: ");
    brojRedova = tastatura.nextInt();

    if (brojRedova % 2 == 0) // broj redova romba
        brojRedova++;           // je paran?

    for (int i = 1; i <= brojRedova; i++) {

        // Broj vodećih blankova u aktuelnom redu
        brojBlankova = brojRedova/2 - i + 1;
        if (brojBlankova < 0)
            brojBlankova = -brojBlankova;

        // Broj zvezdica u aktuelnom redu
        brojZvezdica = 2*(brojRedova/2 - brojBlankova)+1;

        // Prikazati vodeće blankove u redu
        for (int j = 1; j <= brojBlankova; j++)
            System.out.print(" ");

        // Prikazati zvezdice iza blankova u redu
        for (int j = 1; j <= brojZvezdica; j++)
            System.out.print("*");

        // Završiti prikazivanje reda
        System.out.println();
    }
}
```

## 9.

LISTING 4.9: SlovaCifre.java

```
import java.util.*;

public class SlovaCifre {

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

```
String red; // ulazni red
int brojSlova=0, brojCifara=0;
Scanner tastatura = new Scanner(System.in);

System.out.print("Unesite jedan red teksta: ");
red = tastatura.nextLine();

System.out.println("Svi znakovi ulaznog reda su: ");
for (int i = 0; i < red.length(); i++) {
    char znak = red.charAt(i);
    System.out.printf("%3d. znak %c\n", i+1, znak);
    if (Character.isLetter(znak))
        brojSlova++;
    if (Character.isDigit(znak))
        brojCifara++;
}

System.out.println(
    "Broj slova u ulaznom redu je: " + brojSlova);
System.out.println(
    "Broj cifara u ulaznom redu je: " + brojCifara);
}
```

10.

LISTING 4.10: ListaReči.java

```
import java.util.*;

public class ListaReči {

    public static void main(String[] args) {

        String red; // ulazni red
        int r = 0; // indeks početka jedne reči
        boolean izvanReči = true; // stanje unutar/izvan reči

        Scanner tastatura = new Scanner(System.in);
        System.out.print("Unesite jedan red teksta: ");
        red = tastatura.nextLine();

        System.out.println();
        System.out.println("Sve reči u ulaznom reda su: ");

        for (int i = 0; i < red.length(); i++) {
```

```
char znak = red.charAt(i);
if (Character.isLetter(znak)) {
    if (izvanReči) // početak reči
        r = i;
    izvanReči = false;
}
else {
    if (!izvanReči) // kraj reči
        System.out.println(red.substring(r, i));
    izvanReči = true;
}
if (!izvanReči) // poslednja reč?
    System.out.println(red.substring(r, red.length()));
}
}
```



# 5

## *Metodi*

### Odgovori na pitanja

1.  void  
 return  
 public  
 static
  
2.  imena metoda  
 imena metoda i liste parametara  
 tipa rezultata, imena metoda i liste parametara  
 liste parametara
  
3.  Ime metoda je protected.  
 Ime metoda je nadī.  
 Tip rezultata metoda je int.  
 Tip rezultata metoda je boolean.  
 Tip rezultata metoda je double.  
 Broj parametara metoda je tri.  
 Broj parametara metoda je šest.

4.  public static Main(String[] args)  
 public static Main(String args[])  
 public void main(String[] args)  
 public static void main(String[] args)  
 public static main(String[] args)
5.  uglastih (srednjih) zagrada  
 običnih (malih) zagrada  
 vitičastih (velikih) zagrada  
 dvostrukih apostrofa  
 jednostrukih apostrofa
6.  pozivanje metoda.  
 prenošenje po vrednosti.  
 prenošenje po referenci.  
 prenošenje po imenu.
7.  void  
 return  
 public  
 static
8.  Da.  
 Ne.
9.  Da.  
 Ne.
10.  Na ekranu se prikazuje aaaaa.  
 Na ekranu se prikazuje aaaa.  
 Na ekranu se prikazuje aaa.  
 Na ekranu se prikazuje aa.  
 Na ekranu se prikazuje a.

11.  0  
 1  
 2  
 3
12.  Program na ekranu prikazuje int i zatim 5.  
 Program na ekranu prikazuje double i zatim 5.  
 Program na ekranu prikazuje double.  
 Program na ekranu prikazuje int.  
 Program ima grešku, jer se ne može odrediti koju verziju preopterećenog metoda xMetod() treba pozvati.
13.  Program ima grešku, jer se ne može odrediti koju verziju preopterećenog metoda m() treba pozvati.  
 Program ima grešku, jer je druga verzija preopterećenog metoda m() definisana ali se nigde ne poziva.  
 Program se normalno izvršava i prikazuje 2 jedanput.  
 Program se normalno izvršava i prikazuje 2 dvaput.
14.  globalna promenljiva  
 statička promenljiva  
 blokovska promenljiva  
 lokalna promenljiva
15.  0  
 1  
 2  
 Promenljiva k nije definisana izvan tog bloka.
16.  Rekurzivni metodi su oni koji pozivaju sami sebe, bilo direktno ili indirektno.  
 Rekurzivni metodi se pozivaju drugačije od nerekurzivnih metoda.  
 Rekurzivni metodi rešavaju neki zadatak svođenjem polaznog problema na sličan prostiji problem (ili više njih).

- Svaki rekurzivni metod mora imati *bazni slučaj* za najprostiji zadatak čije se rešenje ne dobija rekurzivnim pozivom.
17.  Rezultat poziva faktorijel(3) je 2.  
 Rezultat poziva faktorijel(3) je 3.  
 Rezultat poziva faktorijel(3) je 6.  
 Poziv faktorijel(3) izaziva grešku pošto proizvodi beskonačan lanac poziva istog metoda faktorijel().
18.  Program na ekranu prikazuje 1 2 3 4 5.  
 Program na ekranu prikazuje 1 2 3 4 5 i zatim grešku o prekoračenju granica indeksa niza x.  
 Program na ekranu prikazuje 5 4 3 2 1.  
 Program na ekranu prikazuje 5 4 3 2 1 i zatim grešku o prekoračenju granica indeksa niza x.
19.  Program proizvodi beskonačan lanac poziva istog metoda rMetod().  
 Program na ekranu prikazuje 1 2 3.  
 Program na ekranu prikazuje 3 2 1.  
 Program na ekranu prikazuje 1 2.  
 Program na ekranu prikazuje 2 1.
20.  Program na ekranu ne prikazuje ništa.  
 Program na ekranu prikazuje 1 2.  
 Program na ekranu prikazuje 2 1.  
 Program na ekranu beskonačno prikazuje 1 1 1 1 1 ....  
 Program na ekranu beskonačno prikazuje 2 2 2 2 2 ....
21.  Pozivom rMetod(5) se isti metod rMetod() poziva još 3 puta.  
 Pozivom rMetod(5) se isti metod rMetod() poziva još 4 puta.  
 Pozivom rMetod(5) se isti metod rMetod() poziva još 5 puta.  
 Pozivom rMetod(5) se isti metod rMetod() poziva još 6 puta.
22.  Rezultat poziva rMetod(5) je 5.

- Rezultat poziva rMetod(5) je 10.
  - Rezultat poziva rMetod(5) je 15.
  - Poziv rMetod(5) proizvodi beskonačan lanac poziva istog metoda rMetod().
23.   $n$  je 1.
- $n$  je manje od 1.
  - $n$  je veće od 1.
  - Nema baznog slučaja.
24.   $(s.charAt(0) \neq s.charAt(s.length() - 1))$
- $(s.charAt(0) \neq s.charAt(s.length()))$
  - $((s.charAt(1) \neq s.charAt(s.length() - 1))$
  - $(s.charAt(1) \neq s.charAt(s.length()))$
25.  palindrom(s)
- palindrom(s, leviKraj, desniKraj)
  - palindrom(s, leviKraj + 1, desniKraj)
  - palindrom(s, leviKraj, desniKraj - 1)
  - palindrom(s, leviKraj + 1, desniKraj - 1)
26.  sortiraj(niz)
- sortiraj(niz, niz.length)
  - sortiraj(niz, niz.length - 1)
  - sortiraj(niz, niz.length + 1)
27.  nađiBroj(niz, broj, sredina + 1, leviKraj)
- nađiBroj(niz, broj, sredina - 1, leviKraj)
  - nađiBroj(niz, broj, desniKraj, sredina + 1)
  - nađiBroj(niz, broj, sredina + 1, desniKraj)

## Rešenja zadataka

1.

LISTING 5.1: NZD2.java

```
import java.util.*;  
  
public class NZD2 {  
  
    public static void main(String[] args) {  
  
        Scanner tastatura = new Scanner(System.in);  
  
        System.out.print("Unesite dva pozitivna cela broja: ");  
        int x = tastatura.nextInt();  
        int y = tastatura.nextInt();  
  
        System.out.println("NZD(" + x + "," + y + ") = " + NZD(x, y));  
    }  
  
    public static int NZD(int x, int y) {  
  
        int z; // naredni element niza  
  
        if (x < y) { // zameniti x i y da bude x >= y  
  
            int t = x;  
            x = y;  
            y = t;  
        }  
  
        while(true) {  
  
            z = x % y;  
            if (z == 0) break;  
            x = y;  
            y = z;  
        }  
        return y;  
    }  
}
```

2.

LISTING 5.2: Niz3n1m.java

```
import java.util.*;
```

```
public class Niz3n1m {  
  
    public static void main(String[] args) {  
  
        int n; // početni broj niza  
  
        Scanner tastatura = new Scanner(System.in);  
  
        System.out.print("Unesite početni broj niza: ");  
        n = tastatura.nextInt();  
        while(n <= 0) {  
            System.out.println();  
            System.out.println("Početni broj niza mora biti pozitivan!");  
            System.out.print("Unesite početni broj niza: ");  
            n = tastatura.nextInt();  
        }  
  
        triN1(n);  
    }  
  
    public static void triN1(int N) {  
  
        System.out.println();  
        System.out.println(N);  
        while(N != 1) {  
            if (N % 2 == 0)  
                N = N / 2;  
            else  
                N = 3 * N + 1;  
            System.out.println(N);  
        }  
    }  
}
```

## 3.

LISTING 5.3: KapString.java

```
import java.util.*;  
  
public class KapString {  
  
    public static void main(String[] args) {  
  
        String red; // ulazni red  
  
        Scanner tastatura = new Scanner(System.in);  
        System.out.print("Unesite jedan red teksta: ");
```

```
    red = tastatura.nextLine();

    System.out.println();
    System.out.println("Sve reči sa početnim velikim slovom su: ");
    System.out.println(kapitalizuj(red));
}

public static String kapitalizuj(String s) {

    String t = ""; // rezultujući string
    boolean izvanReči = true; // stanje unutar/izvan reči

    for (int i = 0; i < s.length(); i++) {
        char znak = s.charAt(i);
        if (Character.isLetter(znak)) {
            if (izvanReči) // početak reči
                t = t + Character.toUpperCase(znak);
            else
                t = t + znak;
            izvanReči = false;
        }
        else {
            t = t + znak;
            izvanReči = true;
        }
    }
    return t;
}
}
```

## 4.

LISTING 5.4: HeksSistem.java

```
import java.util.*;

public class HeksSistem {

    public static void main(String[] args) {

        final String HEKS_CIFRE = "0123456789aAbBcCdDeEfF";

        Scanner tastatura = new Scanner(System.in);
        System.out.print("Unesite heksadekadni broj: ");
        String h = tastatura.nextLine();

        for (int i = 0; i < h.length(); i++) {
            if (HEKS_CIFRE.indexOf(h.charAt(i)) == -1) {
```

```
        System.out.println("Greška: uneti broj nije heksadekadni!");
        System.exit(-1);
    }
}

System.out.println("Njegova dekadna vrednost je: " + heksBroj(h));
}

private static int heksVrednost(char z) {

    switch(z) {
        case '0' : return 0;
        case '1' : return 1;
        case '2' : return 2;
        case '3' : return 3;
        case '4' : return 4;
        case '5' : return 5;
        case '6' : return 6;
        case '7' : return 7;
        case '8' : return 8;
        case '9' : return 9;
        case 'a' : case 'A' : return 10;
        case 'b' : case 'B' : return 11;
        case 'c' : case 'C' : return 12;
        case 'd' : case 'D' : return 13;
        case 'e' : case 'E' : return 14;
        case 'f' : case 'F' : return 15;
        default : return -1;
    }
}

public static long heksBroj(String s) {

    long d = 0;
    for (int i = 0; i < s.length(); i++)
        d = d * 16 + heksVrednost(s.charAt(i));
    return d;
}
```

## 5.

LISTING 5.5: DveKocke1.java

```
public class DveKocke1 {

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

```
System.out.println("Zbir dve kocke      Prosečan broj bacanja");
System.out.println("-----      -----");

for (int i = 2; i < 13; i++)
    System.out.printf("%7d %24.2f\n", i, prosekZaZbir(i));
}

public static int baciZaZbir(int zbir) {

    int brojBacanja = 0; // brojač bacanja dve kocke
    int kocka1; // broj koji je pao na prvoj kocki
    int kocka2; // broj koji je pao na drugoj kocki

    do {
        kocka1 = (int) (Math.random() * 6) + 1; // baci prvu kocku
        kocka2 = (int) (Math.random() * 6) + 1; // baci drugu kocku
        brojBacanja++; // uračunati bacanje

    } while ((kocka1 + kocka2) != zbir);

    return brojBacanja;
}

public static double prosekZaZbir(int zbir) {

    final int BROJ_PONAVLJANJA = 100000;
    int ukupnoBacanja = 0; // ukupan broj bacanja za
                           // dati zbir dve kocke

    for (int i = 0; i < BROJ_PONAVLJANJA; i++)
        ukupnoBacanja = ukupnoBacanja + baciZaZbir(zbir);

    return (double)ukupnoBacanja/BROJ_PONAVLJANJA;
}
}
```

# 6

## *Klase i objekti*

### Odgovori na pitanja

1.  Objekat  
 Metod  
 Promenljiva  
 Klasa
  
2.  statički (klasni) i nestatički (objektni)  
 lokalni i globalni  
 proceduralni i neproceduralni  
 spoljašnji i unutrašnji
  
3.  method  
 class  
 main  
 object
  
4.  Glavni metod  
 Metod bez argumenata  
 Konstruktor

- Rekurzivni metod
5.  Podrazumevani konstruktor bez argumenata se automatski dodaje ukoliko u klasi nije eksplisitno definisan nijedan konstruktor.
- U klasi se mora eksplisitno definisati bar jedan konstruktor.
- Konstruktori nemaju tip rezultata, čak ni void.
- Konstruktori moraju imati isto ime kao klasa u kojoj se definišu.
- Konstruktori se pozivaju koristeći operator new kada se konstruiše objekat.
6.  Program ima grešku, jer klasa A nije javna klasa.
- Program ima grešku, jer klasa A nema podrazumevani konstruktor.
- Program nema grešaka i normalno se izvršava ništa ne prikazujući na ekranu.
- Program ima grešku koja se može ispraviti ukoliko se naredba A a = new A(); u metodu main promeni u naredbu A a = new A("poruka");.
7.  Program ima grešku, jer klasa B nije javna klasa.
- Program ima grešku, jer klasa B nema podrazumevani konstruktor.
- Program ima grešku, jer klasa B nema konstruktor sa parametrom tipa int.
- Program nema grešaka i normalno se izvršava ništa ne prikazujući na ekranu.
8.  Promenljiva k sadrži celobrojnu vrednost.
- Promenljivoj k se može dodeliti celobrojna vrednost.
- Promenljiva k sadrži objekat klase Krug.
- Promenljiva k sadrži referencu na objekat klase Krug.
9.  Program ima grešku, jer promenljiva t nije inicijalizovana.
- Program ima grešku, jer promenljiva x nije inicijalizovana.
- Program ima grešku, jer klasa Test nema podrazumevani konstruktor.

- Program ima grešku, jer se u nekoj klasi ne može deklarisati promenljiva tipa te iste klase kao što je to ovde slučaj sa promenljivom t.
  - Program ima grešku, jer promenljiva t ima vrednost null kada se prikazuje polje t.x.
  - Program nema grešaka i normalno se izvršava ništa ne prikazujući na ekranu.
10.  true, 1, null  
 false, 0, null  
 true, 0, null  
 false, 1, null  
 false, 0, void
11.  Lokalne promenljive ne dobijaju automatski početne vrednosti.  
 Polja objekata dobijaju automatski početne vrednosti.  
 Promenljiva nekog primitivnog tipa sadrži vrednost tog primitivnog tipa.  
 Promenljiva nekog klasnog tipa ukazuje na memorijsku adresu u kojoj se nalazi objekat klase.  
 Celobrojna vrednost koja predstavlja važeću memorijsku adresu može se dodeliti promenljivoj klasnog tipa.
12.  Program ima grešku, jer promenljiva prečnik nije inicijalizovana.  
 Program ima grešku, jer je konstanta PI definisana unutar metoda.  
 Program ima grešku, jer konstanta PI ima previše decimala.  
 Program ima grešku, jer konstanta PI ima premalo decimala.  
 Program nema grešaka i normalno se izvršava.
13.  Program ima grešku, jer se metod System.out.println() ne može koristiti u konstruktoru klase.  
 Program ima grešku, jer promenljiva x nije inicijalizovana.  
 Program ima grešku, jer klasa Test nema podrazumevani konstruktor.

- Program ima grešku, jer se u nekoj klasi ne može konstruisati objekat te iste klase.
  - Program nema grešaka i normalno se izvršava prikazujući 0 na ekranu.
14.  Objektna promenljiva sadrži neki objekat.  
 Promenljiva klasnog tipa sadrži neki objekat.  
 Neki objekat može sadržati druge objekte.  
 Neki objekat može sadržati reference na druge objekte.
15.  Javna  
 Privatna  
 Objektna (instancna)  
 Statička (klasna)
16.  Da.  
 Ne.
17.  U redu 5.  
 U redu 9.  
 U oba reda 5 i 9.  
 U nijednom redu.
18.  statički metod  
 klasni metod  
 objektni metod  
 glavni metod
19.  final static MAX\_CENA = 99.98;  
 final static float MAX\_CENA = 99.98;  
 static double MAX\_CENA = 99.98;  
 final double MAX\_CENA = 99.98;  
 final static double MAX\_CENA = 99.98;

20.  Program ima grešku, jer metod xMetod() ne vraća nijednu vrednost.  
 Program ima grešku, jer metod xMetod() nije definisan da bude statički.  
 Program prikazuje n je 1 na ekranu.  
 Program prikazuje n je 2 na ekranu.  
 Program prikazuje n je 3 na ekranu.
21.  t2.i je 1, t2.s je 1  
 t2.i je 1, t2.s je 2  
 t2.i je 2, t2.s je 2  
 t2.i je 2, t2.s je 1
22.  t3.i je 1, t3.s je 1  
 t3.i je 1, t3.s je 2  
 t3.i je 1, t3.s je 3  
 t3.i je 3, t3.s je 1  
 t3.i je 3, t3.s je 3
23.  Program ima grešku, jer klasa A ima privatni podrazumevani konstruktor.  
 Program ima grešku, jer klasa A ima prazan podrazumevani konstruktor.  
 Program ima grešku, jer promenljiva n nije inicijalizovana.  
 Program nema grešaka i normalno se izvršava.
24.  b.n = 101  
 b.n = 100  
 b.n = 99  
 b.n = 98  
 b.n = 0
25.  k = 101  
 k = 100

- k = 99
  - k = 98
  - k = 0
26.  Klasa Krug ima grešku, jer nema metod main().
- Svaki konstruisani objekat klase Krug će imati prečnik 0. Na primer, naredbom Krug k = new Krug(2.35) dobija se krug k prečnika 0 iako se očekuje da njegov prečnik bude 2.35.
- Klasa Krug ima grešku, jer se ne može pisati naredba dodele prečnik = prečnik; u konstruktoru.
- Klasa Krug ima grešku, jer nema podrazumevani konstruktor.
27.  this.t() u konstruktoru Test(double x) može se pojednostaviti i zameniti samo sa t().
- this.x u konstruktoru Test(double x) može se pojednostaviti i zameniti samo sa x.
- this(23) u konstruktoru Test() mora se pozvati pre naredbe System.out.println("Podrazumevani konstruktor");.
- this(23) u konstruktoru Test() mora se zameniti sa this(23.0).

## Rešenja zadataka

1.

LISTING 6.1: MesečniKalendor.java

```
import java.util.*;  
  
public class MesecniKalendor {  
  
    public static void main(String args[]) {  
  
        Scanner tastatura = new Scanner(System.in);  
        System.out.print("Mesec i godina kalendara: ");  
        int mes = tastatura.nextInt();  
        int god = tastatura.nextInt();  
  
        GregorianCalendar kalendar = new GregorianCalendar();  
  
        int brojDana = 0;  
  
        switch(mes) {  
            case 1: case 3: case 5: case 7: case 8: case 10: case 12:  
                brojDana = 31;  
                break;  
            case 2:  
                if(kalendar.isLeapYear(god))  
                    brojDana = 29;  
                else  
                    brojDana = 28;  
                break;  
            case 4: case 6: case 9: case 11:  
                brojDana = 30;  
                break;  
            default:  
                System.out.println(  
                    "Greška: pograšna specifikacija meseca!");  
                System.exit(-1);  
        }  
  
        // Kalendar počinje od prvog dana datog meseca i godine  
        kalendar.set(god, mes - 1, 1); // meseci počinju od 0  
  
        int prviDan = kalendar.get(Calendar.DAY_OF_WEEK);  
  
        System.out.println(" PON UTO SRE ČET PET SUB NED");  
  
        int k = 0; // trenutna kolona prikaza dana kalendara  
        // SUNDAY=1, MONDAY=2, ..., SATURDAY=7
```

```
for (int i = Calendar.MONDAY; i <= Calendar.SATURDAY; i++) {
    if (prviDan == i)
        break;
    System.out.print("      ");
    k++;
}
for (int d = 1; d <= brojDana; d++) {
    System.out.printf("%5d", d);
    k++;
    if (k == 7) {
        System.out.println();
        k = 0;
    }
}
System.out.println();
}
```

2.

LISTING 6.2: TKK.java

```
public class TKK {

    public static void main(String[] args) {

        Krug k1 = new Krug(new Tačka(0, 0), 1);
        System.out.println("Krug: " + k1);
        System.out.println("Obim: " + k1.obim());
        System.out.println("Površina: " + k1.površina());

        System.out.println();

        Krug k2 = new Krug(new Tačka(1.5, 2), 2);
        System.out.println("Krug: " + k2);
        System.out.println("Opisan kvadrat: " + k2.opisanKvadrat());
    }
}

/** Tačka u ravni sa koordinatama (x,y) */
class Tačka {
    private double x, y;                      // koordinate tačke

    public Tačka(double x, double y) { // konstruktor
        this.x = x;
        this.y = y;
    }
}
```

```
public double getX() {
    return x;
}

public double getY() {
    return y;
}

public double rastojanjeOdPočetka() {
    return Math.sqrt(x*x + y*y);
}
}

/** Kvadrat u ravni predstavljen koordinatama donjeg levog
 * temena i dužinom stranice */
class Kvadrat {
    private Tačka teme;      // donje levo teme
    private double strana; // stranica

    public Kvadrat(Tačka teme, double strana) { // konstruktor
        this.teme = teme;
        this.strana = strana;
    }

    public Tačka getTeme() {
        return teme;
    }

    public double getStrana() {
        return strana;
    }

    public double obim() {
        return 4 * strana;
    }

    public double površina() {
        return strana * strana;
    }

    public String toString() {
        return "A=" + teme.getX() + "," + teme.getY() + ", a=" + strana;
    }
}

/** Krug u ravni predstavljen koordinatama centra
 * i dužinom poluprečnika */
```

```
class Krug {  
    private Tačka centar;      // centar kruga  
    private double poluprečnik; // poluprečnik  
  
    public Krug(Tačka centar, double poluprečnik) { // konstruktor  
        this.centar = centar;  
        this.poluprečnik = poluprečnik;  
    }  
  
    public Tačka getCentar() {  
        return centar;  
    }  
  
    public double getPoluprečnik() {  
        return poluprečnik;  
    }  
  
    public double obim() {  
        return 2 * Math.PI * poluprečnik;  
    }  
  
    public double površina() {  
        return poluprečnik * poluprečnik * Math.PI;  
    }  
  
    public boolean sadržiTačku(Tačka A) {  
        double cx = centar.getX(); // koordinate centra kruga  
        double cy = centar.getY();  
  
        double ax = A.getX();      // koordinate date tačke A  
        double ay = A.getY();  
  
        // Rastojanje date tačke od centra kruga  
        double d = Math.sqrt((cx-ax)*(cx-ax)+(cy-ay)*(cy-ay));  
  
        if (d > poluprečnik)  
            return false;  
        else  
            return true;  
    }  
  
    public Kvadrat opisanKvadrat() {  
        double cx = centar.getX();      // koordinate centra kruga  
        double cy = centar.getY();  
  
        double ax = cx - poluprečnik; // koordinate donjeg levog  
        double ay = cy - poluprečnik; // temena kvadrata
```

```
    double d = 2 * poluprečnik; // dužina strane kvadrata

    return new Kvadrat(new Tačka(ax, ay), d);
}

public String toString() {
    return "C=(" + centar.getX() + "," + centar.getY() + "), r=" + poluprečnik;
}
}
```

3.

LISTING 6.3: KompleksanBroj.java

```
public class KompleksanBroj {

    private double r, i; // realni i imaginarni deo

    // Konstruktor
    public KompleksanBroj(double r, double i) {
        this.r = r;
        this.i = i;
    }

    // Geter metodi
    public double Re() { return r; }

    public double Im() { return i; }

    // Moduo kompleksnog broja
    public double moduo() { return Math.sqrt(r*r + i*i); }

    // Statički metod za konjugovan kompleksan broj:
    // KompleksanBroj z = KompleksanBroj.konjugovanBroj(x)
    public static KompleksanBroj konjugovanBroj(KompleksanBroj a) {
        return new KompleksanBroj(a.r, -a.i);
    }

    // Objektni metod za konjugovan kompleksan broj:
    // KompleksanBroj z = x.konjugovanBroj();
    public KompleksanBroj konjugovanBroj() {
        return new KompleksanBroj(this.r, -this.i);
    }

    // Statički metod za sabiranje dva kompleksna broja:
    // KompleksanBroj z = KompleksanBroj.zbir(x, y)
    public static KompleksanBroj zbir(
```

```
        KompleksanBroj a, KompleksanBroj b) {
    return new KompleksanBroj(a.r + b.r, a.i + b.i);
}

// Objektni metod za sabiranje dva kompleksna broja:
// KompleksanBroj z = x.dodaj(y);
public KompleksanBroj dodaj(KompleksanBroj a) {
    return new KompleksanBroj(this.r + a.r, this.i + a.i);
}

// Statički metod za proizvod dva kompleksna broja:
// KompleksanBroj z = KompleksanBroj.proizvod(x, y)
public static KompleksanBroj proizvod(
        KompleksanBroj a, KompleksanBroj b) {
    return new KompleksanBroj(
        a.r * b.r - a.i * b.i, a.r * b.i + a.i * b.r);
}

// Objektni metod za proizvod dva kompleksna broja:
// KompleksanBroj z = x.pomnoži(y);
public KompleksanBroj pomnoži(KompleksanBroj a) {
    return new KompleksanBroj(
        this.r * a.r - this.i * a.i, this.r * a.i + this.i * a.r);
}

public String toString() { return "[" + r + "," + i + "]"; }

public static void main(String[] args) {

    KompleksanBroj x = new KompleksanBroj(1, 1);

    System.out.println("x = " + x);
    System.out.println("Re x = " + x.Re());
    System.out.println("Im x = " + x.Im());
    System.out.println("moduo x = " + x.moduo());
    System.out.println("konjugovano x = " + x.konjugovanBroj());
    System.out.println("x + konjugovano x = " +
        KompleksanBroj.zbir(x, x.konjugovanBroj()));

    System.out.println();

    KompleksanBroj y = new KompleksanBroj(0, 1);

    System.out.println("y = " + y);
    System.out.println("Re y = " + y.Re());
    System.out.println("Im y = " + y.Im());
    System.out.println("moduo y = " + y.moduo());
```

```
        System.out.println("konjugovano y = " + y.konjugovanBroj());
        System.out.println("x + y = " + x.dodaj(y));
        System.out.println("x * y = " + x.pomnozi(y));
    }
}
```

4.

LISTING 6.4: RimskiBroj.java

```
public class RimskiBroj {

    private int n; // celobrojna decimalna reprezentacija

    // Konstruktori
    public RimskiBroj(int n) {
        this.n = n;
    }

    public RimskiBroj(String r) {
        for (int i = 0; i < r.length(); i++)
            switch(r.charAt(i)) {
                case 'm': case 'M': n = n + 1000; break;
                case 'd': case 'D': n = n + 500; break;
                case 'c': case 'C': n = n + 100; break;
                case 'l': case 'L': n = n + 50; break;
                case 'x': case 'X': n = n + 10; break;
                case 'v': case 'V': n = n + 5; break;
                case 'i': case 'I': n = n + 1; break;
            }
    }

    // Pretvaranje rimskog u decimalni broj
    public int toInt() {
        return n;
    }

    // Pretvaranje decimalnog u rimski broj
    public String toString() {
        return d2r(n);
    }

    // Pomoćni rekurzivni metod za pretvaranje
    // decimalnog broja u rimski broj
    private String d2r(int n) {
        if (n >= 1000)
            return "M" + d2r(n - 1000);
        else if (n >= 500)
```

```
        return "D" + d2r(n - 500);
    else if (n >= 100)
        return "C" + d2r(n - 100);
    else if (n >= 50)
        return "L" + d2r(n - 50);
    else if (n >= 10)
        return "X" + d2r(n - 10);
    else if (n >= 5)
        return "V" + d2r(n - 5);
    else if (n >= 1)
        return "I" + d2r(n - 1);
    else
        return "";
}
/*
// Pomoćni nerekurzivni metod za pretvaranje
// decimalnog broja u rimski broj
private String d2r(int n) {
    String r = "";
    while (n >= 1000) { r = r + "M"; n = n - 1000; }
    while (n >= 500) { r = r + "D"; n = n - 500; }
    while (n >= 100) { r = r + "C"; n = n - 100; }
    while (n >= 50) { r = r + "L"; n = n - 50; }
    while (n >= 10) { r = r + "X"; n = n - 10; }
    while (n >= 5) { r = r + "V"; n = n - 5; }
    while (n >= 1) { r = r + "I"; n = n - 1; }
    return r;
}
*/
// Statički metod za sabiranje dva rimska broja:
// RimskiBroj z = RimskiBroj.zbir(x, y)
public static RimskiBroj zbir(RimskiBroj a, RimskiBroj b) {
    return new RimskiBroj(a.n + b.n);
}

// Objektni metod za sabiranje dva kompleksna broja:
// RimskiBroj z = x.dodaj(y);
public RimskiBroj dodaj(RimskiBroj a) {
    return new RimskiBroj(this.n + a.n);
}

// Statički metod za proizvod dva rimska broja:
// RimskiBroj z = RimskiBroj.proizvod(x, y)
public static RimskiBroj proizvod(RimskiBroj a, RimskiBroj b) {
    return new RimskiBroj(a.n * b.n);
}
```

```
// Objektni metod za proizvod dva kompleksna broja:  
// RimskiBroj z = x.pomnoži(y);  
public RimskiBroj pomnoži(RimskiBroj a) {  
    return new RimskiBroj(this.n * a.n);  
}  
  
public static void main(String[] args) {  
  
    RimskiBroj x = new RimskiBroj("xxxiiii"); // 34  
    System.out.println("x = " + x.toInt());  
    System.out.println("x = " + x); // x.toString()  
  
    RimskiBroj y = new RimskiBroj("mdclxvi"); //1666  
    System.out.println("y = " + y.toInt());  
    System.out.println("y = " + y); // y.toString()  
  
    System.out.println();  
    System.out.println( // RimskiBroj.zbir(x, y).toString()  
        "x+y = " + RimskiBroj.zbir(x, y));  
    System.out.println(  
        "x*y = " + RimskiBroj.zbir(x, y).toInt());  
    System.out.println( // x.dodaj(y).toString()  
        "x+y = " + x.dodaj(y));  
    System.out.println(  
        "x*y = " + x.dodaj(y).toInt());  
  
    System.out.println();  
    System.out.println( // RimskiBroj.proizvod(x, y).toString()  
        "x*y = " + RimskiBroj.proizvod(x, y));  
    System.out.println(  
        "x*y = " + RimskiBroj.proizvod(x, y).toInt());  
    System.out.println( // x.pomnoži(y).toString()  
        "x*y = " + x.pomnoži(y));  
    System.out.println(  
        "x*y = " + x.pomnoži(y).toInt());  
}  
}
```

## 5.

LISTING 6.5: DveKocke2.java

```
public class DveKocke2 {  
  
    public static void main(String[] args) {  
  
        System.out.println("Zbir dve kocke      Prosečan broj bacanja");  
        System.out.println("-----      -----");
```

```
for (int i = 2; i < 13; i++)
    System.out.printf("%7d %24.2f\n", i, prosekZaZbir(i));
}

public static int baciZaZbir(int zbir) {

    int brojBacanja = 0; // brojač bacanja dve kocke
    KockaZaIgru kocka1 = new KockaZaIgru(); // prva kocka
    KockaZaIgru kocka2 = new KockaZaIgru(); // druga kocka

    do {
        kocka1.baci(); // baci prvu kocku
        kocka2.baci(); // baci drugu kocku
        brojBacanja++;
    } while ((kocka1.broj + kocka2.broj) != zbir);

    return brojBacanja;
}

public static double prosekZaZbir(int zbir) {

    final int BROJ_PONAVLJANJA = 100000;
    int ukupnoBacanja = 0; // ukupan broj bacanja za
                           // dati zbir dve kocke

    for (int i = 0; i < BROJ_PONAVLJANJA; i++)
        ukupnoBacanja = ukupnoBacanja + baciZaZbir(zbir);

    return (double)ukupnoBacanja/BROJ_PONAVLJANJA;
}
}

class KockaZaIgru {

    public int broj; // broj koji je pao

    public KockaZaIgru() { // konstruktor bez parametara
        baci(); // poziv metoda baci()
    }

    public KockaZaIgru(int n) { // konstruktor sa parametrom
        broj = n;
    }

    public void baci() { // „bacanje” kocke
```

```
        broj = (int)(Math.random()*6) + 1;
    }
}
```

6.

LISTING 6.6: PismoGlava.java

```
import java.util.*;

public class PismoGlava {

    public static void main(String[] args) {

        int brojBacanja; // broj bacanja novčića
        Brojač brojačPisama = new Brojač(); // broj palih pisama
        Brojač brojačGlava = new Brojač(); // broj palih glava
        Scanner tastatura = new Scanner(System.in);

        System.out.print("Unesite broj bacanja novčića: ");
        brojBacanja = tastatura.nextInt();

        while (brojBacanja > 0) {
            brojačPisama.reset();
            brojačGlava.reset();
            for (int i = 0; i < brojBacanja; i++)
                if (Math.random() < 0.5)
                    brojačPisama.uvećaj();
                else
                    brojačGlava.uvećaj();
            System.out.print("U " + brojBacanja + " bacanja, palo je ");
            System.out.println(brojačPisama.getBroj() + " pisama.");
            System.out.print("U " + brojBacanja + " bacanja, palo je ");
            System.out.println(brojačGlava.getBroj() + " glava.");
            System.out.println();
            System.out.print("Unesite broj bacanja novčića: ");
            brojBacanja = tastatura.nextInt();
        }
    }

    class Brojač {

        private int broj; // početna vrednost je 0

        public void uvećaj() {
            broj++;
        }
    }
}
```

```
public void reset() {  
    broj = 0;  
}  
  
public int getBroj() {  
    return broj;  
}  
}
```



# 7

## ***Osnovne strukture podataka***

### **Odgovori na pitanja**

1.  a[2]  
 a(2)  
 a[3]  
 a(3)
  
2.  int[] a = new int[2];  
 int[] a = new int(2);  
 int a = new int[2];  
 int a() = new int[2];
  
3.  i  
 (int)(Math.random() \* 100)  
 (int)(Math.random() \* 100 + 1)  
 Math.random() \* 100  
 i + 10  
 i + 6.5
  
4.  Program ima grešku, jer je dužina niza a premala.

- Program ima grešku, jer elementi niza a nisu inicijalizovani.
  - Program ima grešku, jer element a[0] nije definisan.
  - Program nema grešaka i normalno se izvršava prikazujući a[0] je 0 na ekranu.
5.  int i = new int(30);  
 double[] d = new double[30];  
 int[] i = {3, 4, 3, 2};  
 char[] c = new char();  
 char[] c = new char{'a', 'b', 'c', 'd'};  
 char[] c = {'a', 'b'};
6.  0  
 3  
 4  
 5
7.  Program prikazuje 0 1 2 3 4 na ekranu.  
 Program prikazuje 4 na ekranu.  
 Program ima grešku, jer će se koristiti nepostojeći element a[5] u poslednjoj naredbi print u metodu main.  
 Program ima grešku, jer promenljiva i u poslednjoj naredbi print u metodu main neće imati nijednu vrednost.
8.  Program prikazuje 120 200 16 na ekranu.  
 Program prikazuje 120 200 14 na ekranu.  
 Program prikazuje 120 200 22 na ekranu.  
 Program ima grešku, jer umesto 016 treba pisati 16.
9.  1 2 3  
 1 1 1  
 0 1 2  
 0 1 3

10.  1 2 3  
 1 1 1  
 0 1 2  
 0 1 3
11.  1 2 3 4  
 0 0  
 0 0 3 4  
 0 0 0 0
12.  1 2 3 4  
 0 0  
 0 0 3 4  
 0 0 0 0
13.  Program prikazuje 1 2 3 4 na ekranu.  
 Program prikazuje 0 0 na ekranu.  
 Program ima grešku kod naredbe `x = new int[2]`, jer je promenljiva `x` deklarisana da bude final i ne može se menjati.  
 Elementi niza `x` se ne mogu menjati, jer je promenljiva `x` deklarisana da bude final.
14.  Programski fragment ima grešku, jer se promenljiva lista ne može menjati nakon što joj se dodeli vrednost.  
 Programski fragment nema grešaka i normalno se izvršava. Drugom naredbom se novi niz dodeljuje promenljivoj lista.  
 Programski fragment ima grešku, jer se promenljivoj lista dodeljuje novi niz.  
 Programski fragment ima grešku, jer se promenljivoj lista dodeljuje novi niz različite dužine od prvog.
15.  Program ima grešku kod naredbe `a = new int[2]`, jer se novi niz dodeljuje promenljivoj `a`.

- Program ima grešku kod naredbe `println`, jer `a[1]` nije inicijalizovan.
  - Program na ekranu prikazuje `a[1]` je 0.
  - Program na ekranu prikazuje `a[1]` je 1.
16.  `b = Arrays.copyOf(a, a.length);`
- `b = Arrays.copyOf(a);`
  - `Arrays.copyOf(b, a, a.length);`
  - `Arrays.copyOf(a, b);`
17.  kopija datog niza
- kopija prvog elementa datog niza
  - dužina datog niza
  - referenca na dati niz
18.  Poruka o greški.
- 1 1
  - 2 2
  - 2 1
  - 1 2
19.  1 2 3 4 5
- 5 4 3 2 1
  - 5 4 1 2 3
  - 1 2 5 4 3
20.  Program ima grešku, jer je nepravilan argument `new double[]{3, 3}` u prvom pozivu metoda `xMetod()`.
- Program ima grešku, jer je nepravilan argument `new double[5]` u drugom pozivu metoda `xMetod()`.
  - Program ima grešku, jer je nepravilan argument `new double[3]{1, 2, 3}` u trećem pozivu metoda `xMetod()`.
  - Program ima grešku, jer će sve vrednosti niza a imati vrednost null prilikom izvršavanja drugog poziva metoda `xMetod()`.

21.  stek memorija  
 hip memorija  
 keš memorija  
 virtuelna memorija
22.  kopija tog niza  
 kopija prvog elementa tog niza  
 dužina tog niza  
 referenca na taj niz
23.  return 1;  
 return {1, 2, 3};  
 return int[]{1, 2, 3};  
 return new int[]{1, 2, 3};
24.  1 2 3 4 5  
 5 4 3 2 1  
 5 4 1 2 3  
 1 2 5 4 3
25.  1 2 3 4 5  
 5 4 3 2 1  
 5 4 1 2 3  
 1 2 5 4 3
26.  Promenljiva k sadrži niz od 10 celobrojnih vrednosti.  
 Promenljiva k sadrži niz od 10 objekata klase Krug.  
 Promenljiva k sadrži referencu na niz od 10 promenljivih klasnog tipa Krug.  
 Promenljiva k sadrži objekat klase Krug prečnika 10.
27.  args[0]  
 args[1]

- args[2]  
 args[3]
28.  public void prikaži(String... niska, double... broj)  
 public void prikaži(double... broj, String ime)  
 public void double... prikaži(double d1, double d2)  
 public void prikaži(double... broj)  
 public void prikaži(int n, double... broj)
29.  Program ima grešku, jer je nepravilan poziv prosek(d) u prvoj naredbi println.  
 Program ima grešku, jer je nepravilan poziv prosek(1, 2, 2, 1, 4) u drugoj naredbi println.  
 Program ima grešku, jer je nepravilan poziv prosek(new double[]{1, 2, 3}) u trećoj naredbi println.  
 Program ima grešku, jer je nepravilan poziv prosek(1.0, 2.0, 2.0, 1.0) u četvrtoj naredbi println.  
 Program se izvršava bez greške i prosek datih brojeva se tačno izračunava.  
 Program se izvršava bez greške, ali se prosek datih brojeva ne izračunava tačno.
30.  Arrays(lotoBrojevi)  
 Arrays.sort(lotoBrojevi)  
 Arrays.sorts(lotoBrojevi)  
 Arrays.sortArray(lotoBrojevi)
31.  0  
 -1  
 1  
 2  
 -2
32.  char[][] z = {'a', 'b'};

- char[2][2] z = {{'a', 'b'}, {'c', 'd'}};
  - char[2][] z = {{'a', 'b'}, {'c', 'd'}};
  - char[][] z = {{'a', 'b'}, {'c', 'd'}};
33.  4 i 4  
 4 i 5  
 5 i 4  
 5 i 5
34.  Program ima grešku, jer je new boolean[3][] nepravilno.  
 Program ima grešku, jer će x[2][2] imati vrednost null.  
 Program se normalno izvršava i na ekranu se prikazuje x[2][2] je null.  
 Program se normalno izvršava i na ekranu se prikazuje x[2][2] je false.

## Rešenja zadataka

1.

LISTING 7.1: Sito.java

```
/*
 * Program prikazuje niz svih prostih brojeva manjih od dateg broja m
 * koristeći postupak Eratostenovog sita: redom isključiti proizvode
 * svih prostih brojeva manjih od kvadratnog korena od m, a oni brojevi
 * koji preostanu su prosti. Granica niza prostih brojeva m dobija se
 * preko komandnog reda.
 */
public class Sito {

    public static void main(String[] args) {

        if (args.length == 0) {
            System.out.print("Granica niza prostih brojeva ");
            System.out.println("nije navedena u komandnom redu!");
            System.exit(-1);
        }
        int m = Integer.parseInt(args[0]);

        // Logički niz koji ukazuje da li su brojevi manji od m
        // (indeksi elemenata tog niza) prosti ili ne
        boolean[] prostBroj = new boolean[m];

        // Na početku se pretpostavlja da su svi brojevi prosti,
        // dok se ne otkrije suprotno
        for (int i = 0; i < m; i++)
            prostBroj[i] = true;

        // Za određivanje svih prostih brojeva manjih od m, treba
        // isključiti proizvode svih brojeva manjih od kvadratnog
        // korena od m
        int n = (int) Math.ceil(Math.sqrt(m));

        // Za svaki ceo broj i od 2 do n:
        // Ako i jeste prost, onda svi njegovi proizvodi nisu prosti,
        // pa ih treba isključiti u nizu prostBroj.
        // Ako i nije prost, onda su njegovi proizvodi već isključeni
        // nekim manjim prostim faktorom broja i, pa ovaj slučaj
        // treba zanemariti.
        for (int i = 2; i < n; i++) {
            if (prostBroj[i])
                for (int j = 2*i; j < m; j = j + i)
                    prostBroj[j] = false;
        }
    }
}
```

```
// Prikazivanje niza prostih brojeva manjih od m po 10 u redu
System.out.println("Niz prostih brojeva manjih od " + m + ":");

int j = 0;
for (int i = 2; i < m; i++) {
    if (prostBroj[i]) {
        System.out.print(i + " ");
        j++;
        if (j == 10) {
            System.out.println();
            j = 0;
        }
    }
}
}
```

2.

LISTING 7.2: IgraŽivota.java

```
import java.util.*;

public class IgraŽivota {

    public static void main(String[] args) {

        System.out.println("Ovo je igra života!\n");

        System.out.print("Unesite veličinu (broj vrsta ");
        System.out.print("i kolona) kolonije: ");

        Scanner tastatura = new Scanner(System.in);
        int n = tastatura.nextInt();
        Kolonija kol = new Kolonija(n);

        System.out.print("Unesite broj organizama na početku: ");
        int brojOrganizama = tastatura.nextInt();
        System.out.print("Unesite vrste i kolone ");
        System.out.println("organizama na početku - ");
        for (int i = 0; i < brojOrganizama; i++) {
            System.out.print("Organizam " + (i+1) + ": ");
            int v = tastatura.nextInt();
            int k = tastatura.nextInt();
            kol.zauzmiĆeliju(v,k);
        }

        System.out.println();
        int g = 0;
```

```
while (true) {
    System.out.println("Generacija " + g + ": ");
    kol.prikaži();
    System.out.print("Sledeća generacija (d/n)? ");
    String novaGen = tastatura.next();
    if (novaGen.equals("n")) break;
    kol.novaGen();
    g++;
}
}

class Kolonija {

    private int n;           // veličina kolonije
    private boolean[][] m;   // matrica čelija:
                            //     true  = zauzeta čelija
                            //     false = prazna čelija
    // Konstruktor
    public Kolonija(int n) {
        this.n = n;
        m = new boolean[n+2][n+2]; // prazne čelije oko ivice matrice
    }

    public void zauzmiČeliju(int i, int j) {
        m[i][j] = true;
    }

    public void novaGen() {

        boolean[][] m1 = new boolean[n+2][n+2]; // novo stanje čelija
                                                // broj suseda čelije

        for (int i = 1; i <= n; i++)
            for (int j = 1; j <= n; j++) {
                // Određivanje broja suseda čelije m[i][j]
                bs = 0;
                for (int k = i - 1; k <= i + 1; k++)
                    for (int l = j - 1; l <= j + 1; l++)
                        if (m[k][l] == true)
                            bs++;
                bs = (m[i][j] == true) ? bs - 1 : bs;

                // Određivanje novog stanja čelije m[i][j]
                if (m[i][j] == false)
                    if (bs == 3)
                        m1[i][j] = true;
    }
}
```

```
        else
            m1[i][j] = false;
        else
            if (bs < 2 || bs > 3)
                m1[i][j] = false;
            else
                m1[i][j] = true;
        }
    m = m1;
}

public void prikaži() {
    for (int i = 1; i <= n; i++) {
        for (int j = 1; j <= n; j++)
            System.out.print(m[i][j] ? "*" : " ");
        System.out.println();
    }
}
```

3.

LISTING 7.3: KešMemorija.java

```
import java.util.*;

public class KešMemorija {

    private ElementKeša[] keš;      // keš-memorija
    private int n; // broj elemenata keš-memorije

    // Konstruktor
    public KesMemorija(int veličina) {
        keš = new ElementKeša[veličina];
    }

    public void add(ElementKeša noviElem) {

        ElementKeša elem;
        int i;

        // Da li je element već u kešu?
        for (i = 0; i < n; i++) {
            elem = keš[i];
            if (noviElem.equals(elem)) {
                noviElem = elem;
                break;
            }
        }
    }
}
```

```
}

if (i == n) // element nije nađen
    if (n == keš.length) // i keš je pun
        i--;
    else
        n++;

// Napraviti mesto na početku niza
for (int j = i; j > 0; j--)
    keš[j] = keš[j - 1];

keš[0] = noviElem;
}

// Najskorije korišćen (prvi) element keša
public ElementKeša NKE() {
    return keš[0];
}

public void prikaži() {
    for (int i = 0; i < n; i++)
        System.out.println(keš[i]);
}

public void prikažiSve() {
    for (ElementKeša elem : keš)
        System.out.println(elem);
}

// „Klijentska strana” za testiranje
public static void main(String[] args) {

    KesMemorija kešReči = new KesMemorija(5);

    kešReči.add(new ElementKeša("Pera"));
    kešReči.add(new ElementKeša("Zika"));
    kešReči.add(new ElementKeša("Laza"));
    kešReči.add(new ElementKeša("Zika"));
    kešReči.add(new ElementKeša("Mika"));
    kešReči.add(new ElementKeša("Mika"));
    kešReči.add(new ElementKeša("Pera"));
    kešReči.add(new ElementKeša("Sava"));
    kešReči.add(new ElementKeša("Deki"));
    kešReči.add(new ElementKeša("Zika"));
    kešReči.add(new ElementKeša("Zika"));
```

```
System.out.println("Sadržaj keša:");
kešReči.prikaži();

System.out.println();
kešReči.prikažiSve();
}

}

class ElementKeša {

    private String sadržaj; // sadržaj elementa keša

    // Konstruktor
    public ElementKeša(String s) {
        sadržaj = s;
    }

    public String getSadržaj() {
        return sadržaj;
    }

    public boolean equals(Object o) {
        ElementKeša drugiElement = (ElementKeša) o;
        return sadržaj.equals(drugiElement.sadržaj);
    }

    public String toString() {
        return "element keša: " + sadržaj;
    }
}
```

4.

LISTING 7.4: ČesteReči.java

```
import java.util.*;
import java.io.*;

public class ČesteReči {

    // main() može da proizvede izuzetak FileNotFoundException u
    // slučaju otvaranja nepostojeće datoteke sa ulaznim tekstom
    public static void main(String[] args) throws IOException {

        int brojReči = 0;
        int brojČestihReči = 0;
        KešMemorija kešReči = new KešMemorija(30);
```

```
System.out.println("Lista često ponavljanih reči");
System.out.println("-----");

Scanner dokument = new Scanner(new File("tekst.txt"));
while (dokument.hasNext()) {
    String reč = dokument.next();
    brojReči++;
    kešReči.add(new ElementKeša(reč.toLowerCase()));
    int r = kešReči.NKE().getRef();
    if (r > 1) {
        brojČestihReči++;
        System.out.println(reč + " (" + r + ")");
    }
}
System.out.println("-----");
System.out.println(
    "Ukupan broj reči u tekstu: " + brojReči);
System.out.println(
    "Broj često ponavljanih reči: " + brojČestihReči);
}

}

class KešMemorija {

    private ElementKeša[] keš;      // keš-memorija
    private int n; // broj elemenata keš-memorije

    // Konstruktor
    public KešMemorija(int veličina) {
        keš = new ElementKeša[veličina];
    }

    public void add(ElementKeša noviElem) {

        ElementKeša elem;
        int i;

        // Da li je element već u kešu?
        for (i = 0; i < n; i++) {
            elem = keš[i];
            if (noviElem.equals(elem)) {
                noviElem = elem;
                break;
            }
        }

        if (i == n) // element nije nađen
    }
}
```

```
if (n == keš.length) // i keš je pun
    i--; // ukloniti poslednji element keša
else
    n++;

// Napraviti mesto na početku niza
for (int j = i; j > 0; j--)
    keš[j] = keš[j - 1];

noviElem.uvećajRef();
keš[0] = noviElem;
}

// Najskorije korišćen (prvi) element keša
public ElementKeša NKE() {
    return keš[0];
}

public void prikaži() {
    for (int i = 0; i < n; i++) {
        System.out.print(keš[i].getSadržaj());
        System.out.println(" (" + keš[i].getRef() + ")");
    }
}
}

class ElementKeša {

private String sadržaj; // sadržaj elementa keša
private int ref; // broj ponavljanja (referenci)

// Konstruktor (reč je sadržaj elementa)
public ElementKeša(String reč) {
    sadržaj = reč;
}

public String getSadržaj() {
    return sadržaj;
}

public int getRef() {
    return ref;
}

public void uvećajRef() {
    ref++;
}
```

```
public boolean equals(Object o) {
    ElementKeša drugiElement = (ElementKeša) o;
    return sadržaj.equals(drugiElement.sadržaj);
}

public String toString() {
    return "element keša: " + sadržaj;
}
}
```

5.

LISTING 7.5: DomZdravlja.java

```
import java.util.*;

public class DomZdravlja {

    private static final int NOVI_PACIJENT = 1;
    private static final int ZAVRŠEN_PREGLED = 2;
    private static final int LISTA_PACIJENATA = 3;
    private static final int LISTA_DOKTORA = 4;
    private static final int LISTA_PREGLEDA = 5;
    private static final int KRAJ_RADA = 6;

    private String imeDZ;
    private ArrayList<Doktor> listaDoktora;
    private ArrayList<Pacijent> listaPacijenata;
    private ArrayList<Pregled> listaPregleda;

    // Konstruktor
    public DomZdravlja(String imeDZ) {

        this.imeDZ = imeDZ;
        listaDoktora = new ArrayList<Doktor>();
        listaPacijenata = new ArrayList<Pacijent>();
        listaPregleda = new ArrayList<Pregled>();
    }

    public void otvoriRecepciju(Scanner t) {

        String ime, id, spec;
        String još;

        System.out.println("    Dom zdravlja \'" + imeDZ + "\'");
        System.out.println("\nUnesite listu doktora u smeni:");
        System.out.println();
    }
}
```

```
do {
    System.out.print("        Ime doktora: "); ime = t.nextLine();
    System.out.print("        ID doktora : "); id = t.nextLine();
    System.out.print("Specijalnost doktora: "); spec = t.nextLine();

    Doktor dok = new Doktor(ime, id, spec);
    listaDoktora.add(dok);
    System.out.print("\nSledeći doktor (d/n)? ");
    još = t.nextLine();
} while (još.toUpperCase().equals("D"));

}

public void otvoriČekaonicu(Scanner t) {

    int opcija;
    do {

        prikažiOpcije();

        //      System.out.print("\f");
        opcija = t.nextInt();
        t.nextLine(); // preskoči znak za novi red

        switch (opcija) {
            case NOVI_PACIJENT:
                dodajPacijenta(t);
                break;
            case ZAVRŠEN_PREGLED:
                završiPregled(t);
                break;
            case LISTA_PACIJENATA:
                prikažiPacijente();
                break;
            case LISTA_DOKTORA:
                prikažiDoktore();
                break;
            case LISTA_PREGLEDA:
                prikažiPreglede();
                break;
            case KRAJ_RADA:
                break;
            default:
                System.out.println("\nIzabrali ste pogrešnu opciju!");
        }
    } while (opcija != KRAJ_RADA);
```

```
}

private void prikažiOpcije() {

    // Prikazivanje menija na ekranu
    System.out.println();
    System.out.println("Opcije menija su:");
    System.out.println("    1. Novi pacijent u čekaonici");
    System.out.println("    2. Završen pregled pacijenta");
    System.out.println("    3. Lista pacijenata u čekaonici");
    System.out.println("    4. Lista slobodnih doktora");
    System.out.println("    5. Lista trenutnih pregleda");
    System.out.println("    6. Kraj rada");
    System.out.print("Unesite broj opcije: ");
}

private void prikažiDoktore() {

    int n = listaDoktora.size();
    if (n == 0) {
        System.out.println();
        System.out.println("\nNema slobodnih doktora ...");
    }
    else {
        System.out.println("\nSlobodni doktori:");
        for (int i = 0; i < n; i++)
            listaDoktora.get(i).prikaži();
    }
}

private void prikažiPacijente() {

    int n = listaPacijenata.size();
    if (n == 0) {
        System.out.println();
        System.out.println("\nNema pacijenata koji čekaju ...");
    }
    else {
        System.out.println("\nPacijenti koji čekaju:");
        for (int i = 0; i < n; i++)
            listaPacijenata.get(i).prikaži();
    }
}

private void prikažiPreglede() {

    int n = listaPregleda.size();
```

```
if (n == 0) {
    System.out.println();
    System.out.println("\nNema pregleda koji su u toku ...");
}
else {
    System.out.println("\nPregledi koji su u toku:");
    for (int i = 0; i < n; i++)
        listaPregleda.get(i).prikaži();
}
}

private void dodajPacijenta(Scanner t) {

String ime, jmb, simptom, mob;

System.out.println();
System.out.print("      Ime pacijenta: "); ime = t.nextLine();
System.out.print("      JMB pacijenta: "); jmb = t.nextLine();
System.out.print("      Simptomi bolesti: "); simptom = t.nextLine();
System.out.print("Med. oblast bolesti: "); mob = t.nextLine();

Pacijent pac = new Pacijent(ime, jmb, simptom, mob);
listaPacijenata.add(pac);
proveriČekaonicu();
}

private void završiPregled(Scanner t) {

String id;

System.out.println();
System.out.print("\nID doktora kod koga je završen pregled: ");
id = t.nextLine();

// Traženje doktora u listi pregleda
for (int i = 0; i < listaPregleda.size(); i++) {
    Pregled preg = listaPregleda.get(i);
    Doktor dok = preg.getDok();
    if (id.equals(dok.getID())) {
        Pacijent pac = preg.getPac();
        listaPacijenata.remove(pac);
        listaDoktora.add(dok);
        System.out.println("\nDoktor " + id + " je slobodan.");
        listaPregleda.remove(preg);
        proveriČekaonicu();
        return;
    }
}
```

```
        }
        System.out.print("\nGreška: doktor ");
        System.out.println(id + " nije nadjen u listi pregleda.");
    }

private void proveriČekaonicu() {

    // Provera pacijenata koji čekaju
    for (int i = 0; i < listaPacijenata.size(); i++) {
        Pacijent pac = listaPacijenata.get(i);
        for (int j = 0; j < listaDoktora.size(); j++) {
            Doktor dok = listaDoktora.get(j);
            if (pac.getMob().equals(dok.getSpec())) {
                Pregled preg = new Pregled(dok, pac);
                listaPregleda.add(preg);
                listaPacijenata.remove(pac);
                i--; // pacijent uklonjen iz liste pacijenata
                listaDoktora.remove(dok);
                System.out.println("\nNovi pregled: ");
                preg.prikaži();
                break;
            }
        }
    }

    // „Klijentska strana“ za testiranje
    public static void main(String[] args) {

        Scanner tastatura = new Scanner(System.in);

        DomZdravlja dz = new DomZdravlja("Beograd");
        dz.otvoriRecepцију(tastatura);
        dz.otvoriČekaonicu(tastatura);
    }
}

class Pacijent {

    private String ime;
    private String jmb;
    private String simptom;
    private String mob; // trijaža: medicinska oblast bolesti

    public Pacijent(String ime, String jmb, String simptom, String mob) {
        this.ime = ime;
        this.jmb = jmb;
```

```
    this.simptom = simptom;
    this.mob = mob;
}

public String getIme () {
    return ime;
}

public String getMob () {
    return mob;
}

public void prikaži () {
    System.out.printf("\n-----\n");
    System.out.printf("      Ime pacijenta: %s\n", ime);
    System.out.printf("      JMB pacijenta: %s\n", jmb);
    System.out.printf("      Simptomi bolesti: %s\n", simptom);
    System.out.printf("Med. oblast bolesti: %s\n", mob);
}
}

class Doktor {

    private String ime;
    private String id;
    private String spec;

    public Doktor(String ime, String id, String spec) {
        this.ime = ime;
        this.id = id;
        this.spec = spec;
    }

    public String getIme () {
        return ime;
    }

    public String getID () {
        return id;
    }

    public String getSpec () {
        return spec;
    }

    public void prikaži () {
        System.out.printf("\n-----\n");
    }
}
```

```
        System.out.printf("      Ime doktora: %s\n", ime);
        System.out.printf("      ID doktora: %s\n", id);
        System.out.printf("Specijalnost doktora: %s\n", spec);
    }
}

class Pregled {

    private Doktor dok;
    private Pacijent pac;

    public Pregled(Doktor dok, Pacijent pac) {
        this.dok = dok;
        this.pac = pac;
    }

    public Doktor getDok () {
        return dok;
    }

    public Pacijent getPac () {
        return pac;
    }

    public void prikaži () {
        dok.prikaži();
        pac.prikaži();
    }
}
```

# 8

## ***Nasleđivanje klasa***

### **Odgovori na pitanja**

1.  enkapsulacija  
 nasleđivanje  
 polimorfizam  
 apstrakcija
  
2.  Jedna klasa u Javi može *direktno* proširivati više klasa.  
 Proširena klasa sadrži dodatna polja i metode u odnosu na svoju nasleđenu klasu.  
 „Klasa A nasleđuje klasu B“ znači da je A potklasa od B.  
 Ako klasa A nasleđuje klasu B, tada objekti klase A sadrže sva polja i sve metode klase B.  
 Ako klasa A nasleđuje klasu B, tada se svaki objekat klase A podrazumeva da je i objekat klase B.
  
3.  Greške su u redovima 7, 10 i 14.  
 Greške su u redovima 7, 9 i 25.  
 Greške su u redovima 6, 7 i 10.  
 Greške su u redovima 6, 9 i 10.

- Greške su u redovima 6, 9 i 29.
  - Greške su u redovima 9, 10 i 29.
4.  Program ima grešku, jer klasa Test nema podrazumevani konstruktor Test().
- Program ima grešku, jer klasa Test ima implicitni podrazumevani konstruktor bez parametara Test(), ali nasleđena klasa A nema takav konstruktor. Program bi radio bez greške ukoliko bi se uklonio konstruktor u klasi A.
- Program ima grešku, ali bi radio bez greške ukoliko bi se klasi A eksplicitno dodao konstruktor bez parametara A().
5.  Ništa.
- Pozvan podrazumevani konstruktor klase A
  - Pozvan podrazumevani konstruktor klase B
  - Pozvan podrazumevani konstruktor klase A i u drugom redu Pozvan podrazumevani konstruktor klase B
  - Pozvan podrazumevani konstruktor klase B i u drugom redu Pozvan podrazumevani konstruktor klase A
6.  Službena reč super može poslužiti za pozivanje konstruktora nasleđene klase.
- Službena reč super može poslužiti za pozivanje zaklonjenog metoda nasleđene klase.
- Službena reč super ne može poslužiti za pozivanje zaklonjenog polja nasleđene klase.
7.  Program ima grešku, jer je metod m() nadjačan sa različitim potpisom u klasi B.
- Program ima grešku, jer se b.m(5) ne može pozvati pošto je metod m(int) zaklonjen u klasi B.
- Program ima grešku kod b.i, jer je polje i nepristupačno iz klase B.
- Metod m() nije nadjačan u klasi B. Klasa B nasleđuje metod m(int) od klase A i u B se definiše preopterećen metod m(String).

8.  Metod se može preopteretiti u istoj klasi.  
 Metod se može nadjačati u istoj klasi.  
 Ako su metodi preopterećeni, njihovi potpisi moraju biti isti.  
 Ako jedan metod nadjačava drugi metod, njihovi potpisi moraju biti isti.
9.  Jedna forma polimorfizma u Javi je princip podtipa po kojem promenljiva klasnog tipa može sadržati reference na objekte svog deklarisanog tipa i svakog njegovog podtipa.  
 Objekat tipa B se može preneti kao argument metodu na mesto paramatera tipa A ukoliko je B klasa naslednica od A.  
 Za razrešavanje poziva preopterećenih metoda se primenjuje statičko vezivanje u fazi prevođenja programa.  
 Za razrešavanje poziva nadjačanih metoda se primenjuje dinamičko vezivanje u fazi izvršavanja programa.
10.  Izvršavanjem programa se na ekranu ništa ne prikazuje.  
 Izvršavanjem programa se na ekranu prikazuje tekst: Pobednik je Crvena zvezda.  
 Izvršavanjem programa se na ekranu prikazuje tekst: Pobednik je Crvena zvezda (Beograd).  
 Izvršavanjem programa se na ekranu prikazuje tekst: Pobednik je Partizan.  
 Izvršavanjem programa se na ekranu prikazuje tekst: Pobednik je Partizan (Beograd).  
 Izvršavanjem program se na ekranu prikazuje tekst: Nerešeno.
11.  Program ima sintaksnu grešku, jer naredbu `System.out.println(a)` treba zameniti naredbom `System.out.println(a.toString())`.  
 Program nema greške i poziva se metod `toString()` u klasi `Object` prilikom izvršavanja naredbe `System.out.println(a)`.  
 Program nema greške i poziva se metod `toString()` u klasi A prilikom izvršavanja naredbe `System.out.println(a)`.  
 Program nema greške i poziva se metod `toString()` u klasi Object prilikom izvršavanja naredbe `System.out.println(o)`.

12.  public boolean equals(String s)  
 public boolean equals(Object o)  
 public static boolean equals(Object o)  
 public boolean equals(String s1, String s2)
13.  Program ima grešku, jer se izrazom a1.equals(a2) proverava jednakost objekata tipa različitog od Object.  
 Program se izvršava bez greške i prikazuje se true na ekranu.  
 Program se izvršava bez greške i prikazuje se false na ekranu.
14.  Program ima grešku, jer se izrazom a1.equals(a2) proverava jednakost objekata tipa različitog od Object.  
 Program se izvršava bez greške i prikazuje se true na ekranu.  
 Program se izvršava bez greške i prikazuje se false na ekranu.
15.  Program ima grešku, jer se izrazom a1.equals(a2) proverava jednakost objekata tipa različitog od Object.  
 Program se izvršava bez greške i prikazuje se true na ekranu.  
 Program se izvršava bez greške i prikazuje se false na ekranu.
16.  Greška je u pozivu m(new MasterStudent()).  
 Greška je u pozivu m(new Student()).  
 Greška je u pozivu m(new Osoba()).  
 Greška je u pozivu m(new Object()).
17.  Rezultat izraza c1 instanceof C1 je true.  
 Rezultat izraza c2 instanceof C1 je true.  
 Rezultat izraza c3 instanceof C1 je true.  
 Rezultat izraza c4 instanceof C2 je true.
18.  Kada se promenljiva s dodeljuje promenljivoj o u naredbi Object o = s, konstruiše se novi objekat.

- Kada se konvertuje tip promenljiva o i njena vrednost dodeljuje promenljivoj t u naredbi String t = (String)o, konstruiše se novi objekat.
  - Kada se konvertuje tip promenljiva o i njena vrednost dodeljuje promenljivoj t u naredbi String t = (String)o, sadržaj promenljive o se menja.
  - Promenljive s, o i t ukazuju na isti objekat tipa String.
19.  x = new char[100];  
 x = new int[100];  
 x = new double[100];  
 x = new String[100];  
 x = new java.util.Date[100];
20.  new ArrayList[]  
 new ArrayList[100]  
 new ArrayList()  
 ArrayList()
21.  x.remove("C++")  
 x.remove(0)  
 x.remove(1)  
 x.remove(2)
22.  [Java, Pascal]  
 [Java, C++, Pascal]  
 [Pascal, Java, C++]  
 [Java, Pascal, C++]
23.  x.first()  
 x.get(0)  
 x.get(1)  
 x.get()

24.  `x.getSize()`  
 `x.getLength(0)`  
 `x.length(1)`  
 `x.size()`
25.  `class A { }`  
 `class A { private A(){ } }`  
 `final class A { }`  
 `class A { protected A(){ } }`



## Rešenja zadataka

1.

LISTING 8.1: PovezanaLista.java

```
public class PovezanaLista {  
  
    private ElementListe prvi;    // prvi element liste  
    private ElementListe posl;    // poslednji element liste  
    private int n;                // broj elemenata liste  
  
    // Podrazumevani konstruktor za formiranje prazne liste  
    public PovezanaLista() {}  
  
    // Dužina liste  
    public int dužina() {  
        return n;  
    }  
  
    // Ispitivanje da li je lista prazna  
    public boolean praznaLista() {  
        return prvi == null;  
    }  
  
    // Dodavanje objekta na kraj liste  
    public void dodaj(Object o) {  
  
        // Konstruisati novi element liste  
        ElementListe noviElem = new ElementListe(o);  
  
        if (praznaLista())  
            prvi = posl = noviElem;  
        else {  
            posl.setSled(noviElem);  
            posl = noviElem;  
        }  
        n++;  
    }  
  
    // Ispitivanje da li je dati objekat u listi  
    public boolean nađi(Object o) {  
        ElementListe elem;  
        for (elem = prvi; elem != null; elem = elem.getSled()) {  
            if (elem.getSadržaj().equals(o))  
                break;  
        }  
        return (elem != null);  
    }  
}
```

```
// String reprezentacija povezane liste
public String toString() {
    String s = "";
    for (ElementListe elem = prvi; elem != posl;
                     elem = elem.getSled()) {
        s = s + elem.toString() + ", ";
    }
    if (posl != null)
        s = s + posl.toString();
    return s;
}

// „Klijentska strana“ klase radi testiranja
public static void main(String[] args) {

    // Konstruisanje prazne liste
    PovezanaLista lis = new PovezanaLista();

    // Dodavanje nekih elemenata u listu
    Object o = new Integer(17);
    lis.dodaj(o);
    lis.dodaj(23); lis.dodaj(31); lis.dodaj(47);

    // Prikazivanje elemenata liste
    System.out.println("Sadržaj liste:");
    System.out.println(lis);
    System.out.println("Dužina liste: " + lis.duzina());

    // Traženje objekta u listi
    int x = 23;
    System.out.println();
    System.out.print("Objekat " + x + " se ");
    if (lis.nadi((Integer)x) == false)
        System.out.print("ne ");
    System.out.println("nalazi u listi");
}
}

class ElementListe {

    private Object sadrzaj;    // sadržaj elementa liste
    private ElementListe sled; // pokazivač na sledeći
                               // element liste
    // Konstruktor
    public ElementListe(Object o) {
        sadrzaj = o;
    }
}
```

```
}

// Pristup poljima elementa liste
public void setSled(ElementLista elem) {
    sled = elem;
}
public ElementLista getSled() {
    return sled;
}

public Object getSadržaj() {
    return sadržaj;
}

// String reprezentacija elementa liste
public String toString() {
    return sadržaj.toString();
}
}
```

2.

LISTING 8.2: PoliLinija.java

```
public class PoliLinija {

    private PovezanaLista listaTemenा; // povezana lista temena

    // Konstruisanje poligonalne linije od
    // niza koordinata tačaka temena
    public PoliLinija(double[][] koordTemenа) {

        listaTemenа = new PovezanaLista();

        for (int i = 0; i < koordTemenа.length; i++) {
            Tačka t = new Tačka(koordTemenа[i][0], koordTemenа[i][1]);
            listaTemenа.dodaj(t);
        }
    }

    // Konstruisanje poligonalne linije od
    // niza tačaka temena
    public PoliLinija(Tačka[] temena) {

        listaTemenа = new PovezanaLista();

        for (Tačka t : temena)
            listaTemenа.dodaj(t);
    }
}
```

```
}

// Dodavanje para koordinata tačke poligonalnoj liniji
public void dodajTeme(double x, double y) {
    listaTemenा.Dodaj(new Tačka(x, y));
}

// Dodavanje tačke poligonalnoj liniji
public void dodajTeme(Tačka t) {
    listaTemenा.Dodaj(t);
}

// Ispitivanje da li je tačka neko teme poligonalne linije
public boolean nađiTeme(Tačka t) {
    return listaTemenा.nađi(t);
}

public String toString() {
    return listaTemenा.toString();
}

// „Klijentska strana“ radi testiranja
public static void main(String[] args) {

    // Formiranje niza koordinata tačaka temena
    double[][] koordTemenा = {{2,2}, {1,2}, {-2,3},
        {3,-4}, {-1,-1}, {0,0}};
    PoliLinija pl = new PoliLinija(koordTemenा);
    System.out.println("Poligonalna linija:");
    System.out.println(pl);

    // Dodavanje tačkaka
    pl.dodajTeme(0,1);
    pl.dodajTeme(new Tačka(1,0));
    System.out.println("Poligonalna linija:");
    System.out.println(pl);

    // Ispitivanje da li je data tačka neko teme
    Tačka t = new Tačka(0,0);
    System.out.println();
    System.out.print("Tačka " + t + " ");
    if (pl.nadiTeme(t) == false)
        System.out.print("ni");
    System.out.println("je teme poligonalne linije");
}
}
```

```
class Tačka {  
  
    private double x, y; // koordinate tačke  
  
    // Konstruktor  
    public Tačka(double x, double y) {  
        this.x = x;  
        this.y = y;  
    }  
  
    public double getX() {  
        return x;  
    }  
  
    public double getY() {  
        return y;  
    }  
  
    public boolean equals(Object o) {  
        Tačka t = (Tačka)o;  
        return (this.x == t.x) && (this.y == t.y);  
    }  
  
    public String toString() {  
        return "(" + x + ", " + y + ")";  
    }  
}  
  
class PovezanaLista {  
  
    private ElementLista prvi; // prvi element liste  
    private ElementLista posl; // poslednji element liste  
    private int n; // broj elemenata liste  
  
    // Podrazumevani konstruktor za formiranje prazne liste  
    public PovezanaLista() {}  
  
    // Dužina liste  
    public int dužina() {  
        return n;  
    }  
  
    // Ispitivanje da li je lista prazna  
    public boolean praznaLista() {  
        return prvi == null;  
    }  
}
```

```
// Dodavanje objekta na kraj liste
public void dodaj(Object o) {

    // Konstruisati novi element liste
    ElementListe noviElem = new ElementListe(o);

    if (praznaLista())
        prvi = posl = noviElem;
    else {
        posl.setSled(noviElem);
        posl = noviElem;
    }
    n++;
}

// Ispitivanje da li je dati objekat u listi
public boolean nađi(Object o) {
    ElementListe elem;
    for (elem = prvi; elem != null; elem = elem.getSled()) {
        if (elem.getSadržaj().equals(o))
            break;
    }
    return (elem != null);
}

// String reprezentacija povezane liste
public String toString() {
    String s = "";
    for (ElementListe elem = prvi; elem != posl;
          elem = elem.getSled()) {
        s = s + elem.toString() + ", ";
    }
    if (posl != null)
        s = s + posl.toString();
    return s;
}

class ElementListe {

    private Object sadržaj;      // sadržaj elementa liste
    private ElementListe sled; // pokazivač na sledeći
                               // element liste
    // Konstruktor
    public ElementListe(Object o) {
        sadržaj = o;
    }
}
```

```
// Pristup poljima elementa liste
public void setSled(ElementListe elem) {
    sled = elem;
}
public ElementListe getSled() {
    return sled;
}

public Object getSadržaj() {
    return sadržaj;
}

// String reprezentacija elementa liste
public String toString() {
    return sadržaj.toString();
}
}
```

3.

LISTING 8.3: PovezanaLista1.java

```
public class PovezanaLista1 {

    private ElementListe prvi; // prvi element liste
    private ElementListe posl; // poslednji element liste
    private ElementListe tekući; // tekući element liste
    private int n; // broj elemenata liste

    // Podrazumevani konstruktor za formiranje prazne liste
    public PovezanaLista1() {}

    // Dužina liste
    public int dužina() {
        return n;
    }

    // Da li je lista prazna?
    public boolean praznaLista() {
        return n == 0;
    }

    // Pomeranje tekućeg elementa liste
    public ElementListe tekućiPrvi() {
        tekući = prvi;
        return tekući;
    }
}
```

```
public ElementListe tekućiSledeći() {
    if (tekući != posl) {
        tekući = tekući.getSled();
        return tekući;
    }
    else
        return null;
}

// Sadržaj tekućeg elementa liste
public Object tekućiSadržaj() {
    return tekući.getSadržaj();
}

// Dodavanje objekta na kraj liste
public void dodaj(Object o) {

    // Konstruisati novi element liste
    ElementListe noviElem = new ElementListe(o);

    if (praznaLista())
        prvi = posl = tekući = noviElem;
    else {
        posl.setSled(noviElem);
        posl = tekući = noviElem;
    }
    n++;
}

// Dodavanje objekta u tekuću poziciju liste
public void dodajTekući(Object o) {

    // Konstruisati novi element liste
    ElementListe noviElem = new ElementListe(o);

    // Odrediti prethodni element levo od tekućeg
    ElementListe pret = null;
    for (ElementListe elem = prvi; elem != tekući;
                     elem = elem.getSled())
        pret = elem;
    if (pret == null) { // dodati noviElem na prvu mesto
        noviElem.setSled(prvi);
        prvi = tekući = noviElem;
        if (dužina() == 0) // lista je bila prazna?
            posl = noviElem;
    }
}
```

```
    else {
        pret.setSled(noviElem);
        noviElem.setSled(tekući);
        tekući = noviElem;
    }
    n++;
}

// Uklanjanje tekućeg elementa iz liste
public void ukloniTkući() {
    if (praznaLista()) return;
    // Odrediti prethodni element levo od tekućeg
    ElementListe pret = null;
    for (ElementListe elem = prvi; elem != tekući;
         elem = elem.getSled())
        pret = elem;
    if (pret == null) { // tekući == prvi?
        prvi = prvi.getSled();
        tekući = prvi;
        if (dužina() == 1) // lista postaje prazna?
            posl = null;
    }
    else {
        ElementListe noviSled = tekući.getSled();
        if (noviSled == null) // tekući == posl?
            posl = tekući = pret;
        else
            tekući = noviSled;
        pret.setSled(noviSled);
    }
    n--;
}

// Ispitivanje da li je dati objekat u listi
public ElementListe nadi(Object o) {
    ElementListe elem;
    for (elem = prvi; elem != null; elem = elem.getSled()) {
        if (elem.getSadržaj().equals(o))
            break;
    }
    if (elem != null)
        tekući = elem;
    return elem;
}

// String reprezentacija povezane liste
public String toString() {
```

```
String s = "";
for (ElementListe elem = prvi; elem != posl;
         elem = elem.getSled()) {
    s = s + elem.toString() + ", ";
}
if (posl != null)
    s = s + posl.toString();
return s;
}

// „Klijentska strana“ klase radi testiranja
public static void main(String[] args) {

    // Konstruisanje prazne liste
    PovezanaLista1 lis = new PovezanaLista1();

    // Dodavanje nekih elemenata na kraj liste
    Object o = new Integer(17);
    lis.dodaj(o);
    lis.dodaj(23); lis.dodaj(31); lis.dodaj(47);

    // Prikazivanje elemenata liste
    System.out.println("Sadržaj liste:");
    System.out.println(lis);
    System.out.println("Dužina liste: " + lis.duzina());

    // Dodavanje nekih elemenata u tekuću poziciju
    lis.dodajTkući(51);
    lis.tekućiPrvi();
    lis.dodajTkući(3);
    lis.tekućiSledeći(); lis.tekućiSledeći();
    lis.dodajTkući(11);
    lis.dodaj(0);

    // Prikazivanje elemenata liste
    System.out.println("Sadržaj liste:");
    System.out.println(lis);
    System.out.println("Dužina liste: " + lis.duzina());

    // Uklanjanje tekućih elemenata
    lis.ukloniTkući();
    lis.tekućiPrvi();
    lis.ukloniTkući();
    lis.tekućiSledeći(); lis.tekućiSledeći();
    lis.ukloniTkući();

    // Prikazivanje elemenata liste
```

```
System.out.println("Sadržaj liste:");
System.out.println(lis);
System.out.println("Dužina liste: " + lis.dužina());

// Traženje objekta u listi
int x = 31;
System.out.println();
System.out.print("Objekat " + x + " se ");
if (lis.nađi((Integer)x) == null)
    System.out.print("ne ");
System.out.println("nalazi u listi");

lis.ukloniTekući();

// Prikazivanje elemenata liste
System.out.println("Sadržaj liste:");
System.out.println(lis);
System.out.println("Dužina liste: " + lis.dužina());
}

}

class ElementListe {

    private Object sadržaj; // sadržaj elementa liste
    private ElementListe sled; // pokazivač na sledeći
                               // element liste

    // Konstruktor
    public ElementListe(Object o) {
        sadržaj = o;
    }

    // Pristup poljima elementa liste
    public void setSled(ElementListe elem) {
        sled = elem;
    }
    public ElementListe getSled() {
        return sled;
    }

    public Object getSadržaj() {
        return sadržaj;
    }

    // String reprezentacija elementa liste
    public String toString() {
        return sadržaj.toString();
    }
}
```

}



# 9

## ***Posebne klase i interfejsi***

### **Odgovori na pitanja**

1.  Definicija nabrojivog tipa može stajati unutar nekog metoda.  
 Nabrojni tip mora biti član neke klase.  
 Vrednosti nabrojivog tipa su kodirani običnim celim brojevima 0, 1, 2 i tako dalje.  
 Za vrednosti nabrojivog tipa koristi se konvencija o imenima za obične konstante.  
 Nabrojni tip je specijalna vrsta ugnježđene klase.
  
2.  **public enum** osnovnaBoja {crvena, žuta, plava}  
 **public int[]** osnovnaBoja = {1, 2, 3}  
 **public enum** OsnovnaBoja {CRVENA, ŽUTA, PLAVA}  
 **public String[]** osnovnaBoja = {"crvena", "žuta", "plava"}  
 **public String** boja1 = "crvena", boja2 = "žuta", boja3 = "plava"
  
3.  SREDA je 3. dan u nedelji  
 Sreda je 3. dan u nedelji  
 Dan.SREDA je 3. dan u nedelji  
 SREDA je 2. dan u nedelji

- Sreda je 2. dan u nedelji
  - Dan.SREDA je 2. dan u nedelji
4.  SvetloSemafora.ZELENO
- ZELENO
  - zeleno
  - Kreni
  - Pazi
5.  class A { abstract void aMetod() { } }
- class A { abstract void aMetod(); }
  - abstract class A { abstract void aMetod(); }
  - public class abstract A { abstract void aMetod(); }
6.  public abstract aMetod();
- public abstract void aMetod();
  - public void abstract aMetod();
  - public void aMetod() { }
  - public abstract void aMetod() { }
7.  Moguće je konstruisati objekte apstraktne klase.
- Apstraktna klasa može biti tip neke objektne promenljive.
  - Klasa koja nasleđuje apstraktну klasu može biti apstraktna.
  - Klasa koja nasleđuje konkretnu klasu može biti apstraktna.
8.  Apstraktne klase ne mogu imati konstruktore.
- Klasa koja sadrži apstraktni metod mora biti apstraktna.
  - Moguće je definisati apstraktnu klasu koja ne sadrži nijedan apstraktni metod.
  - Apstraktne klase mogu imati konkretna polja i metode.
9.  A a = new A();
- A a = new B();

- B b = new A();  
 B b = new B();
10.  interface A { void print() { }; }
- abstract interface A { print(); }
- abstract interface A { abstract void print() { }; }
- interface A { void print(); }
11.  Ništa.
- b je instanca A
- b je instanca C
- b je instanca A i u drugom redu b je instanca C
12.  A a = new A();  
 A a = new B();  
 B b = new A();  
 B b = new B();
13.  Program ima grešku, jer klasa Voćka nema podrazumevani konstruktor.
- Program ima grešku kod dodeljivanja početnih vrednosti nizu voće.
- Program ima grešku kod sortiranja niza voće, jer klasa Voćka ne implementira interfejs Comparable i zato objekti klase Voćka nisu uporedivi.
- Program nema greške, ali se ništa ne prikazuje na ekranu.
14.  Klasa Radnik ima grešku, jer nema nijedan konstruktor.
- Klasa Radnik ima grešku, jer je metod max() definisan da bude static.
- Klasa Radnik ima grešku, jer tip vraćene vrednosti metoda max() treba da bude int.
- Klasa Radnik nema grešaka.

15.  Ugnježđene klase se mogu definisati kao članovi druge klase ili lokalno unutar metoda druge klase.
- Statičke i objektne ugnježđene klase se, kao i obične klase, mogu definisati samo sa modifikatorom public ili bez njega.
- Statička ugnježđena klasa ima potpun pristup statickim članovima obuhvatajuće klase, čak i ukoliko su oni deklarisani kao privatni.
- Objektna ugnježđena klasa je analogna objektnom polju ili objektnom metodu. Zbog toga je objektna ugnježđena klasa vezana za objekat obuhvatajuće klase.
16.  A.class  
 Test\$A.class  
 A\$Test.class  
 Test&A.class
17.  Objektne ugnježđene klase moraju implementirati neki interfejs.  
 Objektne ugnježđene klase se mogu instancirati samo unutar obuhvatajuće klase.  
 Objektne ugnježđene klase imaju pristup svim objektnim članovima obuhvatajuće klase, čak i ukoliko su oni deklarisani kao privatni.  
 Objektne ugnježđene klase se definišu kao članovi obuhvatajuće klase sa modifikatorom object.
18.  Anonimne klase su lokalne klase bez imena.  
 Anonimne klase uvek nasleđuju neku klasu ili implementiraju neki interfejs, ali ne sadrže eksplisitne reči extends ili implements.  
 Anonimne klase moraju implementirati sve apstraktne metode u nasleđenoj klasi ili interfejsu.  
 Anonimne klase uvek koriste podrazumevani konstruktor bez argumenta nasleđene klase radi konstruisanja svojih instanci.
19.  A.class  
 Test\$A.class  
 A\$Test.class  
 TestA.class  
 Test\$1.class

## Rešenja zadataka

1.

LISTING 9.1: Karta.java

```
public class Karta {

    private final VrednostKarte vrednost;
    private final BojaKarte boja;

    // Konstruktor
    public Karta(VrednostKarte vrednost, BojaKarte boja) {
        if (vrednost == null || boja == null)
            System.out.println(
                "Greška: boja i vrednost karte ne mogu biti null.");
        this.vrednost = vrednost;
        this.boja = boja;
    }

    // Geter metodi za vrednost i boju karte
    public VrednostKarte getVrednost() {
        return vrednost;
    }

    public BojaKarte getBoja() {
        return boja;
    }

    public String toString() {
        return vrednost + " " + boja;
    }

    public static void main(String[] args) {
        System.out.println("Šipil od 52 karte je:\n");
        for (BojaKarte b : BojaKarte.values())
            for (VrednostKarte v : VrednostKarte.values() ) {
                Karta k = new Karta(v, b);
                System.out.println(k);
            }
    }
}

enum BojaKarte {

    KARO, HERC, PIK, TREF;

    public String toString() {
```

```
switch (this) {
    case KARO: return "Karo";
    case HERC: return "Herc";
    case PIK: return "Pik";
    case TREF: return "Tref";
    default:
        System.out.println(
            "Greška: ovaj slučaj boje karte je nemoguć.");
        return null;
    }
}

enum VrednostKarte {

    KEC, DVOJKA, TROJKA, ČETVORKA, PETICA, ŠESTICA, SEDMICA,
    OSMICA, DEVETKA, DESETKA, ŽANDAR, DAMA, KRALJ;

    public int decVrednost() {

        switch (this) {
            case ŽANDAR: case DAMA: case KRALJ:
                return 2 + ordinal();
            default:
                return 1 + ordinal();
        }
    }

    public String toString() {
        return "" + decVrednost();
    }
}
```

## 2.

LISTING 9.2: Konvertor.java

```
import java.util.*;

public class Konvertor implements Konverzije {

    public double pretvoriInče (double in) { // u centimetre
        return in * INČ_CM;
    }

    public double pretvoriCentimetre(double cm) { // u inče
        return cm / INČ_CM;
    }
}
```

```
public double pretvoriUnce(double oz) { // u grame
    return oz * UNCA_GRAM;
}

public double pretvoriGrame(double g) { // u unce
    return g / UNCA_GRAM;
}

public double pretvoriFunte(double lb) { // u kilograme
    return lb * FUNTA_KG;
}

public double pretvoriKilograme(double kg) { // u funte
    return kg / FUNTA_KG;
}

public double pretvoriKonjskeSnage(double ks) { // u vate
    return ks * KS_VAT;
}

public double pretvoriVate(double w) { // u konjske snage
    return w / KS_VAT;
}

// „Klijentska strana“ klase radi testiranja
public static void main(String[] args) {

    Konvertor konv = new Konvertor();

    Scanner tastatura = new Scanner(System.in);

    System.out.print("Koliko auto ima konjskih snaga? ");
    double snaga = tastatura.nextDouble();
    System.out.print("Snaga auta u kilovatima je: ");
    System.out.printf(
        "%6.2fKW\n", konv.pretvoriKonjskeSnage(snaga)/1000);

    System.out.println();
    System.out.print("Koliki vam je obim struka (u cm)? ");
    double struk = tastatura.nextDouble();
    System.out.print("Kolika vam je dužina nogu (u cm)? ");
    double dužina = tastatura.nextDouble();
    System.out.print("Vaša mera za pantalone je: ");
    System.out.printf(
        "W=%3din, ", (int)konv.pretvoriCentimetre(struk));
    System.out.printf(
```

```

        "L=%3din\n", (int)konv.pretvoriCentimetre(dužina));
    }
}

interface Konverzije {

    // Konstante za konverzije mernih jedinica
    double INČ_CM = 2.54;
    double UNCA_GRAM = 28.3495231;
    double FUNTA_KG = 0.4535924;
    double KS_VAT = 745.7;

    // Apstraktni metodi za konverzije mernih jedinica
    double pretvoriInče(double in);           // u centimetre
    double pretvoriCentimetre(double cm);      // u inče
    double pretvoriUnce(double oz);           // u grame
    double pretvoriGrame(double g);           // u unce
    double pretvoriFunte(double lb);          // u kilograme
    double pretvoriKilograme(double kg);       // u funte
    double pretvoriKonjskeSnage(double ks);   // u vate
    double pretvoriVate(double w);            // u konjske snage
}

```

## 3.

LISTING 9.3: KošarkaškaUtakmica.java

```

import java.util.*;

public class KošarkaškaUtakmica {

    public static void main(String[] args) {

        Zapisnik z = new Zapisnik();
        Semafor s = new Semafor();
        MobilniTelefon m = new MobilniTelefon();

        System.out.println("Počinje derbi ...");
        Utakmica derbi = new Utakmica("Partizan", "Zvezda");

        // Registrovanje svih prenosnika derbija
        derbi.dodajPrenosnikaUtakmice(s);
        derbi.dodajPrenosnikaUtakmice(z);
        derbi.dodajPrenosnikaUtakmice(m);

        System.out.println("Simulacija utakmice ...");
        derbi.domaćinPostigaoKoš(3);
        derbi.gostPostigaoKoš(2);
    }
}

```

```
derbi.krajČetvrtine(1);

derbi.gostPostigaoKoš(2);
derbi.gostPostigaoKoš(2);
derbi.krajČetvrtine(2);

derbi.krajČetvrtine(3);

derbi.domaćinPostigaoKoš(3);
derbi.domaćinPostigaoKoš(3);
derbi.gostPostigaoKoš(1);
derbi.krajČetvrtine(4);

System.out.println();
System.out.println("Završen derbi ...");
z.prikažiRezultat();
}

}

interface PrenosnikUtakmice {
    public void domaćiTím(String ime);
    public void gostujućiTím(String ime);
    public void domaćinPoentirao(int poeni);
    public void gostPoentirao(int poeni);
    public void završenaČetvrtina(int četvrtina);
}

class Utakmica {
    private String imeDomaćina, imeGosta;
    private ArrayList prenosi;

    public Utakmica(String imeDomaćina, String imeGosta) {
        this.imeDomaćina = imeDomaćina;
        this.imeGosta = imeGosta;
        prenosi = new ArrayList();
    }

    public void dodajPrenosnikaUtakmice(PrenosnikUtakmice p) {

        // Dodavanje prenosnika utakmice listi prenosnika
        p.domaćiTim(imeDomaćina);
        p.gostujućiTim(imeGosta);

        prenosi.add(p);
    }

    public void domaćinPostigaoKoš(int poeni) {
```

```
// Obavestiti publiku o tome da je domaćin postigao koš
for (int i = 0; i < prenosi.size(); i++) {
    PrenosnikUtakmice p = (PrenosnikUtakmice) prenosi.get(i);
    p.domaćinPoentirao(poeni);
}

public void gostPostigaoKoš(int poeni) {

    // Obavestiti publiku o tome da je gost postigao koš
    for (int i = 0; i < prenosi.size(); i++) {
        PrenosnikUtakmice p = (PrenosnikUtakmice) prenosi.get(i);
        p.gostPoentirao(poeni);
    }
}

public void krajČetvrtine(int četvrtina) {

    // Obavestiti publiku o završenoj četvrtini
    for (int i = 0; i < prenosi.size(); i++) {
        PrenosnikUtakmice p = (PrenosnikUtakmice) prenosi.get(i);
        p.završenaČetvrtina(četvrtina);
    }
}

class Semafor implements PrenosnikUtakmice {
    private String domaćin, gost;
    private int rezultatDomaćina, rezultatGosta;
    private int četvrtinaUtakmice;

    public Semafor() {
        System.out.println("Pali se semafor ...");
        četvrtinaUtakmice = 1;
    }

    public void prikažiRezultat() {
        System.out.println("*****");
        System.out.print(domaćin + " " + rezultatDomaćina + " -- ");
        System.out.print(gost + " " + rezultatGosta);
        prikažiČetvrtinu();
        System.out.println("*****");
    }

    public void prikažiČetvrtinu() {
        if(četvrtinaUtakmice <= 4)
```

```
        System.out.println("Četvrtina: " + četvrtinaUtakmice);
    else
        System.out.println("Završena utakmica");
}

public void domaćinPoentirao(int poeni) {
    System.out.println();
    System.out.println(
        "Domaći tim " + domaćin + " postigao " + poeni + " poena.");
    rezultatDomaćina += poeni;
    prikažiRezultat();
}

public void gostPoentirao(int poeni) {
    System.out.println();
    System.out.println(
        "Gostujući tim " + gost + " postigao " + poeni + " poena.");
    rezultatGosta += poeni;
    prikažiRezultat();
}

public void završenaČetvrtina(int četvrtina) {
    System.out.println();
    System.out.println(četvrtina +
        ". četvrtina se upravo završila.");
    četvrtinaUtakmice++;
    prikažiRezultat();
}

public void domaćiTim(String ime) {
    domaćin = ime;
}

public void gostujućiTim(String ime) {
    gost = ime;
}
}

class Zapisnik implements PrenosnikUtakmice {
    private String domaćin, gost;
    private int rezultatDomaćina, rezultatGosta;
    private int četvrtinaUtakmice;

    public Zapisnik() {
        System.out.println("Otvara se zapisnik ...");
        četvrtinaUtakmice = 1;
    }
}
```

```
public void domaćinPoentirao(int poeni) {
    rezultatDomaćina += poeni;
}

public void gostPoentirao(int poeni) {
    rezultatGosta += poeni;
}

public void završenaČetvrtina(int četvrtina) {
    četvrtinaUtakmice++;
}

public void domaćiTím(String ime) {
    domaćin = ime;
}

public void gostujućiTim(String ime) {
    gost = ime;
}

public void prikažiRezultat() {
    System.out.println("Rezultat utakmice:");
    System.out.println(domaćin + ": " + rezultatDomaćina);
    System.out.println(gost + ": " + rezultatGosta);
    prikažiČetvrtinu();
}

public void prikažiČetvrtinu() {
    if(četvrtinaUtakmice <= 4)
        System.out.println("Četvrtina: " + četvrtinaUtakmice);
    else
        System.out.println("Završena utakmica");
}
}

class MobilniTelefon implements PrenosnikUtakmice {
    private String domaćin, gost;
    private int rezultatDomaćina, rezultatGosta;
    private int četvrtinaUtakmice;

    public MobilniTelefon() {
        System.out.println("SMS poruka: počinje prenos utakmice ...");
        četvrtinaUtakmice = 1;
    }

    public void prikažiRezultat() {
```

```
System.out.print(domaćin + " " + rezultatDomaćina + " -- ");
System.out.println(gost + " " + rezultatGosta);
prikažiČetvrtinu();
}

public void prikažiČetvrtinu() {
    if(četvrtinaUtakmice <= 4)
        System.out.println("Četvrtina: " + četvrtinaUtakmice);
    else
        System.out.println("Završena utakmica");
}

public void domaćinPoentirao(int poeni) {
    System.out.println();
    System.out.println("Početak SMS poruke");
    System.out.println(
        "Domaći tim " + domaćin + " postigao " + poeni + " poena.");
    rezultatDomaćina += poeni;
    prikažiRezultat();
    System.out.println("Kraj SMS poruke");
}

public void gostPoentirao(int poeni) {
    System.out.println();
    System.out.println("Početak SMS poruke");
    System.out.println(
        "Gostujući tim " + gost + " postigao " + poeni + " poena.");
    rezultatGosta += poeni;
    prikažiRezultat();
    System.out.println("Kraj SMS poruke");
}

public void završenaČetvrtina(int četvrtina) {
    System.out.println();
    System.out.println("Početak SMS poruke");
    System.out.println(četvrtina +
        ". četvrtina se upravo završila.");
    četvrtinaUtakmice++;
    prikažiRezultat();
    System.out.println("Kraj SMS poruke");
}

public void domaćiTim(String ime) {
    domaćin = ime;
}

public void gostujućiTim(String ime) {
```

```
        gost = ime;
    }
}
```

4.

LISTING 9.4: GO.java

```
public class GO {

    public static void main(String[] args) {

        Krug k = new Krug(new Tačka(0, 0), 1);
        System.out.println("Krug: " + k);
        k.transliraj(new Tačka(1, 1));
        System.out.println("Krug: " + k);
        System.out.println("Obim: " + k.obim());
        System.out.println("Površina: " + k.površina());

        System.out.println();

        Pravougaonik p = new Pravougaonik(
            new Tačka(1, 2), new Tačka(3, 5));
        System.out.println("Pravougaonik: " + p);
        p.transliraj(new Tačka(1, 1));
        System.out.println("Pravougaonik: " + p);
        System.out.println("Obim: " + p.obim());
        System.out.println("Površina: " + p.površina());
    }
}

abstract class GeometrijskiOblik {

    protected Tačka refTačka; // referentna tačka oblika

    // Konstruktor
    public GeometrijskiOblik(Tačka refTačka) {
        this.refTačka = refTačka;
    }

    // Konkretan metod
    public Tačka getRefTačka() {
        return refTačka;
    }

    // Apstraktni metodi
    public abstract void transliraj(Tačka novaRefTačka);
}
```

```
public abstract double obim();

public abstract double povrsina();
}

/** Proširena klasa Krug koja krug u ravni predstavlja
koordinatama centra i dužinom poluprečnika */
class Krug extends GeometrijskiOblik {

    private Tačka centar;          // centar kruga
    private double poluprečnik; // poluprečnik

    // Konstruktor
    public Krug(Tačka centar, double poluprečnik) {
        super(centar);
        this.centar = centar;
        this.poluprečnik = poluprečnik;
    }

    public void transliraj(Tačka novaRefTačka) {
        refTačka = novaRefTačka;
        centar = novaRefTačka;
    }

    public double obim() {
        return 2 * Math.PI * poluprečnik;
    }

    public double povrsina() {
        return poluprečnik * poluprečnik * Math.PI;
    }

    public String toString() {
        return "<" + centar + "," + poluprečnik + ">";
    }
}

/** Proširena klasa Pravougaonik koja pravougaonik u ravni predstavlja
koordinatama donjeg levog i gornjeg desnog temena */
class Pravougaonik extends GeometrijskiOblik {

    private Tačka dlt, gdt; // donje levo i gornje desno teme

    // Konstruktor
    public Pravougaonik(Tačka dlt, Tačka gdt) {
        super(dlt);
        this.dlt = dlt;
```

```
        this.gdt = gdt;
    }

    public void transliraj(Taćka novaRefTačka) {
        refTačka = novaRefTačka;
        double širina = gdt.getX() - dlt.getX();
        double visina = gdt.getY() - dlt.getY();
        dlt = novaRefTačka;
        gdt = new Tačka(novaRefTačka.getX() + širina,
                         novaRefTačka.getY() + visina);
    }

    public double obim() {
        double širina = gdt.getX() - dlt.getX();
        double visina = gdt.getY() - dlt.getY();
        return 2 * (širina + visina);
    }

    public double površina() {
        double širina = gdt.getX() - dlt.getX();
        double visina = gdt.getY() - dlt.getY();
        return širina * visina;
    }

    public String toString() {
        return "<" + dlt + "," + gdt + ">";
    }
}

/** Tačka sa koordinatama (x,y) u ravni */
class Tačka {

    private double x, y;                                // koordinate tačke

    public Tačka(double x, double y) { // konstruktor
        this.x = x;
        this.y = y;
    }

    public double getX() {
        return x;
    }

    public double getY() {
        return y;
    }
}
```

```
public String toString() {
    return "(" + x + "," + y + ")";
}
```

5.

LISTING 9.5: GO1.java

```
import java.util.*;

public class GO1 {

    public static void main(String[] args) {

        GeometrijskiOblik[] nizGO = new GeometrijskiOblik[20];
        Tačka O = new Tačka(0, 0); // koordinatni početak

        // Konstruisati krug ili pravougaonik na slučajan način
        for (int i = 0 ; i < 20 ; i++) {
            if (Math.random() < 0.5) { // krug
                double r = Math.random()*10; // slučajan poluprečnik
                nizGO[i] = new Krug(O, r);
            }
            else { // pravougaonik
                double š = Math.random()*10; // slučajna širina
                double v = Math.random()*10; // slučajna visina
                nizGO[i] = new Pravougaonik(O, new Tačka(š, v));
            }
        }

        // Sortiranje slučajnog niza geometrijskih oblika
        Arrays.sort(nizGO);

        // Prikazivanje niza u rastućem redosledu
        System.out.println("Rastući niz geometrijskih oblika:\n");
        for (GeometrijskiOblik go : nizGO) {
            if (go instanceof Krug)
                System.out.print("Krug, P = ");
            else
                System.out.print("Pravougaonik, P = ");
            System.out.printf("%6.2f\n", go.površina());
        }
    }

    abstract class GeometrijskiOblik implements Comparable {
```

```
protected Tačka refTačka; // referentna tačka oblika

// Konstruktor
public GeometrijskiOblik(Tačka refTačka) {
    this.refTačka = refTačka;
}

// Konkretan metod
public Tačka getRefTačka() {
    return refTačka;
}

// Apstraktni metodi
public abstract void transliraj(Tačka novaRefTačka);

public abstract double obim();

public abstract double površina();

// Implementacija metoda za poređenje geometrijskih oblika
public int compareTo (Object o) {
    GeometrijskiOblik go = (GeometrijskiOblik)o;
    if (this.površina() < go.površina()) return -1;
    if (this.površina() > go.površina()) return +1;
    return 0;
}

}

/** Proširena klasa Krug koja krug u ravni predstavlja
koordinatama centra i dužinom poluprečnika */
class Krug extends GeometrijskiOblik {

    private Tačka centar;      // centar kruga
    private double poluprečnik; // poluprečnik

    // Konstruktor
    public Krug(Tačka centar, double poluprečnik) {
        super(centar);
        this.centar = centar;
        this.poluprečnik = poluprečnik;
    }

    public void transliraj(Tačka novaRefTačka) {
        refTačka = novaRefTačka;
        centar = novaRefTačka;
    }
}
```

```
public double obim() {
    return 2 * Math.PI * poluprečnik;
}

public double površina() {
    return poluprečnik * poluprečnik * Math.PI;
}

public String toString() {
    return "<" + centar + "," + poluprečnik + ">";
}
}

/** Proširena klasa Pravougaonik koja pravougaonik u ravni predstavlja
koordinatama donjeg levog i gornjeg desnog temena */
class Pravougaonik extends GeometrijskiOblik {

    private Tačka dlt, gdt; // donje levo i gornje desno teme

    // Konstruktor
    public Pravougaonik(Tačka dlt, Tačka gdt) {
        super(dlt);
        this.dlt = dlt;
        this.gdt = gdt;
    }

    public void transliraj(Tačka novaRefTačka) {
        refTačka = novaRefTačka;
        double širina = gdt.getX() - dlt.getX();
        double visina = gdt.getY() - dlt.getY();
        dlt = novaRefTačka;
        gdt = new Tačka(novaRefTačka.getX() + širina,
                         novaRefTačka.getY() + visina);
    }

    public double obim() {
        double širina = gdt.getX() - dlt.getX();
        double visina = gdt.getY() - dlt.getY();
        return 2 * (širina + visina);
    }

    public double površina() {
        double širina = gdt.getX() - dlt.getX();
        double visina = gdt.getY() - dlt.getY();
        return širina * visina;
    }
}
```

```
public String toString() {
    return "<" + dlt + "," + gdt + ">";
}

/** Tačka sa koordinatama (x,y) u ravni */
class Tačka {

    private double x, y;                      // koordinate tačke

    public Tačka(double x, double y) { // konstruktor
        this.x = x;
        this.y = y;
    }

    public double getX() {
        return x;
    }

    public double getY() {
        return y;
    }

    public String toString() {
        return "(" + x + "," + y + ")";
    }
}
```

6.

LISTING 9.6: KućnaZabava.java

```
public class KućnaZabava {

    public static void main(String[] args) {

        // Testiranje metoda TV-a
        TV sony = new TV("Sony", 56,
                          new int[]{50, 100}, new int[]{0, 99});
        sony.uključiIsključi();
        sony.zvukMinus(); sony.zvukMinus();
        sony.uključiIsključi();
        sony.uključiIsključi();
        sony.zvukPlus();
        sony.ugasiZvuk();
        sony.zvukPlus(); sony.zvukPlus();
        sony.izaberikanal(99); sony.kanalPlus(); sony.kanalPlus();
    }
}
```

```
sony.izaberikanal(0); sony.kanalMinus(); sony.kanalMinus();

// Testiranje metoda DVD plejera
DVDPlejer samsung = new DVDPlejer("Samsung", new int[]{50, 100});
samsung.uključiIsključi();
samsung.zvukPlus(); samsung.zvukPlus();
samsung.ubaciDisk("Ko to tamo peva");
samsung.ubaciDisk("Prljavi Hari");
samsung.prikažiDisk();
samsung.uključiIsključi();
samsung.uključiIsključi();
samsung.prikažiDisk();
samsung.izbacidiDisk();
samsung.prikažiDisk();
System.out.println();

// Testiranje polimorfizma pet uređaja za kućnu zabavu
Uređaj u;
for (int i = 0 ; i < 5 ; i++) {
    // Konstruisati TV ili DVD plejer na slučajan način
    if (Math.random() < 0.5)
        u = new TV(Math.random() < 0.5 ? "Sony" : "Panasonic",
                    Math.random() < 0.5 ? 56 : 71,
                    new int[]{50, 100}, new int[]{0, 99});
    else
        u = new DVDPlejer(Math.random() < 0.5 ? "Samsung" : "LG",
                           new int[]{50, 100});
    // Uključivanje uređaja i gašenje zvuka, bez obzira na tip
    u.uključiIsključi();
    ((DaljinskiUpravljač)u).ugasiZvuk();
}
}

abstract class Uređaj {

    protected String marka;      // marka uređaja
    protected boolean uključen;   // uključen/isključen uređaj

    abstract boolean uključiIsključi(); // uključiti/isključuti uređaj
}

interface DaljinskiUpravljač {

    public boolean uključiIsključi(); // uključiti/isključuti uređaj
    public int zvukPlus();           // pojačati zvuk uređaja
    public int zvukMinus();          // utišati zvuk uređaja
}
```

```
public void ugasiZvuk();           // ugasiti zvuk uređaja
}

class TV extends Uredaj implements DaljinskiUpzravljač {

    private int ekran;
    private int[] zvuk; // min zvuk = 0
                        // zvuk[0] = trenutni zvuk
                        // zvuk[1] = max zvuk
    private int[] kanal; // min kanal = 0
                        // kanal[0] = trenutni kanal
                        // kanal[1] = max kanal

    public TV(String marka, int ekran, int[] zvuk, int[] kanal) {

        this.marka = marka;
        this.ekran = ekran;
        this.zvuk = zvuk;
        this.kanal = kanal;
    }

    public boolean uključiIsključi() {

        uključen = !uključen;
        System.out.print("TV " + marka + ", ekran: " + ekran);
        System.out.println(", " + (uključen ? "u" : "is") + "ključen.");
        return uključen;
    }

    public int zvukMinus() {

        if (uključen) {
            zvuk[0] = Math.max(0, zvuk[0] - 1);
            System.out.print("TV " + marka + ", ekran: " + ekran);
            System.out.println(", zvuk: " + zvuk[0]);
        }
        return zvuk[0];
    }

    public int zvukPlus() {

        if (uključen) {
            zvuk[0] = Math.min(zvuk[0] + 1, zvuk[1]);
            System.out.print("TV " + marka + ", ekran: " + ekran);
            System.out.println(", zvuk: " + zvuk[0]);
        }
        return zvuk[0];
    }
}
```

```
}

public void ugasiZvuk() {

    if (uključen) {
        zvuk[0] = 0;
        System.out.print("TV " + marka + ", ekran: " + ekran);
        System.out.println(", zvuk: " + zvuk[0]);
    }
}

public int izaberikanal(int k) {

    if (uključen) {
        if (k >= 0 && k <= kanal[1])
            kanal[0] = k;
        System.out.print("TV " + marka + ", ekran: " + ekran);
        System.out.println(", kanal: " + kanal[0]);
    }
    return kanal[0];
}

public int kanalPlus() {

    if (uključen) {
        kanal[0] = kanal[0] < kanal[1] ? kanal[0] + 1 : 0;
        System.out.print("TV " + marka + ", ekran: " + ekran);
        System.out.println(", kanal: " + kanal[0]);
    }
    return kanal[0];
}

public int kanalMinus() {

    if (uključen) {
        kanal[0] = kanal[0] > 0 ? kanal[0] - 1 : kanal[1];
        System.out.print("TV " + marka + ", ekran: " + ekran);
        System.out.println(", kanal: " + kanal[0]);
    }
    return kanal[0];
}

class DVDPlayer extends Uredaj implements DaljinskiUpravljač {

    private String disk; // ime diska u plejeru
    private int[] zvuk; // min zvuk = 0
```

```
// zvuk[0] = trenutni zvuk
// zvuk[1] = max zvuk

public DVDPlayer(String marka, int[] zvuk) {

    this.marka = marka;
    this.zvuk = zvuk;
}

public boolean uključiIIsključi() {

    uključen = !uključen;
    System.out.print("DVD plejer " + marka);
    System.out.println(", " + (uključen ? "u" : "is") + "ključen.");
    return uključen;
}

public int zvukMinus() {

    if (uključen) {
        zvuk[0] = Math.max(0, zvuk[0] - 1);
        System.out.print("DVD plejer " + marka);
        System.out.println(", zvuk: " + zvuk[0]);
    }
    return zvuk[0];
}

public int zvukPlus() {

    if (uključen) {
        zvuk[0] = Math.min(zvuk[0] + 1, zvuk[1]);
        System.out.print("DVD plejer " + marka);
        System.out.println(", zvuk: " + zvuk[0]);
    }
    return zvuk[0];
}

public void ugasiZvuk() {

    if (uključen) {
        zvuk[0] = 0;
        System.out.print("DVD plejer " + marka);
        System.out.println(", zvuk: " + zvuk[0]);
    }
}

public void ubaciDisk(String d) {
```

```
if (uključen) {
    if (disk == null)
        disk = d;
    System.out.print("DVD plejer " + marka);
    System.out.println(", disk: \\" + disk + "\\\"");
}
}

public void prikažiDisk() {

if (uključen) {
    if (disk == null) {
        System.out.print("DVD plejer " + marka);
        System.out.println(", nema diska");
    }
    else {
        System.out.print("DVD plejer " + marka);
        System.out.println(", prikazuje se: \\" + disk + "\\\"");
    }
}
}

public void izbacidiDisk() {

if (uključen) {
    disk = null;
    System.out.print("DVD plejer " + marka);
    System.out.println(", nema diska");
}
}
}
```

7.

LISTING 9.7: PovezanaLista2.java

```
public class PovezanaLista2 {

private ElementLista prvi; // prvi element liste
private ElementLista posl; // poslednji element liste
private int n; // broj elemenata liste

// Ugnježđena klasa za elemente liste
// Napomena: pristupni metodi za njena polja nisu potrebni
private class ElementLista {

private Object sadržaj; // sadržaj elementa liste
}
```

```
private ElementListe sled; // pokazivač na sledeći
                           // element liste

// Konstruktor
public ElementListe(Object o) {
    sadržaj = o;
}

// String reprezentacija elementa liste
public String toString() {
    return sadržaj.toString();
}
}

// Podrazumevani konstruktor za formiranje prazne liste
public PovezanaLista2() {}

// Dužina liste
public int dužina() {
    return n;
}

// Ispitivanje da li je lista prazna
public boolean praznaLista() {
    return prvi == null;
}

// Dodavanje objekta na kraj liste
public void dodaj(Object o) {

    // Konstruisati novi element liste
    ElementListe noviElem = new ElementListe(o);

    if (praznaLista())
        prvi = posl = noviElem;
    else {
        posl.sled = noviElem;
        posl = noviElem;
    }
    n++;
}

// Ispitivanje da li je dati objekat u listi
public boolean nađi(Object o) {
    ElementListe elem;
    for (elem = prvi; elem != null; elem = elem.sled) {
        if (elem.sadržaj.equals(o))
            return true;
    }
    return false;
}
```

```
        break;
    }
    return (elem != null);
}

// String reprezentacija povezane liste
public String toString() {
    String s = "";
    for (ElementLista elem = prvi; elem != posl; elem = elem.sled) {
        s = s + elem.toString() + ", ";
    }
    if (posl != null)
        s = s + posl.toString();
    return s;
}

// „Klijentska strana“ klase radi testiranja
public static void main(String[] args) {

    // Konstruisanje prazne liste
    PovezanaLista2 lis = new PovezanaLista2();

    // Dodavanje nekih elemenata u listu
    Object o = new Integer(17);
    lis.dodaj(o);
    lis.dodaj(23); lis.dodaj(31); lis.dodaj(47);

    // Prikazivanje elemenata liste
    System.out.println("Sadržaj liste:");
    System.out.println(lis);
    System.out.println("Dužina liste: " + lis.duzina());

    // Traženje objekta u listi
    int x = 23;
    System.out.println();
    System.out.print("Objekat " + x + " se ");
    if (lis.nadi((Integer)x) == false)
        System.out.print("ne ");
    System.out.println("nalazi u listi");
}
}
```

## 8.

LISTING 9.8: IgraŽivotai1.java

```
import java.util.*;
```

```
public class IgraŽivotal {  
  
    public static void main(String[] args) {  
  
        System.out.println("Ovo je igra života!\n");  
  
        System.out.print("Unesite veličinu (broj vrsta ");  
        System.out.print("i kolona) kolonije: ");  
  
        Scanner tastatura = new Scanner(System.in);  
        int n = tastatura.nextInt();  
        Kolonija kol = new Kolonija(n);  
  
        System.out.print("Unesite broj organizama na početku: ");  
        int brojOrganizama = tastatura.nextInt();  
        System.out.print("Unesite vrste i kolone ");  
        System.out.println("organizama na početku - ");  
        for (int i = 0; i < brojOrganizama; i++) {  
            System.out.print("Organizam " + (i+1) + ": ");  
            int v = tastatura.nextInt();  
            int k = tastatura.nextInt();  
            kol.zauzmiĆeliju(v,k);  
        }  
  
        System.out.println();  
        int g = 0;  
        while (true) {  
            System.out.println("Generacija " + g + ": ");  
            kol.prikaži();  
            System.out.print("Sledeća generacija (d/n)? ");  
            String novaGen = tastatura.next();  
            if (novaGen.equals("n")) break;  
            kol.novaGen();  
            g++;  
        }  
    }  
}  
  
class Kolonija {  
  
    private int n;          // veličina kolonije  
    private Ćelija[][] m;  // matrica ćelija  
  
    // Ugnježđeni nabrojivi tip za stanje ćelija  
    private enum Stanje {  
        PRAZNO, ZAUZETO;
```

```
public String toString() {
    return (ordinal() == 0) ? " " : "*";
}

// Ugnježđena klasa za čelije matrice
private class Čelija {

    private int vrsta, kolona; // koordinate čelije u matrici
    private Stanje sadržaj; // prazna ili zauzeta čelija
    private int susedi; // broj suseda čelije

    // Konstruktor
    private Čelija(int vrsta, int kolona) {
        this.vrsta = vrsta;
        this.kolona = kolona;
        this.sadržaj = Stanje.PRAZNO;
    }

    private void brojSuseda() {

        int bs = 0;
        for (int i = vrsta - 1; i <= vrsta + 1; i++)
            for (int j = kolona - 1; j <= kolona + 1; j++)
                if (m[i][j].sadržaj == Stanje.ZAUZETO)
                    bs++;
        susedi = (sadržaj == Stanje.ZAUZETO) ? bs - 1 : bs;
    }

    private void novoStanje() {

        switch (sadržaj) {
            case PRAZNO :
                if (susedi == 3)
                    sadržaj = Stanje.ZAUZETO;
                break;
            case ZAUZETO :
                if (susedi < 2 || susedi > 3)
                    sadržaj = Stanje.PRAZNO;
                break;
        }
    }

    private void prikaži() {
        System.out.print(sadržaj);
    }
}
```

```

// Konstruktor
public Kolonija(int n) {
    this.n = n;
    m = new Ćelija[n+2][n+2]; // prazne ćelije oko ivice matrice
    for (int i = 0; i < n+2; i++)
        for (int j = 0; j < n+2; j++)
            m[i][j] = new Ćelija(i, j);
}

public void zauzmiĆeliju(int i, int j) {
    m[i][j].sadržaj = Stanje.ZAUZETO;
}

public void novaGen() {
    for (int i = 1; i <= n; i++)
        for (int j = 1; j <= n; j++)
            m[i][j].brojSuseda();

    // Novo stanje tek nakon određivanja suseda
    for (int i = 1; i <= n; i++)
        for (int j = 1; j <= n; j++)
            m[i][j].novoStanje();
}

public void prikaži() {
    for (int i = 1; i <= n; i++) {
        for (int j = 1; j <= n; j++)
            m[i][j].prikaži();
        System.out.println();
    }
}
}

```

## 9.

LISTING 9.9: Sortiranje.java

```

public class Sortiranje {

    public static void main(String[] args) {

        // Konstruisanje niza kompleksnih brojeva tako da
        // njihovi realni i imaginarni delovi budu slučajni
        KompleksanBroj[] a = new KompleksanBroj[10];
        for (int i = 0; i < a.length; i++)
            a[i] = new KompleksanBroj(Math.random()*5, Math.random()*5);
        for (KompleksanBroj z : a) System.out.println(z);
    }
}

```

```
System.out.println();

// Sortiranje niza kompleksnih brojeva prema njihovim modulima.
// Ta relacija poretka kompleksnih brojeva definisana je objektom
// anonimne klase koja implementira interfejs Sortirač.Upoređivač.
Sortirač.sortiraj(a, new Sortirač.Upoređivač() {
    public int uporedi(Object x, Object y) {
        KompleksanBroj z1 = (KompleksanBroj)x;
        KompleksanBroj z2 = (KompleksanBroj)y;
        double rezultat = z1.moduo() - z2.moduo();
        return (int)Math.signum(rezultat);
    }
};

for (KompleksanBroj z : a) System.out.println(z);
System.out.println();

// Sortiranje niza kompleksnih brojeva prema njihovim realnim,
// pa zatim imaginarnim delovima.
// Ta relacija poretka kompleksnih brojeva definisana je objektom
// anonimne klase koja implementira interfejs Sortirač.Upoređivač.
Sortirač.sortiraj(a, new Sortirač.Upoređivač() {
    public int uporedi(Object x, Object y) {
        KompleksanBroj z1 = (KompleksanBroj)x;
        KompleksanBroj z2 = (KompleksanBroj)y;
        double rezultat = z1.Re() - z2.Re();
        if (rezultat == 0)
            rezultat = z1.Im() - z2.Im();
        return (int)Math.signum(rezultat);
    }
};

for (KompleksanBroj z : a) System.out.println(z);
}

/**
 * Klasa Sortirač u kojoj je definisan statički metod za sortiranje niza
 * objekata u rastućem redosledu. Relativan redosled objekata određen je
 * objektom tipa Upoređivač.
 */
class Sortirač {

    /**
     * Ugnježđeni interfejs Upoređivač u kojem je definisan metod
     * uporedi() za poređenje dva objekta. Objekat tipa Upoređivač
     * se koristi kao parametar za sortiranje niza objekata.
}
```

```
/*
public static interface Uporedivač {
    /**
     * Metod za poređenje dva objekta tako da:
     * ako x > y, rezultat > 0;
     * ako x == y, rezultat = 0;
     * ako x < y, rezultat < 0.
    */
    public int uporedi(Object x, Object y);
}

/**
 * Metod kojim se sortira deo niza objekata koji na nalazi između
 * indeksa leviKraj i desniKraj.
 * Za sortiranje se koristi rekurzivni algoritam quicksort.
*/
public static void sortiraj(
    Object[] a, int leviKraj, int desniKraj, Uporedivač u) {

    // Određivanje indeksa pivota na slučajan način tako da bude
    // između indeksa leviKraj i desniKraj
    int k = (int)(Math.random()*(desniKraj-leviKraj+1)) + leviKraj;
    Object pivot = a[k];

    // Preuređivanje niza tako da elementi niza manji od pivota budu
    // prebačeni levo od pivota, a elementi niza veći od pivota da
    // budu prebačeni desno od pivota.
    int i = leviKraj, j = desniKraj;
    do {
        while((i < desniKraj) && u.uporedi(pivot, a[i]) > 0) i++;
        while((j > leviKraj) && u.uporedi(pivot, a[j]) < 0) j--;
        // Zamena mesta većeg i manjeg elementu od pivota
        if (i < j) {
            Object tmp = a[i]; a[i] = a[j]; a[j] = tmp;
        }
        if (i <= j) { i++; j--; }
    } while(i <= j);

    // Rekurzivno sortiranje prvog i drugog dela niza
    if (leviKraj < j) sortiraj(a, leviKraj, j, u);
    if (i < desniKraj) sortiraj(a, i, desniKraj, u);
}

/**
 * Preopterećeni metod kojim se sortira ceo niz objekata.
*/
public static void sortiraj(Object[] a, Uporedivač u) {
```

```
        sortiraj(a, 0, a.length - 1, u);
    }
}

public class KompleksanBroj {

    private double r, i; // realni i imaginarni deo

    // Konstruktor
    public KompleksanBroj(double r, double i) {
        this.r = r;
        this.i = i;
    }

    // Geter metodi
    public double Re() { return r; }

    public double Im() { return i; }

    // Moduo kompleksnog broja
    public double moduo() { return Math.sqrt(r*r + i*i); }

    // Statički metod za konjugovan kompleksan broj:
    // KompleksanBroj z = KompleksanBroj.konjugovanBroj(x)
    public static KompleksanBroj konjugovanBroj(KompleksanBroj a) {
        return new KompleksanBroj(a.r, -a.i);
    }

    // Objektni metod za konjugovan kompleksan broj:
    // KompleksanBroj z = x.konjugovanBroj();
    public KompleksanBroj konjugovanBroj() {
        return new KompleksanBroj(this.r, -this.i);
    }

    // Statički metod za sabiranje dva kompleksna broja:
    // KompleksanBroj z = KompleksanBroj.zbir(x, y)
    public static KompleksanBroj zbir(
        KompleksanBroj a, KompleksanBroj b) {
        return new KompleksanBroj(a.r + b.r, a.i + b.i);
    }

    // Objektni metod za sabiranje dva kompleksna broja:
    // KompleksanBroj z = x.dodaj(y);
    public KompleksanBroj dodaj(KompleksanBroj a) {
        return new KompleksanBroj(this.r + a.r, this.i + a.i);
    }
}
```

```
// Statički metod za proizvod dva kompleksna broja:  
// KompleksanBroj z = KompleksanBroj.proizvod(x, y)  
public static KompleksanBroj proizvod(  
    KompleksanBroj a, KompleksanBroj b) {  
    return new KompleksanBroj(  
        a.r * b.r - a.i * b.i, a.r * b.i + a.i * b.r);  
}  
  
// Objektni metod za proizvod dva kompleksna broja:  
// KompleksanBroj z = x.pomnoži(y);  
public KompleksanBroj pomnoži(KompleksanBroj a) {  
    return new KompleksanBroj(  
        this.r * a.r - this.i * a.i, this.r * a.i + this.i * a.r);  
}  
  
public String toString() {  
    return "<" + r + "," + i + ">, " + moduo();  
}  
}
```



# 10

## ***Grafičko programiranje***

### **Odgovori na pitanja**

1.  vođeni događajima  
 sporiji od konzolnih programa  
 zasnovani na grafičkom korisničkom interfejsu  
 objektno orijentisani
  
2.  Color  
 Font  
 Component  
 JFrame  
 JComponent
  
3.  Color  
 Font  
 Component  
 JFrame  
 JComponent
  
4.  Program ima grešku, jer dugmeOK treba da bude tipa Button.

- Program ima grešku, jer se dugmeOK ne može direktno dodati okviru tipa JFrame.
  - Program ima grešku, jer se naredbom new JFrame("Moj okvir") ne može konstruisati okvir.
  - Program nema grešaka i normalno se izvršava.
5.  Prikazuje se samo dugme OK.  
 Prikazuje se samo dugme Cancel.  
 Prikazuju se oba dugmeta OK i Cancel, pri čemu se dugme OK nalazi levo od dugmeta Cancel.  
 Prikazuju se oba dugmeta OK i Cancel, pri čemu se dugme OK nalazi desno od dugmeta Cancel.
6.  1  
 2  
 3  
 0
7.  1  
 2  
 3  
 0
8.  rezolucija  
 grid  
 piksel  
 dot
9.  Koordinatni početak (0, 0) se nalazi u donjem levom uglu ekrana.  
 Koordinatni početak (0, 0) se nalazi u gornjem levom uglu ekrana.  
 Koordinatni početak (0, 0) se nalazi u centru ekrana.  
 Koordinatni početak (0, 0) se može programski promeniti.
10.  (200, 150)

- (200, 100)
  - (100, 100)
  - (100, 150)
11.  Gornji levi ugao okvira se nalazi u tački sa koordinatama (100, 150).  
 Gornji levi ugao okvira se nalazi u tački sa koordinatama (300, 200).  
 Širina okvira je 300 piksela i visina okvira je 200 piksela.  
 Širina okvira je 100 piksela i visina okvira je 150 piksela.
12.  (0, 0)  
 (10, 10)  
 (100, 100)  
 neodređene
13.  FlowLayout.  
 GridLayout.  
 BorderLayout.  
 Nijedan.
14.  FlowLayout.  
 GridLayout.  
 BorderLayout, ali dugme treba dodati centralnom polju panela.  
 BorderLayout, ali dugme treba dodati istočnom ili zapadnom polju panela.  
 BorderLayout, ali dugme treba dodati severnom ili južnom polju panela.
15.  `p.setLayout(new BorderLayout());`  
 `p.setLayout(BorderLayout());`  
 `p.setLayout(new BorderLayout(5, 10));`  
 `p.setLayout(new BorderLayout(BorderLayout.CENTER));`  
 `p.setLayout(BorderLayout(BorderLayout.CENTER));`

16.  k.add(p);  
 p.add(k);  
 p.insert(k);  
 p.append(k);  
 k.addContentPane(p);
17.  repaint()  
 update()  
 paintComponent()  
 init()
18.  Objekat tipa Graphics se može konstruisati pomoću new Graphics().  
 Svaki put kada se komponenta prikaže na ekranu, automatski se konstruiše njen objekat tipa Graphics.  
 Metod paintComponent() ne treba nikad pozivati direktno, jer se on automatski poziva od strane JVM.  
 Pozivanje metoda paintComponent() od strane JVM se može implicitno zahtevati metodom repaint().
19.  JLabel  
 JButton  
 JFrame  
 JComponent
20.  Program ima grašku, jer se u konstruktoru klase Test ne sme dodavati komponenta.  
 Program se normalno izvršava i prikazuje poruku Zdravo narode! u okviru.  
 Program se normalno izvršava, ali se ništa ne prikazuje u okviru.  
 Program bi prikazao poruku Zdravo narode! u okviru ukoliko bi se u glavnom metodu zamenio izraz new JFrame() sa new Test().  
 Program bi prikazao poruku Zdravo narode! u okviru ukoliko bi se u glavnom metodu zamenio izraz new JFrame() sa new Test("Zdravo narode!").

21.  Program ima grašku, jer se u konstruktoru klase Test ne sme dodavati komponenta.
- Program ima grašku, jer poruka ima vrednost null kada se izvršava naredba `g.drawString(poruka, 20, 20)` u metodu `paintComponent()`.
- Program se normalno izvršava, ali se ništa ne prikazuje u okviru.
- Program bi prikazao poruku Zdravo narode! u okviru ukoliko bi se u glavnom metodu izraz `new Test()` zamenio sa `new Test("Zdravo narode!")`.
- Program bi prikazao poruku Zdravo narode! u okviru ukoliko bi se u konstruktoru klase Test zamenio izraz `new MojaKomponenta()` sa `(new MojaKomponenta()).setPoruka("Zdravo narode!")`.
22.  `g.drawRect(20, 50, 20, 20);`
- `g.drawRectFill(20, 20, 20, 50);`
- `g.drawRect(20, 20, 20, 50);`
- `g.drawRectFill(20, 50, 20, 20);`
23.  `g2.draw(Rectangle2D(20, 20, 20, 50));`
- `g2.draw(Rectangle2D.Double(20, 20, 20, 50));`
- `g2.draw(new Rectangle2D.Double(20, 20, 20, 50));`
- `g2.draw(new Rectangle2D.Float(20, 20, 20, 50));`
- `g2.draw(new Rectangle2D.Double(20.0, 20.0, 20.0, 50.0));`
- `g2.draw(new Rectangle2D.Float(20.0, 20.0, 20.0, 50.0));`
- `g2.draw(new Rectangle2D.Float(20.0F, 20.0F, 20.0F, 50.0F));`
- `g2.draw(new Rectangle2D(20, 20, 20, 50));`
24.  `Point2D t = new Point2D(getWidth(), getHeight());`
- `Point2D t = new Point2D.Double(getWidth(), getHeight());`
- `Point2D.Double t = new Point2D.Double(getWidth(), getHeight());`
- `Point2D t = new Point2D.Double(0, 0);`
- `Point2D t = new Point2D.Double(0, getHeight());`
- `Point2D t = new Point2D.Double(getWidth(), 0);`
- `Point2D t = new Point2D.Double(getHeight(), getWidth());`

25.  new Color(0, 0, 0)  
 new Color(0, 266, 0)  
 new Color(255, 255, 255)  
 new Color(1, 2, 3)
26.  k.setBackground(Color.PINK);  
 k.setBackground(new Color(0, 128, 128));  
 k.setBackground(Color(0, 128, 128));  
 setBackground(Color.YELLOW)  
 k.setForeground(Color.RED);
27.  new Font("SansSerif", Font.BOLD, 36)  
 new Font("SansSerif", Font.CAPS, 20)  
 new Font("SansSerif", Font.BOLD, 10.5)  
 new Font("Serif", Font.BOLD + Font.ITALIC, 12)  
 new Font("Dialog", Font.PLAIN, 10)  
 new Font("Cyrilllic", Font.PLAIN, 12)  
 new Font(Serif, Font.PLAIN, 12)
28.  ItemEvent  
 MouseEvent  
 MouseMotionEvent  
 ActionEvent  
 WindowEvent
29.  ItemEvent  
 MouseEvent  
 MouseMotionEvent  
 ActionEvent  
 WindowEvent
30.  Komponenta u kojoj se desio neki događaj se naziva izvor događaja.

- Ako komponenta može generisati neki događaj, svaka naslednica te komponente može generisati isti tip događaja.
- Sve klase događaja su naslednice klase EventObject.
- Klase događaja i interfejsi rukovalaca događaja se nalaze u paketu java.events.
- Ime klase događaja u Javi je standardnog oblika <Ime>Event, a interfejsa rukovaoca odgovarajućeg događaja je oblika <Ime>Listener.

31.  getSource()

- getActionCommand()
- getTimeStamp()
- getWhen()

32.  objektu izvora događaja

- objektu rukovaoca događaja
- objektima izvora i rukovaoca događaja
- klasi Object
- klasi EventObject

33.  Svaka klasa događaja u Javi ima odgovarajući interfejs rukovaoca događaja.

- Klasa rukovaoca događaja mora implementirati interfejs rukovaoca odgovarajućeg događaja.
- Jednom izvoru događaja može biti pridruženo više rukovaoca događaja.
- Jeden rukovalac događaja može obrađivati događaje iz više izvora.
- Rukovalac događaja se automatski pridružuje izvoru događaja prema tipu događaja koje obrađuje.

34.  dugme.addListener(rukovalacDugmeta);

- dugme.addActionListener(rukovalacDugmeta);
- dugme.addActionEventListener(rukovalacDugmeta);
- dugme.addEventListener(rukovalacDugmeta);

35.  ButtonListener  
 ButtonPressedListener  
 MouseListener  
 ActionListener
36.  ActionEvent  
 ActionListener  
 EventObject  
 WindowListener
37.  Program ima logičku grešku, jer nijedan rukovalac akcijskog događaja dugmeta nije pridružen dugmetu OK.  
 Program se normalno izvršava i svaki put kada se pritisne na dugme OK u okviru prikazuje se poruka Dugme OK je pritisnuto.  
 Program se normalno izvršava, ali se ništa ne prikazuje u okviru.  
 Program se normalno izvršava prikazujući dugme OK u okviru, ali ne i poruku Dugme OK je pritisnuto kada se pritisne na dugme OK.
38.  Deklaracija `import java.awt.event.*` je suvišna, jer je njen efekat obuhvaćen deklaracijom `import java.awt.*`.  
 Program ima grešku, jer klasa Test ne može istovremeno da proširuje klasu JFrame i implementira interfejs ActionListener.  
 Program se normalno izvršava i svaki put kada se pritisne na dugme OK u okviru prikazuje se poruka Dugme OK je pritisnuto.  
 Program se normalno izvršava prikazujući dugme OK u okviru, ali ne i poruku Dugme OK je pritisnuto kada se pritisne na dugme OK.
39.  Program ima grešku u redovima 12 i 13, jer dve različite komponente ne mogu imati isti rukovalac njihovim događajima.  
 Program ima grešku u redovima 12 i 13, jer klasa Test ne implementira interfejs ActionListener.  
 Program ima grešku u redu 16, jer metod actionPerformed() ima pogrešan potpis.  
 Program ima grešku u redu 21, jer se objekat tipa Test dobijen izrazom `new Test()` dodeljuje promenljivoj okvir tipa JFrame.

- Program se normalno izvršava prikazujući dva dugmeta u okviru, ali ne i poruku Jedno od dva dugmeta je pritisnuto kada se pritisne na bilo koje od ova dva dugmeta u okviru.
40.  Program u okviru prikazuje dugme NOK levo od dugmeta OK.  
 Program prikazuje poruku Pritisnuto dugme OK kada se u okviru tasterom miša pritisne dugme OK.  
 Program prikazuje poruku Pritisnuto dugme NOK kada se u okviru tasterom miša pritisne dugme NOK.  
 Ukoliko se izostavi prva naredba super.actionPerformed(e) u metodu actionPerformed() klase Test, ne prikazuje se poruka Pritisnuto dugme NOK kada se u okviru tasterom miša pritisne dugme NOK.
41.  ActionAdapter  
 MouseAdapter  
 WindowAdapter

## Rešenja zadataka

1.

LISTING 10.1: Kamion.java

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

public class Kamion {

    public static void main(String[] args) {

        // Konstruisanje okvira
        JFrame okvir = new JFrame("Kamion");
        okvir.setSize(300, 300);
        okvir.setLocation(100, 150);
        okvir.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);

        SlikaKamiona fap = new SlikaKamiona();

        okvir.add(fap);
        okvir.setVisible(true);
    }
}

class SlikaKamiona extends JComponent {

    public void paintComponent(Graphics g) {

        Graphics2D g2 = (Graphics2D) g;

        // Bojenje pozadine
        Color bledoPlava = new Color(0.75f, 0.750f, 1.0f);
        g2.setPaint(bledoPlava);
        g2.fill(new Rectangle2D.Double(0,0,300,300));

        // Crtanje šasije kamiona
        g2.setPaint(Color.RED);
        g2.fill(new Rectangle2D.Double(50,100,120,80));
        g2.fill(new Rectangle2D.Double(170,130,80,50));

        // Crtanje kabine kamiona
        Polygon trougao = new Polygon();
        trougao.addPoint(170,100);
        trougao.addPoint(170,130);
        trougao.addPoint(200,130);
        g2.setPaint(Color.YELLOW);
```

```
g2.fillPolygon(trougao);

// Crtanje zadnjeg točka
g2.setPaint(Color.DARK_GRAY);
g2.fill(new Ellipse2D.Double(70,160,40,40));
g2.setPaint(Color.WHITE);
g2.fill(new Ellipse2D.Double(80,170,20,20));

// Crtanje prednjeg točka
g2.setPaint(Color.DARK_GRAY);
g2.fill(new Ellipse2D.Double(190,160,40,40));
g2.setPaint(Color.WHITE);
g2.fill(new Ellipse2D.Double(200,170,20,20));

/* Crtanje logoa na stranici kamiona */
g2.setFont(new Font("Serif", Font.ITALIC, 25));
g2.setPaint(Color.WHITE);
g2.drawString("Java",70,125);
g2.drawString("prevoznik",70,150);
}

}
```

2.

LISTING 10.2: Šah.java

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

public class Sah {

    public static void main(String[] args) {

        // Konstruisanje okvira
        JFrame okvir = new JFrame("Šahovska tabla");
        okvir.setSize(300, 320);
        okvir.setLocation(100, 200);
        okvir.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);

        okvir.add(new ŠahovskaTabla(240));
        okvir.setVisible(true);
    }
}

class ŠahovskaTabla extends JComponent {

    private int širinaPolja;
```

```

private int X0 = 20; // početna x-koordinata table unutar okvira
private int Y0 = 20; // početna y-koordinata table unutar okvira

public ŠahovskaTabla(int širinaTable) {
    širinaPolja = širinaTable / 8;
}

public void paintComponent(Graphics g) {

    Graphics2D g2 = (Graphics2D) g;

    for (int i = 0; i < 8; i++) {
        // y-koordinata polja u i-tom redu
        int yi = Y0 + (i * širinaPolja);
        for (int j = 0; j < 8; j++) {
            // x-koordinate polja u i-tom redu i j-toj koloni
            int xj = X0 + (j * širinaPolja);
            if (((i + j) % 2) == 0) // belo polje?
                g2.setPaint(new Color(0.7f, 0.3f, 0.0f));
            else
                g2.setPaint(Color.YELLOW);
            g2.fillRect(xj, yi, širinaPolja, širinaPolja);
        }
    }
}
}

```

## 3.

LISTING 10.3: Ribice.java

```

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

public class Ribice {

    public static void main(String[] args) {

        Akvarijum a = new Akvarijum(400, 350);
        SlikaAkvarijuma slika = new SlikaAkvarijuma(a);

        // Konstruisanje okvira programa
        JFrame okvir = new JFrame("Akvarijum");
        okvir.setSize(400, 350);
        okvir.setLocation(100, 100);
        okvir.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
    }
}

```

```
okvir.add(slika);
okvir.setVisible(true);
}

}

class Riba {

    private Color boja;          // boja ribe
    private Point2D refTačka;    // gornji levi ugao ribe
    private int dužina;          // dužina ribe
    private int visina;          // visina ribe
    private boolean desniSmer;   // smer kretanja ribe

    // Konstruktor
    public Riba(Color boja, Point2D refTačka,
                int dužina, int visina, boolean desniSmer) {
        this.boja = boja;
        this.refTačka = refTačka;
        this.dužina = dužina;
        this.visina = visina;
        this.desniSmer = desniSmer;
    }

    public void nacrtaj(Graphics2D g2d) {
        // Boja ribe
        g2d.setPaint(boja);

        // Određivanje parametara za crtanje delova ribe
        double refTačkaX = refTačka.getX();
        double refTačkaY = refTačka.getY();
        double desnaIvicaRibe = refTačkaX + dužina;
        double donjaIvicaRibe = refTačkaY + visina;
        double centarRibeY = refTačkaY + visina / 2.0;

        // Delovi ribe su proporcionalni
        double dužinaTela = 0.8 * dužina;
        double levaIvicaTela; // zavisi od smera kretanja ribe

        double dužinaOka = 0.1 * dužina;
        double gornjaIvicaOka =
            centarRibeY - (0.1 * dužina) - dužinaOka / 2;
        double levaIvicaOka; // zavisi od smera kretanja ribe

        double dužinaRepa = 0.25 * dužina;
        double visinaRepa = 0.12 * dužina;
        double vrhRepa = centarRibeY - visinaRepa;
```

```
double dnoRepa = centarRibeY + visinaRepa;
double krajRepa; // zavisi od smera kretanja ribe
double dodirTelaRepa; // zavisi od smera kretanja ribe

if (desniSmer) { // riba okrenuta nadesno
    levaIvicaTela = desnaIvicaRibe - dužinaTela;
    levaIvicaOka = desnaIvicaRibe - 0.26 * dužina;
    krajRepa = refTačkaX;
    dodirTelaRepa = krajRepa + dužinaRepa;
}
else {
    levaIvicaTela = refTačkaX;
    levaIvicaOka =
        refTačkaX + (0.26 * dužina) - dužinaOka;
    krajRepa = desnaIvicaRibe;
    dodirTelaRepa = krajRepa - dužinaRepa;
}

// Crtanje ovalnog tela ribe
Ellipse2D.Double telo = new Ellipse2D.Double(
    levaIvicaTela, refTačkaY, dužinaTela, visina);
g2d.fill(telo);

// Crtanje trougaonog repa ribe
GeneralPath konturaRepa = new GeneralPath();
konturaRepa.moveTo(krajRepa, vrhRepa);
konturaRepa.lineTo(krajRepa, dnoRepa);
konturaRepa.lineTo(dodirTelaRepa, centarRibeY);
konturaRepa.closePath();
g2d.fill(konturaRepa);

// Crtanje kružnog oka ribe
g2d.setPaint(Color.DARK_GRAY);
Ellipse2D.Double oko = new Ellipse2D.Double(
    levaIvicaOka, gornjaIvicaOka, dužinaOka, dužinaOka);
g2d.fill(oko);
}

}

class Akvarijum {

    private int dužina, visina;
    private Riba[] ribice = new Riba[4];

    public Akvarijum(int dužina, int visina) {
        this.dužina = dužina;
        this.visina = visina;
```

```
for (int i = 0; i < ribice.length; i++) {
    // Određivanje dužine i visine ribice (visina uvek 50% dužine)
    int d = 40 + i * 15;
    int v = (int)Math.round(0.5*d);
    // Određivanje slučajne boje ribice
    Color b = Color.WHITE;
    switch ((int)(4*Math.random()) + 1) {
        case 1:
            b = Color.YELLOW;
            break;
        case 2:
            b = Color.MAGENTA;
            break;
        case 3:
            b = Color.BLUE;
            break;
        case 4:
            b = Color.RED;
            break;
    }
    // Dodavanje ribica u akvarijum ravnomerno po dužini i visini
    ribice[i] = new Riba(b,
        new Point2D.Double(dužina*i/5+5, visina*i/4+5),
        d, v,
        Math.random()<0.5 ? true : false);
}
}

public void nacrtaj(Graphics2D g2d) {
    for (Riba r : ribice)
        r.nacrtaj(g2d);
}
}

class SlikaAkvarijuma extends JPanel {

    private Akvarijum a;

    public SlikaAkvarijuma(Akvarijum a) {
        this.a = a;
    }

    public void paintComponent(Graphics g) {

        Graphics2D g2d = (Graphics2D) g;

        // Bojenje pozadine
```

```
        g2d.setPaint(Color.CYAN);
        g2d.fill(new Rectangle2D.Double(0,0,getWidth(),getHeight()));

        // Crtanje akvarijuma
        a.nacrtaj(g2d);
    }
}
```

4.

LISTING 10.4: Kalkulator.java

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

/**
 * Klasa Kalkulator predstavlja jednostavan kalkulator koji može
 * obavljati osnovne operacije (+, -, *, /) sa realnim brojevima.
 * Klasa sadrži i metod main() i metod init(), pa se program može
 * izvršiti i samostalno i kao applet. (Dimenzije apleta treba da
 * budu oko 250 x 150 piksela.)
 */
public class Kalkulator extends JApplet {

    // Glavni metod za samostalni program
    public static void main(String[] args) {
        JFrame okvir = new JFrame("Jednostavan kalkulator");
        okvir.add(new PanelKalkulatora());
        okvir.pack(); // veličina okvira prema "pakovanom" sadržaju
        okvir.setLocation(100,100);
        okvir.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE );
        okvir.setVisible(true);
    }

    // Glavni metod za applet
    public void init() {
        add(new PanelKalkulatora());
    }

    // Ugnježđena klasa panela kalkulatora koja istovremeno predstavlja
    // rukovalac događaja za dugmad operacija u njemu
    public static class PanelKalkulatora extends JPanel
        implements ActionListener {

        private JTextField xPolje, yPolje; // ulazna polja za brojeve
        private JLabel rezultat;           // oznaka za prikaz rezultata
    }
}
```

```
public PanelKalkulatora() {  
  
    // Siva pozadine panela radi razmaka između komponenti  
    setBackground(Color.GRAY);  
  
    // Prazna ivica oko panela koja se vidi u boji pozadine  
    setBorder(BorderFactory.createEmptyBorder(5,5,5,5));  
  
    // Konstruisanje ulaznih polja  
    xPolje = new JTextField("0", 10);  
    xPolje.setBackground(Color.WHITE);  
    yPolje = new JTextField("0", 10);  
    yPolje.setBackground(Color.WHITE);  
  
    // Konstruisanje panela za ulazna polja i oznake.  
    // U njima se za komponente koristi podrazumevani FlowLayout.  
    JPanel xPanel = new JPanel();  
    xPanel.add(new JLabel(" x = "));  
    xPanel.add(xPolje);  
  
    JPanel yPanel = new JPanel();  
    yPanel.add(new JLabel(" y = "));  
    yPanel.add(yPolje);  
  
    // Konstruisanje panela za četiri dugmeta osnovnih operacija.  
    // U panelu se za komponente koristi GridLayout kako bi  
    // dugmad bila iste veličine i ispunila ceo panel.  
    JPanel dPanel = new JPanel();  
    dPanel.setLayout(new GridLayout(1,4));  
  
    JButton sabDugme = new JButton("+");  
    sabDugme.addActionListener(this);  
    dPanel.add(sabDugme);  
  
    JButton oduDugme = new JButton("-");  
    oduDugme.addActionListener(this);  
    dPanel.add(oduDugme);  
  
    JButton mnoDugme = new JButton("*");  
    mnoDugme.addActionListener(this);  
    dPanel.add(mnoDugme);  
  
    JButton deljDugme = new JButton("/");  
    deljDugme.addActionListener(this);  
    dPanel.add(deljDugme);
```

```
// Konstruisanje oznake za prikazivanje rezultata.  
// Rezultat se prikazuje crvenom bojom na beloj pozadini,  
// stoga je oznaka neprozirna (opaque).  
rezultat = new JLabel("x + y = 0", JLabel.CENTER);  
rezultat.setForeground(Color.red);  
rezultat.setBackground(Color.white);  
rezultat.setOpaque(true);  
  
// U glavnom panelu se za komponente koristi GridLayout  
setLayout(new GridLayout(4,1,3,3));  
add(xPanel);  
add(yPanel);  
add(dPanel);  
add(rezultat);  
  
// Fokus na početku prirodno dobija prvo ulazno polje  
xPolje.requestFocus();  
  
} // kraj konstruktora  
  
/**  
 * Rukovalac dugmadi za operacije kalkulatora:  
 * pretvoriti brojeve iz ulaznih polja;  
 * izvršiti odgovarajuću operaciju prema kliknutom dugmetu;  
 * prikazati rezultat u polju izlazne oznake.  
 * U slučaju greške, odgovarajuća poruka se takođe prikazuje  
 * u polju izlazne oznake.  
 */  
public void actionPerformed(ActionEvent događaj) {  
  
    double xBroj, yBroj; // brojevi u ulaznim poljima  
  
    try {  
        String xString = xPolje.getText();  
        xBroj = Double.parseDouble(xString);  
    }  
    catch (NumberFormatException ex) {  
        // String xString nije broj  
        rezultat.setText("Greška: x nije broj!");  
        xPolje.requestFocus();  
        return;  
    }  
  
    try {  
        String yString = yPolje.getText();  
        yBroj = Double.parseDouble(yString);  
    }
```

```
        catch (NumberFormatException ex) {
            rezultat.setText("Greška: y nije broj!");
            yPolje.requestFocus();
            return;
        }

        // Izvršavanje odgovarajuće operacije kliknutog dugmeta.
        // Dugme koje je kliknuto prepoznaje se prema "akcijskoj
        // komandi" nastalog događaja.
        String op = dogadjaj.getActionCommand();
        if (op.equals("+"))
            rezultat.setText("x + y = " + (xBroj+yBroj));
        else if (op.equals("-"))
            rezultat.setText("x - y = " + (xBroj-yBroj));
        else if (op.equals("*"))
            rezultat.setText("x * y = " + (xBroj*yBroj));
        else if (op.equals("/")) {
            if (yBroj == 0)
                rezultat.setText("Greška: deljenje nulom!");
            else
                rezultat.setText("x / y = " + (xBroj/yBroj));
        }
    }
}
```

## 5.

LISTING 10.5: DigitalniSat.java

```
import java.awt.*;
import java.awt.event.*;
import javax.swing.*;           // koristi se javax.swing.Timer,
import java.util.Calendar; // a ne java.util.Timer

public class DigitalniSat {

    public static void main(String[] args) {
        JFrame okvir = new JFrame("Digitalni sat");
        okvir.add(new PanelDigitalnogSata());
        okvir.pack(); // veličina okvira prema "pakovanom" sadržaju
        okvir.setLocation(100,100);
        okvir.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
        okvir.setVisible(true);
    }
}

class PanelDigitalnogSata extends JPanel {
```

```

private JTextField vreme; // ažurira se rukovaocem tajmera

// Konstruktor
public PanelDigitalnogSata() {
    vreme = new JTextField(6);
    vreme.setFont(new Font(Font.SANS_SERIF, Font.PLAIN, 48));
    vreme.setEditable(false);
    vreme.getCaret().setVisible(false);

    add(vreme);

    // Konstruisanje tajmera sa intervalom od 1 sekunde,
    // kao i njegovog anonimnog rukovaoca
    Timer t = new Timer(1000, new ActionListener() {
        public void actionPerformed(ActionEvent e) {
            Calendar sada = Calendar.getInstance();
            int č = sada.get(Calendar.HOUR_OF_DAY);
            int m = sada.get(Calendar.MINUTE);
            int s = sada.get(Calendar.SECOND);
            vreme.setText("'" + č + ":" + m + ":" + s);
        }
    });
    t.start(); // startovanje tajmera
} // kraj konstruktora
}

```

## 6.

LISTING 10.6: ProricanjeSudbine.java

```

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

< /**
 * Program prikazuje slučajnu sudbinu iz niza sudbina.
 */
public class ProricanjeSudbine {

    public static void main(String[] args) {
        JFrame okvir = new JFrame("Proricanje sudbine");
        okvir.add(new ProricačSudbine());
        okvir.pack();
        okvir.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
        okvir.setVisible(true);
    }
}

```

```
class ProricačSudbine extends JPanel implements ActionListener {
    private String[] sudbine = {
        "Uplatite loto i sedmica je vaša!",
        "Uskoro ćete upoznati zgodnu osobu.",
        "Problemi na poslu ili u školi.",
        "Dobićete ocenu 10 iz Java programiranja.",
        "Uskoro ćete oputovati u toplije krajeve.",
        "Auto će vam se pokvariti.",
        "Za rođendan ćete dobiti poklon od voljene osobe.",
        "Neko vam radi iza leđa.",
        "Čuvajte se neiskrenih prijatelja.",
        "Poklonite pažnju voljenoj osobi.",
        "Uskoro vas očekuje povišica.",
        "Posetite zubara da ne biste zažalili.",
        "Ostavite alkohol i pušenje.",
        "Manje jedite zaljučenu hranu.",
        "Kupite sebi nešto da popravite raspoloženje."
    };
    private JButton dugmeSledeća;      // dugme za sledeću sudbinu
    private JLabel oznakaSudbine;      // vodeći tekst sudbine
    private JTextField poljeSudbine;   // tekst poruke sudbine

    // Konstruktor
    public ProricačSudbine() {

        setLayout(new BorderLayout()); // raspored komponenti u panelu

        dugmeSledeća = new JButton("Sledeća");
        oznakaSudbine = new JLabel("Vaša sudbina: ");
        poljeSudbine = new JTextField(slučajnaSudbina(), 25);
        poljeSudbine.setEditable(false);

        // Panel za oznaku i poruku sudbine
        JPanel panelSudbine = new JPanel();
        panelSudbine.add(oznakaSudbine);
        panelSudbine.add(poljeSudbine);

        // Panel za dugme sledeće sudbine
        JPanel panelDugmeta = new JPanel();
        dugmeSledeća.addActionListener(this);
        panelDugmeta.add(dugmeSledeća);

        // Dodavanje panela na odgovarajuće mesto
        add(panelSudbine,BorderLayout.NORTH);
        add(panelDugmeta,BorderLayout.SOUTH);
    }
}
```

```
// Biranje slučajne subbine iz niza subbine
public String slučajnaSudbina() {
    int i = (int)(Math.random() * subbine.length);
    return subbine[i];
}

// Obrada dogadaja pritiska na dugme
public void actionPerformed(ActionEvent d) {
    poljeSudbine.setText(slučajnaSudbina());
}
}
```

## 7.

LISTING 10.7: PomeranjeKvadrata.java

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

public class PomeranjeKvadrata extends JApplet {

    // Glavni metod samostalnog programa
    public static void main(String[] args) {
        JFrame okvir = new JFrame("Pokretni kvadrat");
        okvir.add(new PokretniKvadrat());
        okvir.setSize(300,300);
        okvir.setLocation(100,100);
        okvir.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
        okvir.setVisible(true);
    }

    // Glavni metod apleta
    public void init() {
        add(new PokretniKvadrat());
    }

    public static class PokretniKvadrat extends JPanel
        implements KeyListener, FocusListener, MouseListener {

        private static final int STRANA_KVADRATA = 40;

        private Color bojaKvadrata;

        private int xKvadrata, yKvadrata; // koordinate kvadrata

        // Konstruktor
    }
}
```

```
public PokretniKvadrat() {  
  
    xKvadrata = 100;           // početna pozicija kvadrata  
    yKvadrata = 100;  
    bojaKvadrata = Color.BLUE; // početna boja kvadrata  
  
    setBackground(Color.WHITE);  
  
    addKeyListener(this);      // rukovalac događaja panela  
    addFocusListener(this);    // jeste sam panel  
    addMouseListener(this);  
}  
  
// Crtanje ivice, kvadrata i poruke u panelu.  
// Poruka i boja ivice zavise od toga da li panel  
// ima ulazni fokus.  
public void paintComponent(Graphics g) {  
  
    Graphics2D g2 = (Graphics2D) g;  
  
    super.paintComponent(g2); // bojenje pozadine panela  
  
    // Crtanje ivice (3 piksela) u plavo-zelenoj boji ako panel  
    // ima ulazni fokus ili u bledo sivoj boji ako nema  
    if (hasFocus())  
        g2.setPaint(Color.CYAN);  
    else  
        g2.setPaint(Color.LIGHT_GRAY);  
  
    int širina = getSize().width; // širina panela  
    int visina = getSize().height; // visina panela  
    g2.drawRect(0,0,širina-1,visina-1);  
    g2.drawRect(1,1,širina-3,visina-3);  
    g2.drawRect(2,2,širina-5,visina-5);  
  
    // Crtanje kvadrata  
    g2.setPaint(bojaKvadrata);  
    g2.fillRect(xKvadrata, yKvadrata,  
                STRANA_KVADRATA, STRANA_KVADRATA);  
  
    // Crtanje poruke prema tome da li panel ima ulazni fokus  
    g2.setPaint(Color.RED);  
    if (hasFocus()) {  
        g2.drawString(  
            "Tasteri sa strelicama pomeraju kvadrat",10,20);  
        g2.drawString(  
            "Tasteri S, C, Z, P menjaju boju kvadrata",10,40);  
    }  
}
```

```
    }
    else
        g2.drawString(
            "Kliknite mišem radi aktiviranja prozora",10,20);
}

// ----- Metodi rukovaoca događaja -----

// Metod koji se poziva kada panel dobija ulazni fokus.
// Metod samo poziva repaint() da bi se panel prikazao sa
// rezedo ivicom i odgovarajućom porukom.
public void focusGained(FocusEvent dogadjaj) {
    repaint();
}

// Metod koji se poziva kada panel izgubi ulazni fokus.
// Metod samo poziva repaint() da bi se panel prikazao sa
// bleđo sivom ivicom i odgovarajućom porukom.
public void focusLost(FocusEvent dogadjaj) {
    repaint();
}

// Metod koji se poziva kada se pritisne taster na
// tastaturi dok panel ima ulazni fokus.
// Kvadrat menja boju u sivu, crvenu, zelenu ili plavu prema
// tome da li su pritisnuti tasteri S, C, Z, P.
public void keyTyped(KeyEvent dogadjaj) {

    char taster = dogadjaj.getKeyChar(); // slovo tastera

    if (taster == 'S' || taster == 's') {
        bojaKvadrata = Color.GRAY;
        repaint(); // nacrtati panel sa novom bojom
    }
    else if (taster == 'C' || taster == 'c') {
        bojaKvadrata = Color.RED;
        repaint();
    }
    else if (taster == 'Z' || taster == 'z') {
        bojaKvadrata = Color.GREEN;
        repaint();
    }
    else if (taster == 'P' || taster == 'p') {
        bojaKvadrata = Color.BLUE;
        repaint();
    }
}
```

```
// Metod koji se poziva kada se pritisne taster na
// tastaturi dok panel ima ulazni fokus.
// Kvadrat se pomera prema smeru tastera sa strelicama,
// ali se ne može pomeriti izvan panela.
public void keyPressed(KeyEvent dogadjaj) {

    int taster = dogadjaj.getKeyCode(); // kodni broj tastera

    if (taster == KeyEvent.VK_LEFT) {
        xKvadrata -= 8;
        if (xKvadrata < 3)
            xKvadrata = 3;
        repaint();
    }
    else if (taster == KeyEvent.VK_RIGHT) {
        xKvadrata += 8;
        if (xKvadrata > getWidth() - 3 - STRANA_KVADRATA)
            xKvadrata = getWidth() - 3 - STRANA_KVADRATA;
        repaint();
    }
    else if (taster == KeyEvent.VK_UP) {
        yKvadrata -= 8;
        if (yKvadrata < 3)
            yKvadrata = 3;
        repaint();
    }
    else if (taster == KeyEvent.VK_DOWN) {
        yKvadrata += 8;
        if (yKvadrata > getHeight() - 3 - STRANA_KVADRATA)
            yKvadrata = getHeight() - 3 - STRANA_KVADRATA;
        repaint();
    }
}

// Metod koji se poziva kada se otpusti pritisnut taster na
// tastaturi dok panel ima ulazni fokus.
// Metod ne radi ništa, ali je neophodan jer se nalazi u
// interfejsu KeyListener.
public void keyReleased(KeyEvent evt) { }

// Metod koji se poziva kada se klikne mišem u panelu.
// Metod samo zahteva ulazni fokus za panel.
public void mousePressed(MouseEvent evt) {
    requestFocus();
}
```

```
// Metodi koji ne rade ništa, ali su neophodni jer se nalaze u
// interfejsu MouseListener.
public void mouseEntered(MouseEvent evt) { }
public void mouseExited(MouseEvent evt) { }
public void mouseReleased(MouseEvent evt) { }
public void mouseClicked(MouseEvent evt) { }
}
}
```

## 8.

LISTING 10.8: IgraKliznaTabla1.java

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

public class IgraKliznaTabla1 {

    public static void main(String[] args) {

        KliznaTabla tabla = new KliznaTabla(4);
        IzgledTable panel = new IzgledTable(tabla);

        JFrame okvir = new JFrame("Klizna tabla");
        int strana = panel.getStranaPanela();
        okvir.setSize(strana + 8, strana + 25);
        okvir.setResizable(false);
        okvir.add(panel);
        okvir.setLocationRelativeTo(null);
        okvir.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
        okvir.setVisible(true);

        Scanner tastatura = new Scanner(System.in);

        System.out.print("Vaš sledeći potez (0 za kraj): ");
        int p = tastatura.nextInt();
        while(p > 0) {
            if (tabla.pomeri(p)) {
                System.out.println("Dozvoljen potez.");
            }
            else
                System.out.println("Nedozvoljen potez!");
            panel.nacrtajTablu();
            System.out.print("Vaš sledeći potez (0 za kraj): ");
            p = tastatura.nextInt();
        }
    }
}
```

```
        System.exit(0);
    }
}

class KliznaTabla {

    private int n;      // veličina klizne table
    private int[][] t; // matrica pločica
    private int iPM;   // vrsta praznog mesta
    private int jPM;   // kolona praznog mesta

    // Konstruktor za konstruisanje početne konfiguracije table
    // sa pločicama u opadajućem numeričkom redosledu.
    public KliznaTabla(int n) {
        this.n = n;
        t = new int[n][n];
        // Pločice se tabli dodaju u „,obrnutom“ redosledu
        int p = n*n - 1;
        for (int i = 0; i < n; i++) {
            for (int j = 0; j < n; j++) {
                t[i][j] = p;
                p--;
            }
        }
        // Pamćenje vrste i kolone u kojima se nalazi prazno mesto
        iPM = n - 1;
        jPM = n - 1;
    }

    // Metod getN() vraća veličinu klizne table
    public int getN() {
        return n;
    }

    // Metod getTabla() vraća matricu pločica
    public int[][] getT() {
        return t;
    }

    // Metod pomeri() pomera pločicu p na prazno mesto.
    // Rezultat metoda je false ukoliko potez nije dozvoljen.
    public boolean pomeri(int p) {

        int i = -1; // vrsta u kojoj se nalazi p
        int j = -1; // kolona u kojoj se nalazi p

        // Da li je pločica p susedna sa praznim mestom?
        boolean dozvoljenPotez = false;
```

```
if (iPM > 0 && t[iPM - 1][jPM] == p) {
    i = iPM - 1;
    j = jPM;
    dozvoljenPotez = true;
}
if (iPM < n - 1 && t[iPM + 1][jPM] == p) {
    i = iPM + 1;
    j = jPM;
    dozvoljenPotez = true;
}
if (jPM > 0 && t[iPM][jPM - 1] == p) {
    i = iPM;
    j = jPM - 1;
    dozvoljenPotez = true;
}
if (jPM < n - 1 && t[iPM][jPM + 1] == p) {
    i = iPM;
    j = jPM + 1;
    dozvoljenPotez = true;
}

if (dozvoljenPotez) {
    t[iPM][jPM] = t[i][j];
    t[i][j] = 0;
    iPM = i;
    jPM = j;
}
return dozvoljenPotez;
}
}

class IzgledTable extends JPanel {
private KliznaTabla tabla;           // tabla koja se prikazuje
private int stranaPločice = 40;      // dužina strane pločice u pikselima
private int stranaTable;            // dužina strane table u pikselima
private int stranaPanela;           // dužina strane panela u pikselima

// Konstruktor
public IzgledTable(KliznaTabla tabla) {
    this.tabla = tabla;
    stranaTable = stranaPločice * tabla.getN();
    stranaPanela = stranaTable + 2 * stranaPločice;
}

public int getStranaPanela() {
    return stranaPanela;
}
```

```
private void nacrtajPločicu(Graphics2D g2d, int p, int i, int j) {  
  
    int x = stranaPločice + (stranaPločice * j);  
    int y = stranaPločice + (stranaPločice * i);  
    if (p == 0) {  
        g2d.setPaint(Color.BLACK);  
        g2d.drawRect(x, y, stranaPločice, stranaPločice);  
        g2d.fillRect(x, y, stranaPločice, stranaPločice);  
    }  
    else {  
        g2d.setPaint(Color.WHITE);  
        g2d.fillRect(x, y, stranaPločice, stranaPločice);  
        g2d.setPaint(Color.BLACK);  
        g2d.drawRect(x, y, stranaPločice, stranaPločice);  
        g2d.drawString(p + "", x + 15, y + 25);  
    }  
}  
  
public void paintComponent(Graphics g) {  
  
    Graphics2D g2d = (Graphics2D)g;  
  
    g2d.setPaint(Color.YELLOW);  
    g2d.fillRect(0, 0, stranaPanela, stranaPanela);  
  
    int[][] t = tabla.getT();  
    for (int i = 0; i < t.length; i++)  
        for (int j = 0; j < t[i].length; j++)  
            nacrtajPločicu(g2d, t[i][j], i, j);  
}  
  
public void nacrtajTablu() {  
    this.repaint();  
}  
}
```

9.

LISTING 10.9: IgraKliznaTabla2.java

```
import javax.swing.*;  
import java.awt.*;  
import java.awt.event.*;  
  
public class IgraKliznaTabla2 {  
  
    public static void main(String[] args) {
```

```
KliznaTabla tabla = new KliznaTabla(4);
IzgledTable panelTable = new IzgledTable(tabla);

JFrame okvir = new JFrame("Klizna tabla");
okvir.setSize(300, 300);
okvir.add(panelTable);
okvir.setLocationRelativeTo(null);
okvir.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
okvir.setVisible(true);
}

}

class KliznaTabla {

    private int n;      // veličina klizne table
    private int[][] t; // matrica pločica
    private int iPM;   // vrsta praznog mesta
    private int jPM;   // kolona praznog mesta

    // Konstruktor za konstruisanje početne konfiguracije table
    // sa pločicama u opadajućem numeričkom redosledu.
    public KliznaTabla(int n) {
        this.n = n;
        t = new int[n][n];
        // Pločice se tabli dodaju u „,obrnutom“ redosledu
        int p = n*n - 1;
        for (int i = 0; i < n; i++)
            for (int j = 0; j < n; j++) {
                t[i][j] = p;
                p--;
            }
        // Pamćenje vrste i kolone u kojima se nalazi prazno mesto
        iPM = n - 1;
        jPM = n - 1;
    }

    // Metod getN() vraća veličinu klizne table
    public int getN() {
        return n;
    }

    // Metod getTabla() vraća matricu pločica
    public int[][] getT() {
        return t;
    }
}
```

```
// Metod pomeri() pomera pločicu p na prazno mesto.  
// Rezultat metoda je false ukoliko potez nije dozvoljen.  
public boolean pomeri(int p) {  
  
    int i = -1; // vrsta u kojoj se nalazi p  
    int j = -1; // kolona u kojoj se nalazi p  
  
    // Da li je pločica p susedna sa praznim mestom?  
    boolean dozvoljenPotez = false;  
    if (iPM > 0 && t[iPM - 1][jPM] == p) {  
        i = iPM - 1;  
        j = jPM;  
        dozvoljenPotez = true;  
    }  
    if (iPM < n - 1 && t[iPM + 1][jPM] == p) {  
        i = iPM + 1;  
        j = jPM;  
        dozvoljenPotez = true;  
    }  
    if (jPM > 0 && t[iPM][jPM - 1] == p) {  
        i = iPM;  
        j = jPM - 1;  
        dozvoljenPotez = true;  
    }  
    if (jPM < n - 1 && t[iPM][jPM + 1] == p) {  
        i = iPM;  
        j = jPM + 1;  
        dozvoljenPotez = true;  
    }  
  
    if (dozvoljenPotez) {  
        t[iPM][jPM] = t[i][j];  
        t[i][j] = 0;  
        iPM = i;  
        jPM = j;  
    }  
    return dozvoljenPotez;  
}  
}  
  
class DugmePločice extends JButton implements ActionListener {  
  
    private KliznaTabla tabla; // model table kojoj pripada pločica  
    private IzgledTable panel; // izgled table kojoj pripada pločica  
  
    // Konstruktor  
    public DugmePločice(KliznaTabla tabla, IzgledTable panel) {
```

```
    this.tabla = tabla;
    this.panel = panel;
    addActionListener(this);
}

public void actionPerformed(ActionEvent d) {

    String oznaka = getText();
    if (!oznaka.equals("")) { // nije kliknuto na prazno mesto
        int p = Integer.parseInt(oznaka); // broj na kliknutom dugmetu
        tabla.pomeri(p);
        panel.nacrtaj();
    }
}
}

class IzgledTable extends JPanel {

    private KliznaTabla tabla;           // tabla koja se prikazuje
    private DugmePločice[][] tablaDugmadi; // matrica dugmadi table

    // Konstruktor
    public IzgledTable(KliznaTabla tabla) {

        this.tabla = tabla;
        int n = tabla.getN();
        tablaDugmadi = new DugmePločice[n][n];

        setLayout(new GridLayout(n, n));

        for (int i = 0; i < n; i++)
            for (int j = 0; j < n; j++) {
                tablaDugmadi[i][j] = new DugmePločice(tabla, this);
                add(tablaDugmadi[i][j]);
            }
        nacrtaj();
    }

    public void nacrtaj() {

        int n = tabla.getN();
        int[][] t = tabla.getT();

        for (int i = 0; i < n; i++)
            for (int j = 0; j < n; j++) {
                if (t[i][j] == 0) {
                    tablaDugmadi[i][j].setBackground(Color.BLACK);
                }
            }
    }
}
```

```
        tablaDugmadi[i][j].setText("");
    }
    else {
        tablaDugmadi[i][j].setBackground(Color.YELLOW);
        tablaDugmadi[i][j].setText("") + t[i][j]);
    }
}
}
```

## 10.

LISTING 10.10: Editor.java

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

public class Editor extends JFrame {

    public static void main(String[] args) {
        JFrame okvir = new OkvirEditora();
        okvir.setSize(700, 500);
        okvir.setLocationRelativeTo(null);
        okvir.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
        okvir.setVisible(true);
    }
}

class OkvirEditora extends JFrame {

    private static final String opisPrograma =
        "Vrlo jednostavan editor koji pokazuje osnovne mogućnosti\n" +
        "za rad sa menijima u Javi. U programskoj traci za menije\n" +
        "se nalaze dva menija \"Datoteka\" i \"Pomoć\" koji sadrže\n" +
        "dodatne opcije za rad sa tekstualnom datotekom.\n" +
        "Verzija: 0.1alfa";

    private JTextArea poljeTeksta; // polje za editovanje teksta
    private File datoteka; // datoteka koja se edituje

    // Konstruktor za konstruisanje glavnog prozora sa menjem i
    // poljem u kojem se može unositi i menjati tekst.
    public OkvirEditora() {
        super("Jednostavan editor: nesačuvan tekst");
        try {
```

```
UIManager.setLookAndFeel(
    UIManager.getSystemLookAndFeelClassName());
}
catch(Exception ex) { }
setIconImage(Toolkit.getDefaultToolkit().getImage(
    getClass().getResource("images/Notepad.gif")));
setJMenuBar(napraviMeni());
poljeTeksta = new JTextArea(25,50);
poljeTeksta.setMargin(new Insets(3,5,0,0)); // prostor oko teksta
JScrollPane pomeračTeksta = new JScrollPane(poljeTeksta);
setContentPane(pomeračTeksta);
}

// Konstruisanje programske trake za menije koja sadrži menije
//   Datoteka za opcije Nova, Otvori, Sačuvaj i Kraj;
//   Pomoć za opciju Opis.
private JMenuBar napraviMeni() {

    // Rukovalac za opcije menija
    ActionListener rukovalac = new ActionListener() {

        // Metod koji se poziva kada se izabere opcija iz menija
        public void actionPerformed(ActionEvent dogadjaj) {
            String opcija = dogadjaj.getActionCommand();
            if (opcija.equals("Nova"))
                počni();
            else if (opcija.equals("Otvori ..."))
                otvori();
            else if (opcija.equals("Sačuvaj ..."))
                sačuvaj();
            else if (opcija.equals("Kraj"))
                završi();
            else if (opcija.equals("Opis"))
                opiši();
        }
    };
}

JMenu meniDatoteka = new JMenu("Datoteka");
meniDatoteka.setMnemonic('D');

 JMenuItem opcijaNova = new JMenuItem("Nova");
 opcijaNova.setMnemonic('N');
 opcijaNova.setAccelerator(KeyStroke.getKeyStroke(
     KeyEvent.VK_N, ActionEvent.CTRL_MASK));
 opcijaNova.setIcon(new ImageIcon(
     getClass().getResource("images/New.gif")));
 opcijaNova.addActionListener(rukovalac);
```

```
meniDatoteka.add(opcijaNova);

JMenuItem opcijaOtvori = new JMenuItem("Otvori ...");
opcijaOtvori.setMnemonic('O');
opcijaOtvori.setAccelerator(KeyStroke.getKeyStroke(
    KeyEvent.VK_O, ActionEvent.CTRL_MASK));
opcijaOtvori.setIcon(new ImageIcon(
    getClass().getResource("images/Open.gif")));
opcijaOtvori.addActionListener(rukovalac);
meniDatoteka.add(opcijaOtvori);

JMenuItem opcijaSačuvaj = new JMenuItem("Sačuvaj ...");
opcijaSačuvaj.setMnemonic('S');
opcijaSačuvaj.setAccelerator(KeyStroke.getKeyStroke(
    KeyEvent.VK_S, ActionEvent.CTRL_MASK));
opcijaSačuvaj.setIcon(new ImageIcon(
    getClass().getResource("images/Save.gif")));
opcijaSačuvaj.addActionListener(rukovalac);
meniDatoteka.add(opcijaSačuvaj);

meniDatoteka.addSeparator();

JMenuItem opcijaZavrši = new JMenuItem("Kraj");
opcijaZavrši.setMnemonic('K');
opcijaZavrši.setAccelerator(KeyStroke.getKeyStroke(
    KeyEvent.VK_K, ActionEvent.CTRL_MASK));
opcijaZavrši.addActionListener(rukovalac);
meniDatoteka.add(opcijaZavrši);

JMenu meniPomoć = new JMenu("Pomoć");
meniPomoć.setMnemonic('P');

JMenuItem opcijaOpis = new JMenuItem("Opis");
opcijaOpis.setMnemonic('O');
opcijaOpis.setAccelerator(KeyStroke.getKeyStroke(
    KeyEvent.VK_P, ActionEvent.CTRL_MASK));
opcijaOpis.addActionListener(rukovalac);
meniPomoć.add(opcijaOpis);

JMenuBar trakaZaMeni = new JMenuBar();
trakaZaMeni.add(meniDatoteka);
trakaZaMeni.add(meniPomoć);
return trakaZaMeni;

}

private void počni() {
```

```
poljeTeksta.setText("");
datoteka = null;
setTitle("Jednostavan editor: nesačuvan tekst");
}

private void sačuvaj() {
    JFileChooser izbor = new JFileChooser(); // dijalog za izbor
                                                // datoteke
    File izabranaDatoteka; // početno ime datoteke u dijalušu
    if (datoteka == null)
        izabranaDatoteka = new File("ImeDatoteke.txt");
    else
        izabranaDatoteka = new File(datoteka.getName());
    izbor.setSelectedFile(izabranaDatoteka);
    izbor.setDialogTitle("Izaberite datoteku radi čuvanje teksta");
    int opcijaIzbora = izbor.showSaveDialog(this);
    if (opcijaIzbora != JFileChooser.APPROVE_OPTION)
        return; // dijalog je zatvoren bez izabrane datoteke
    izabranaDatoteka = izbor.getSelectedFile();
    if (izabranaDatoteka.exists()) { // zameniti postojeću datoteku?
        int odgovor = JOptionPane.showConfirmDialog(this,
            "Datoteka '" + izabranaDatoteka.getName()
            + "' već postoji.\nDa li želite da je zamenite?",
            "Potvrdite zamenu datoteke",
            JOptionPane.YES_NO_OPTION,
            JOptionPane.WARNING_MESSAGE );
        if (odgovor != JOptionPane.YES_OPTION)
            return; // korisnik ne želi da zameni datoteku
    }
    PrintWriter pisač;
    try {
        pisač = new PrintWriter(new FileWriter(izabranaDatoteka));
        pisač.print(poljeTeksta.getText()); // pisanje teksta iz
                                            // polja u datoteku
        pisač.close();
    }
    catch (Exception ex) {
        JOptionPane.showMessageDialog(this,
            "Greška prilikom otvaranja ili pisanja datoteke:\n" + ex);
        return;
    }
    datoteka = izabranaDatoteka;
    setTitle("Jednostavan editor: " + datoteka.getName());
}

public void otvori() {
    JFileChooser izbor = new JFileChooser(); // dijalog za izbor
```

```
// datoteke
izbor.setDialogTitle("Izaberite datoteku radi otvaranja");
izbor.setSelectedFile(null); // početno prazno ime
// datoteke u dijalogu
int opcijaIzbora = izbor.showOpenDialog(this);
if (opcijaIzbora != JFileChooser.APPROVE_OPTION)
    return; // dijalog je zatvoren bez izabrane datoteke
File izabranaDatoteka = izbor.getSelectedFile();
BufferedReader čitač;
String tekst = "";
try {
    čitač = new BufferedReader(new FileReader(izabranaDatoteka));
    while(čitač.ready())
        tekst += čitač.readLine() + "\n";
    čitač.close();
}
catch (Exception ex) {
    JOptionPane.showMessageDialog(this,
        "Greška prilikom otvaranja ili čitanja datoteke:\n" + ex);
    return;
}
poljeTeksta.setText(tekst);
poljeTeksta.setCaretPosition(0); // kursor ide na početak teksta
datoteka = izabranaDatoteka;
setTitle("Jednostavan editor: " + datoteka.getName());
}

private void završi() {
    System.exit(0);
}

public void opiši() {
    JOptionPane.showMessageDialog(this,
        opisPrograma,
        "Opis programa",
        JOptionPane.INFORMATION_MESSAGE);
}
```

## 11.

LISTING 10.11: Loptice.java

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

```
public class Loptice {  
  
    public static void main(String[] args) {  
        JFrame okvir = new JFrame("Loptice skočice");  
  
        // 10 loptica i 30 ms između frejmova animacije.  
        okvir.add(new LopticeSkočice(10, 30));  
        okvir.pack();  
        okvir.setLocation(100,100);  
        okvir.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);  
        okvir.setResizable(false);  
        okvir.setVisible(true);  
    }  
}  
  
class LopticeSkočice extends JComponent {  
  
    Loptica[] loptice; // niz loptica koje skaču u komponenti  
  
    // Konstruktor u kojem se konstruišu n loptica i tajmer koji  
    // odbrojava u intervalima od ms milisekundi na osnovu kojih  
    // se realizuje animacija. Pored toga, u konstrukturu se  
    // određuje podrazumevana veličina komponente 400 x 400 piksela  
    // i rukovalac pritiska na taster miša.  
    LopticeSkočice(int n, int ms) {  
  
        setPreferredSize(new Dimension(400,400));  
  
        loptice = new Loptica[n];  
        for (int i = 0; i < loptice.length; i++)  
            loptice[i] = new Loptica(0, 400, 0, 400);  
  
        // Ako se klikne u komponenti, sve loptice treba da krenu  
        // prema mestu na kome se nalazi strelica miša.  
        addMouseListener(new MouseAdapter() {  
            public void mousePressed(MouseEvent događaj) {  
                for (Loptica l : loptice)  
                    l.kreniPrema(događaj.getX(), događaj.getY());  
            }  
        });  
  
        // Tajmer kojim se realizuje animacija pozivanjem metoda  
        // repaint() u periodičnim intervalima od ms milisekundi.  
        Timer t = new Timer(ms, new ActionListener() {  
            public void actionPerformed(ActionEvent događaj) {  
                repaint();  
            }  
        });  
    }  
}
```

```
});

t.start(); // startovanje tajmera

} // kraj konstruktora

// Periodično crtanje komponente: sve loptice se najpre
// pomeraju duž njihove trajektorije i zatim crtaju.
public void paintComponent(Graphics g) {

    Graphics2D g2d = (Graphics2D) g;

    // Crtanje pozadine komponente.
    g2d.setPaint(Color.DARK_GRAY);
    g2d.fillRect(0, 0, getWidth(), getHeight());

    // Realizacija jednog frejma animacije.
    for (Loptica l : loptice) {
        l.pomeri();
        l.nacrtaj(g2d);
    }
}

class Loptica {

    private double x, y;          // trenutna pozicija loptice
    private double xmin, xmax;    // horizontalne granice pozicije loptice:
                                // xmin <= x <= xmax
    private double ymin, ymax;    // vertikalne granice pozicije loptice:
                                // ymin <= y <= ymax
    private double dx, dy;        // vektor brzine loptice (brzina+pravac):
                                // pomeranje loptice za dx piksla
                                // horizontalno i dy piksela vertikalno
    private Color boja;          // boja loptice
    private double prečnik;       // prečnik loptice

    // Konstruktor
    public Loptica(double xmin, double xmax, double ymin, double ymax) {
        this.xmin = xmin;
        this.xmax = xmax;
        this.ymin = ymin;
        this.ymax = ymax;
        this.x = (xmin + xmax) / 2; // početna pozicija loptice
        this.y = (ymin + ymax) / 2; // je u centru pravougaonika
        this.prečnik = 6;
    }
}
```

```
this.boja = Color.YELLOW;
double ugao = 2*Math.PI*Math.random(); // slučajan pravac loptice
double brzina = 4+8*Math.random();      // slučajna brzina loptice
dx = Math.cos(ugao) * brzina;
dy = Math.sin(ugao) * brzina;
}

// Crtanje loptice u grafičkom kontekstu.
public void nacrtaj(Graphics2D g2d) {
    g2d.setPaint(boja);
    Ellipse2D loptica = new Ellipse2D.Double(
        x - prečnik, y - prečnik, 2 * prečnik, 2 * prečnik);
    g2d.fill(loptica);
}

// Pomeranje loptice za jednu jedinicu vremena.
public void pomeri() {

    // Ako su granice suviše male, loptica se ne pomera.
    if (xmax - xmin < 2 * prečnik || ymax - ymin < 2 * prečnik)
        return;

    // Određivanje nove pozicije loptice (može biti izvan granica).
    double novoX = x + dx;
    double novoY = y + dy;

    // Ako se nova pozicija loptica nalazi izvan granica jedne od
    // strana pravougla, promena nove pozicije tako da ostane unutar
    // pravougla i bude tačka "refleksije" u odnosu na odgovarajuću
    // stranu pravougla.
    if (novoY < ymin + prečnik) {
        novoY = 2*(ymin + prečnik) - novoY;
        dy = Math.abs(dy);
    }
    else if (novoY > ymax - prečnik) {
        novoY = 2*(ymax - prečnik) - novoY;
        dy = -Math.abs(dy);
    }
    if (novoX < xmin + prečnik) {
        novoX = 2*(xmin + prečnik) - novoX;
        dx = Math.abs(dx);
    }
    else if (novoX > xmax - prečnik) {
        novoX = 2*(xmax - prečnik) - novoX;
        dx = -Math.abs(dx);
    }
}
```

```
x = novoX;
y = novoY;
}

// Promena pravca kretanja loptice prema tački (a,b).
public void kreniPrema(int a, int b) {
    double vx = a - x;
    double vy = b - y;
    double d = Math.sqrt(vx*vx + vy*vy);      // rastojanje od
                                                // (x,y) do (a,b)
    if (d != 0) {
        double s = Math.sqrt(dx*dx + dy*dy); // trenutna brzina loptice
        dx = vx / d * s;
        dy = vy / d * s;
    }
}
}
```

12.

LISTING 10.12: AnalogniSat.java

```
import java.awt.*;
import java.awt.EventQueue;
import java.awt.event.*;
import java.awt.geom.*;
import javax.swing.*;
import javax.swing.Timer;
import java.util.*;

public class AnalogniSat {

    public static void main(String[] args) {
        JFrame okvir = new JFrame();
        okvir.setTitle("Analogni sat");
        okvir.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
        okvir.add(new Sat());
        okvir.pack();
        okvir.setVisible(true);
    }
}

class Sat extends JComponent {

    private Dimension dimSata; // dimenzije komponente
    private int čas;           // čas tačnog vremena
    private int min;           // minut tačnog vremena
    private int sek;           // sekunda tačnog vremena
```

```
private Line2D kazaljkaČas;
private Line2D kazaljkaMin;
private Line2D kazaljkaSek;
private Line2D oznakaČas;
private Ellipse2D krugSata;

// Konstruktor kojim se zadaje prečnik kruga sata
public Sat(int r) {
    dimSata = new Dimension(r, r);

    Point2D p1 = new Point2D.Double(0,0);
    Point2D p2 = new Point2D.Double(0,-r/2.0 + 0.05*r);
    Point2D p3 = new Point2D.Double(0,-r/2.0 + 0.12*r);
    Point2D p4 = new Point2D.Double(0,-r/2.0 + 0.03*r);
    Point2D p5 = new Point2D.Double(0,-r/2.0 + 0.05*r);
    Point2D p6 = new Point2D.Double(0,-r/2.0);

    kazaljkaMin = new Line2D.Double(p1, p2);
    kazaljkaČas = new Line2D.Double(p1, p3);
    kazaljkaSek = new Line2D.Double(p1, p4);

    krugSata = new Ellipse2D.Double(
        0, 0, dimSata.width, dimSata.height);

    oznakaČas = new Line2D.Double(p5, p6);

    GregorianCalendar sada = new GregorianCalendar();
    čas = sada.get(Calendar.HOUR);
    min = sada.get(Calendar.MINUTE);
    sek = sada.get(Calendar.SECOND);

    // Tajmer kojim se realizuje animacija tačnog vremena
    // pozivanjem metoda repaint() u intervalima od 1 sekunde.
    Timer t = new Timer(1000, new ActionListener() {
        public void actionPerformed(ActionEvent dogadaj) {
            sek = sek + 1;
            int m = sek / 60;
            sek = sek % 60;
            min = min + m;
            int č = min/60;
            min = min % 60;
            čas = (čas + č) % 12;
            repaint();
        }
    });
    t.start(); // startovanje tajmera
}
```

```
// Podrazumevani konstruktor kojim se zadaje podrazumevani  
// prečnik kruga sata od 200 piksela.  
public Sat() {  
    this(200);  
}  
  
// Metodi za određivanje dimenzija komponente (Java Bean)  
public void setPreferredSize(Dimension dim) {  
    dimSata = dim;  
}  
  
public Dimension getPreferredSize() {  
    return (new Dimension(dimSata.width + 2, dimSata.height + 2));  
}  
  
public Dimension getMinimumSize() {  
    return dimSata;  
}  
  
public Dimension getMaximumSize() {  
    return (new Dimension(dimSata.width + 5, dimSata.height + 5));  
}  
  
// Periodično crtanje sata sa tačnim vremenom  
public void paintComponent(Graphics g) {  
  
    Graphics2D g2d = (Graphics2D)g;  
    super.paintComponent(g2d);  
  
    AffineTransform staraAT = g2d.getTransform();  
  
    g2d.setRenderingHint(RenderingHints.KEY_ANTIALIASING,  
                         RenderingHints.VALUE_ANTIALIAS_ON);  
  
    int ugaoSek = sek * 6;  
    int ugaoMin = min * 6 + sek/10;  
    int ugaoČas = čas * 30 + min/2;  
  
    // Crtanje kruga sata crnom bojom  
    g2d.setColor(Color.BLACK);  
    g2d.setStroke(new BasicStroke(3));  
    g2d.draw(krugSata);  
  
    // Crtanje pozadine kruga sata žutom bojom  
    g2d.setColor(Color.YELLOW);  
    g2d.fill(krugSata);
```

```
// Crtanje oznaka časova crnom bojom
g2d.setTransform(staraAT);
g2d.setColor(Color.BLACK);
g2d.translate(dimSata.width/2, dimSata.width/2);
for (int i = 0; i < 12; i++) {
    g2d.rotate(30 * Math.PI / 180);
    g2d.setStroke(new BasicStroke(1));
    g2d.draw(oznakaČas);
}

// Crtanje kazaljke časa plavom bojom
g2d.setTransform(staraAT);
g2d.translate(dimSata.width/2, dimSata.width/2);
g2d.rotate(ugaoČas * Math.PI / 180);
g2d.setColor(Color.BLUE);
g2d.setStroke(new BasicStroke(3));
g2d.draw(kazaljkaČas);

// Crtanje kazaljke minuta crvenom bojom
g2d.setTransform(staraAT);
g2d.translate(dimSata.width/2, dimSata.width/2);
g2d.rotate(ugaoMin * Math.PI / 180);
g2d.setColor(Color.RED);
g2d.setStroke(new BasicStroke(2));
g2d.draw(kazaljkaMin);

// Crtanje kazaljke sekunde crnom bojom
g2d.setTransform(staraAT);
g2d.translate(dimSata.width/2, dimSata.width/2);
g2d.rotate(ugaoSek * Math.PI / 180);
g2d.setColor(Color.BLACK);
g2d.setStroke(new BasicStroke(1));
g2d.draw(kazaljkaSek);
}

}
```

Odlukom Senata Univerziteta "Singidunum", Beograd, broj 636/08 od 12.06.2008, ovaj udžbenik je odobren kao osnovno nastavno sredstvo na studijskim programima koji se realizuju na integrisanim studijama Univerziteta "Singidunum".

CIP - Каталогизација у публикацији  
Народна библиотека Србије, Београд

004.438JAVA(075.8)(076)  
004.42:004.738.5(075.8)(076)

**ЖИВКОВИЋ, Дејан, 1956-**

Osnove Java programiranja : zbirka  
pitanja i zadataka sa rešenjima / Dejan  
Živković. - 2. izd. - Beograd : Univerzitet  
Singidunum, 2010 (Loznica : Mladost Grup). -  
VII, 308 str. ; 24 cm

Tiraž 70.

ISBN 978-86-7912-238-4

а) Програмски језик "Java" - Вежбе б)

Интернет - Програмирање - Вежбе

COBISS.SR-ID 172420620

© 2010.

Sva prava zadržana. Ni jedan deo ove publikacije ne može biti reprodukovani u bilo kom  
vidu i putem bilo kog medija, u delovima ili celini bez prethodne pismene saglasnosti  
izdavača.