Dejan Živković

OSNOVE JAVA PROGRAMIRANJA

Zbirka pitanja i zadataka sa rešenjima

UNIVERZITET SINGIDUNUM

Beograd, 2012.

OSNOVE JAVA PROGRAMIRANJA – Zbirka pitanja i zadataka sa rešenjima
drugo izdanje

Autor:
Prof. dr Dejan Živković

Recenzenti:
Prof. dr Dragan Cvetković
Prof. dr Slavko Pešić

Izdavač:
UNIVERZITET SINGIDUNUM
Beograd, Danijelova 32
www.singidunum.ac.rs

Za izdavača:
Prof. dr Milovan Stanišić

Tehnička obrada:
Dejan Živković

Dizajn korica:
Dejan Živković

Godina izdanja:
2012.

Tiraž:
200 primeraka

Štampa:
Mladost Grup
Loznica

ISBN:

Predgovor

Ova zbirka je potpuno prilagođena knjizi *Osnove Java programiranja (treće izdanje)*, Univerzitet Singidunum, jer sadrži odgovore na sva pitanja i rešenja svih zadataka iz te knjige. Zbirka se, međutim, može koristiti i kao samostalni praktikum za vežbanje tokom učenja Java programiranja.

Kod sastavljanja programskih zadataka je učinjen napor kako bi njihovo rešenje zahtevalo samo one elemente programskog jezika Java 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.

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

Dejan Živković
Beograd, Srbija
maj 2012.

Sadržaj

Spisak programa

1

Uvod u Java programiranje

1.1 Odgovori na pitanja

1. Kako se nazivaju fizički delovi od kojih se sastoji računar (procesor, memorija, disk, tastatura, monitor, …)?

   A. Softver B. Hardver C. Operativni sistem D. Windows

2. Kako se nazivaju svi programi u računaru čijim izvršavanjem računar obavlja korisne zadatke za ljude?

   A. Softver B. Hardver C. Operativni sistem D. Windows

3. Šta je „mozak” računara?

   A. Hardver B. Procesor (CPU) C. Memorija D. Disk

4. Od čega se sastoji računarski program, pojednostavljeno rečeno?

   A. Bajtova B. Kontrola C. Naredbi i podataka D. Teksta

5. Koliko bitova ima jedan bajt?

   A. 4 B. 8 C. 16 D. 32

6. Na kom jeziku moraju biti napisani programi da bi računar mogao da ih izvrši?

   A. Mašinskom jeziku B. Engleskom jeziku C. Pseudo jeziku
   D. Jeziku visokog nivoa

7. Šta prevodi programe napisane na programskom jeziku visokog nivoa u programe na mašinskom jeziku?

   A. Operativni sistemi B. Procesori C. Ljudi D. Prevodioci (kompajleri)

8. Kako se naziva program preveden na mašinski jezik Java virtuelne mašine?

   A. Java bajtkod B. Java objektni kod C. Java aplet D. Java aplikacija

9. Kako se mora nazvati datoteka u kojoj se nalazi sledeći Java program koji se sastoji od jedne klase `Test`?

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

   A. Test.txt B. Test.class C. Test.java D. Main.java E. Main.txt

10. Kojim sufiksom se završava ime datoteke u kojoj se nalazi preveden Java program (Java bajtkod)?

    A. java B. obj C. class D. exe

11. Koja DOS komanda služi za prevođenje Java programa koji se nalazi u datoteci Test.java?

    A. `javac Test.java` B. `compile Test.java` C. `prevedi Test.java` D. `javap Test.java`

12. Koja DOS komanda služi za izvršavanje (interpretiranje) prevedenog Java programa u datoteci Test.class?

    A. `javac Test` B. `izvrsi Test` C. `java Test` D. `java Test.class`

13. Koja grafička razvojna okruženja služe za pisanje Java programa?

    A. NetBeans B. DOS C. Windows D. DrJava E. Linux

2

Uvod u Javu

2.1 Odgovori na pitanja

1. Kako se u OOP zovu programske jedinice sa zajedničkim svojstvima koje opisuje jedna klasa?

   A. Promenljive B. Procedure C. Izrazi D. Objekti

2. Kako se u OOP zovu programske jedinice kojima pripadaju objekti?

   A. Promenljive B. Procedure C. Klase D. Nizovi

3. Koje su dve glavne karakteristike svakog objekta u OOP?

   A. Obeležja (atributi) i mogućnosti (ponašanje) objekta.
   B. Sastavni delovi i izgled objekta.
   C. Broj i oblik objekta.
   D. Veličina i način upotrebe objekta.

4. Kako se zovu programske jedinice u kojima su grupisane raspoložive klase Java platforme?

   A. Datoteke B. Paketi C. Moduli D. Folderi

5. Kojom deklaracijom se nekom programu priključuju raspoložive klase Java platforme?

   A. `import` B. `export` C. `module` D. `package`

6. Koje su od ovih rečenica o paketima tačne?

   A. Po konvenciji, u Javi se imena paketa pišu svim malim slovima.

B. Deklaracija `package` nije obavezna.
C. Deklaracija `import` nije obavezna.
D. Klase u paketu `java.lang` se automatski priključuju svakomu programu.

7. Zašto je važan dobar stil programiranja?
   A. Program se neće prevesti zato što je napisan lošim stilom.
   B. Program će se brže izvršavati zato što je napisan dobrim stilom.
   C. Program će biti razumljiviji zato što je napisan dobrim stilom.
   D. Program će imati verovatno manji broj grešaka zato što je napisan dobrim stilom.
   E. Program će se lakše menjati zato što je napisan dobrim stilom.

3

Osnovni elementi Jave

3.1 Odgovori na pitanja

1. U kojem od ovih slučajeva *nije* ispravno napisan komentar u Javi?
   A. `/** tekst komentara */`
   B. `// tekst komentara`
   C. `-- tekst komentara`
   D. `/* tekst komentara */`
   E. `** tekst komentara **`

2. Koje su od ovih reči rezervisane (službene) reči u Javi?
   A. `public` B. `static` C. `void` D. `class` E. `tačno`

3. U kojem su od ovih slučajeva ispravno napisana imena (identifikatori) u Javi?
   A. `9x` B. `ekran` C. `brojStudenata` D. `znak+ili-`

4. U kojem su od ovih slučajeva ispravno napisana imena (identifikatori) u Javi?
   A. `3praseta` B. `tri praseta` C. `prečnik` D. `bzvz`

5. U koje dve kategorije se dele svi mogući tipovi podataka u Javi?
   A. Specifični i generalni B. Celobrojni i realni C. Numerički i tekstualni D. Primitivne i klasni

6. Koji od ovih celobrojnih tipova podataka zahteva najviše memorije za predstavljanje celih brojeva?
   A. `long` B. `int` C. `short` D. `byte`

7. Koji od ovih tipova podataka u Javi služe za predstavljanje realnih brojeva?

A. `float` B. `int` C. `long` D. `double` E. `boolean`

8. Koliko bajtova u memoriji zauzima jedan znak u Javi tipa `char`?

A. Jedan B. Dva C. Tri D. Četiri

9. Koje su moguće logičke vrednosti tipa `boolean` u Javi?

A. `tačno` B. `true` C. `false` D. `0` E. `1` F. `netačno`

10. U kojem su od ovih slučajeva ispravno napisana imena promenljivih prema *konvenciji* u Javi za davanje imena promenljivim?

A. `kredit` B. `Kredit` C. `KREDIT` D. `kamatnaStopa`
E. `KamatnaStopa` F. `kamatna_stopa`

11. Kojim znakom se završava skoro svaka naredba u Javi?

A. tačkom B. tačkom-zapetom C. zapetom D. zvezdicom

12. U kojem su od ovih slučajeva ispravno napisane naredbe za definisanje promenljivih u Javi?

A. `int dužina; int širina;`

B. `int dužina, širina;`

C. `int dužina; širina;`

D. `int dužina, int širina;`

13. U kojem su od ovih slučajeva ispravno napisane naredbe za prikazivanje teksta `Java je kul!` na ekranu?

A. `System.out.println('Java je kul!');`

B. `System.println("Java je kul!");`

C. `System.out.writeln("Java je kul!");`

D. `System.out.println("Java je kul!");`

E. `System.out.print("Java je kul!");`

F. `System.out.printf("Java je kul!");`

14. U kojem su od ovih slučajeva ispravno napisana naredba dodele vrednosti 17 promenljivoj `x`?

A. `17 = x;` B. `x = 17;` C. `x := 17;` D. `x == 17;`

15. Kojom se od ovih naredbi ispravno definiše promenljiva `x` tipa `int` sa početnom vrednošću 17?

A. `int x = '17';`
B. `int x == 17;`
C. `int x = 17;`
D. `int x = 17.0;`

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

A. `float f = -34;`
B. `int t = 23;`
C. `short s = 10;`
D. `int t = (int)false;`
E. `int t = 4.5;`

17. Kojim metodom se u Javi odmah prekida izvršavanje programa?

A. `System.halt(0)`
B. `System.exit(0)`
C. `System.stop(0)`
D. `System.terminate(0)`

18. Koji metod u Javi izračunava broj `x` na stepen `y`?

A. `Math.power(x, y)`
B. `Math.exp(x, y)`
C. `Math.pow(x, y)`
D. `Math.pow(y, x)`

19. Koja je od ovih naredbi definisanja promenljive ispravna u Javi?

A. `char c = 'A';`
B. `char c = '23';`
C. `char c = "A";`
D. `char c = "23";`

20. Koje su od ovih naredbi definisanja promenljive ispravne u Javi?

A. `string s = 'A';`
B. `String s = '23';`
C. `String s = "A";`
D. `String s = "23";`

21. Šta se dodeljuje promenljivoj `c` kao rezultat izvršavanja ovog programskog fragmenta?

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

A. `'a'` B. `'v'` C. `'"'` D. Ništa, zbog greške

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

A. `String s = "neki string";`
B. `String s3 = s1 + s2;`
C. `s1 >= s2`
D. `int i = s1.length;`
E. `s1.charAt(0) = '?';`

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

A. `String s3 = s1 - s2;`
B. `s1 == s2`
C. `boolean b = s1.compareTo(s2);`
D. `char c = s1[0];`
E. `char c = s1.charAt(s1.length());`
F. `char c = s1.charAt(s1.length() - 1);`

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

```
s1.equals(s2) == s2.equals(s1)
```

A. 0 B. 1 C. `true` D. `false`

25. Šta je rezultat izraza `"java" + "program"` u Javi?

A. `javaprogram` B. `Java program` C. `Java_program` D. `Javaprogram` E. Ništa, zbog greške

26. Kojim se od ovih naredbi ispravno pretvara string `s` u celobrojnu vrednost promenljive `i` tipa `int`?

A. `i = Integer.parseInt(s);`
B. `i = (new Integer(s)).intValue();`
C. `i = Integer.valueOf(s).intValue();`
D. `i = Integer.valueOf(s);`

E. `i = (int)(Double.parseDouble(s));`

27. Kojim se od ovih naredbi ispravno pretvara string `s` u realnu vrednost promenljive `d` tipa `double`?

A. `d = Double.parseDouble(s);`

B. `d = (new Double(s)).doubleValue();`

C. `d = Double.valueOf(s).doubleValue();`

D. `d = (double)(Integer.parseInt(s));`

28. Kojim se od ovih naredbi ispravno pretvara realna vrednost promenljive `d` tipa `double` u string `s`?

A. `s = d;`

B. `s = d.toString();`

C. `s = (new Double(d)).toString();`

D. `s = (Double.valueOf(d)).toString();`

3.2 Rešenja zadataka

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

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

Program 3.1: Prikazivanje imena i prezimena

```
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. Napisati program koji na ekranu ispisuje vaše inicijale velikim slovima koja su „nacrtana" zvezdicama (znakom *) u matrici dimenzije 11×12. Na primer, na ekranu treba da se za inicijale DŽ dobije otprilike sledeća slika:

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

Program 3.2: Prikazivanje inicijala

```
public class Inicijali {

   public static void main(String[] args) {

      System.out.println("                        **  **");
      System.out.println("                          **");
      System.out.println("   *******              ************");
      System.out.println("   **     **                    **");
      System.out.println("   **       **                **");
      System.out.println("   **        **              **");
      System.out.println("   **        **             **");
      System.out.println("   **        **            **");
      System.out.println("   **       **           **");
      System.out.println("   **     **           **");
      System.out.println("   *******             ************");
   }
}
```

3. Napisati program koji format papira A4 (210×297 mm) prikazuje u inčima (1 inč = 2.54 cm).

Program 3.3: Dimenzije A4 formata papira u inčima

```
public class A4 {

   public static void main(String[] args) {

      final double INČ_CM = 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/INČ_CM, visinaPapira/INČ_CM);
   }
}
```

4. Napisati program koji Celzijusove stepene pretvara u Farenhajtove po formuli $f = 9c/5 + 32$.

Program 3.4: Pretvaranje Celzijusovih stepena u Farenhajtove

```
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");
```

```
    }
}
```

4

Izrazi

4.1 Odgovori na pitanja

1. Šta je rezultat izraza `45 / 4` u Javi?

   A. 10 B. 11 C. 11.25 D. 12

2. Koji od ovih izraza kao rezultat daju 0.5 u Javi?

   A. `1 / 2`
   B. `1.0 / 2`
   C. `1.0 / 2.0`
   D. `(double) (1 / 2)`
   E. `(double) 1 / 2`
   F. `1 / 2.0`

3. Koji od ovih izraza kao rezultat daje 1 u Javi?

   A. `2%1` B. `15%4` C. `25%5` D. `37%6`

4. 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. `a / b * b`
   B. `a / (b * b)`
   C. `1.0 * a / b * b`
   D. `1.0 * a / (b * b)`
   E. `(double) a / (b * b)`

5. 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);
```

A. `x je 5 i y je 6`
B. `x je 6.0 i y je 6.0`
C. `x je 6 i y je 6`
D. `x je 5.5 i y je 5`
E. `x je 5.5 i y je 5.0`

6. Kojim se od ovih naredbi ispravno dodaje 1 celobrojnoj promenljivi `i` tipa `int`?

A. `i = i + (2 - 1);`
B. `i = i + 1;`
C. `i += 1;`
D. `i = 1 + i;`
E. `i++;`

7. Analizirajte sledeći program:

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

A. Program na ekranu prikazuje `Mesec je 09`.
B. Program na ekranu prikazuje `Mesec je 9`.
C. Program na ekranu prikazuje `Mesec je 9.0`.
D. Program ima grešku, jer `09` nije ispravno napisana oktalna vrednost.

8. Kako se u Javi označava relacijski operator „manje ili jednako”?

A. `<` B. `=<` C. `>=` D. `<=` E. `<<` F. `!=`

9. Kako se u Javi označava relacijski operator „jednako”?

A. `<>` B. `!=` C. `==` D. `=`

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

A. `(true) && (3 => 4)`
B. `!(x > 0) && (x > 0)`
C. `(x > 0) || (x < 0)`
D. `(x != 0) || (x = 0)`
E. `(-10 < x < 0)`

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

A. `1 < x < 100 && x < 0`
B. `((x < 100) && (x > 1)) || (x < 0)`
C. `((x < 100) && (x > 1)) && (x < 0)`
D. `(x != 0) || (x = 0)`
E. `(1 > x > 100) || (x < 0)`
F. `(1 < x < 100) || (x < 0)`

12. Šta je vrednost promenljive `x` posle izračunavanja izraza

```
(y > 10) && (x++ > 10)
```

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

A. 9 B. 10 C. 11 D. 12

13. Šta je vrednost promenljive `x` posle izračunavanja izraza

```
(y > 10) || (x++ > 10)
```

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

A. 9 B. 10 C. 11 D. 12

14. Koju vrednost ima promenljiva `y` posle izvršavanja ovog programskog fragmenta? (*Savet:* operator izbora ima viši prioritet od operatora dodele.)

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

A. -10 B. 0 C. 10 D. 20 E. Ništa, zbog greške

4.2 Rešenja zadataka

1. 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.

Program 4.1: Izračunavanje kamate

```
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);
   }
}
```

2. 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.

Program 4.2: Izračunavanje hipotenuze pravouglog trougla

```
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);
    }
}
```

5

Složene naredbe

5.1 Odgovori na pitanja

1. Kojim se od ovih naredbi ispravno prikazuje površina kruga ako je njegov prečnik `r` pozitivan?

   A. `if (r != 0) System.out.println(r*r*Math.PI);`

   B. `if (r >= 0) System.out.println(r*r*Math.PI);`

   C. `if (r > 0) System.out.println(r*r*Math.PI);`

   D. `if (r > 0) {System.out.println(r*r*Math.PI);}`

   E. `if {r > 0} System.out.println(r*r*PI);`

   F. `if (r <= 0) System.out.println(r*r*Math.PI);`

   G. `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");
```

   A. `x > 0 i y > 0`

   B. `x < 0 i z > 0`

   C. `x < 0 i z < 0`

   D. Ništa se ne prikazuje.

3. Analizirajte sledeći programski fragment:

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

   A. Programski fragment ima grešku i ne može se izvršiti.
   B. Programski fragment se normalno izvršava, ali se ništa ne prikazuje na ekranu.
   C. Programski fragment se normalno izvršava i prikazuje se tekst `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);
```

   A. Fragment A ima grešku.
   B. Fragment B ima grešku.
   C. Oba fragmenta imaju grešku.
   D. 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");
```

   A. Ništa se neće prikazati.
   B. `Plata je veća od 3000`
   C. `Plata je veća od 4000`
   D. `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);
```

A. `cena = 0.0`

B. Ništa se neće prikazati.

C. Programski fragment ima grešku i neće se izvršiti.

D. `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");
   }
}
```

A. Ništa se ne prikazuje.

B. `s1 i s2 ukazuju na isti string`

C. `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");
    }
}
```

A. Ništa se ne prikazuje.

B. `s1 i s2 ukazuju na isti string`

C. `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");
    }
}
```

A. Ništa se ne prikazuje.

B. `s1 i s2 imaju isti sadržaj`

C. `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");
    }
}
```

A. Ništa se ne prikazuje.

B. `s1 i s2 imaju isti sadržaj`

C. `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");
   }
}
```

A. Ništa se ne prikazuje.

B. `s1 i s2 ukazuju na isti string`

C. `s1 i s2 imaju isti sadržaj`

D. `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;
}
```

A. 1 B. 2 C. 3 D. 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);
    }
```

A. `abcd` B. `a` C. `aA` D. `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");
}
```

A. `ocena je 15`

B. `ocena je 15` u jednom redu, a zatim `ocena je 15 ili 30` u sledećem redu.

C. Ništa se neće prikazati.

D. `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;
}
```

A. Programski fragment ima grešku, jer nedostaju naredbe `break` u svim slučajevima.

B. Programski fragment ima grešku, jer nedostaje slučaj `default` u naredbi `switch`.

C. Programski fragment ima grešku, jer kontrolna promenljiva `d` u naredbi `switch` ne može biti tipa `double`.

D. 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);
```

A. Programski fragment ima grešku i neće se izvršiti.

B. `20`

C. Ništa se neće prikazati.

D. 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++;
}
```

A. 9 B. 10 C. 11 D. 0

18. Analizirajte sledeći programski fragment:

```
int brojač = 0;
// Tačka 1
while (brojač < 10) {
   System.out.println("Kako ste?");
   brojač++;
// Tačka 2
}
// Tačka 3
```

A. Uslov `brojač < 10` je uvek tačan u tački 1.
B. Uslov `brojač < 10` je uvek netačan u tački 1.
C. Uslov `brojač < 10` je uvek tačan u tački 2.
D. Uslov `brojač < 10` je uvek netačan u tački 2.
E. Uslov `brojač < 10` je uvek tačan u tački 3.
F. Uslov `brojač < 10` je uvek netačan u tački 3.

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);
```

A. 9 B. 10 C. 11 D. 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);
```

A. Programski fragment ima grešaku i neće se izvršiti.
B. `i = 5`
C. Ništa se neće prikazati.
D. 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;
}
```

A. Programski fragment ima grešku, jer nedostaje treći deo (završnica) u zagradama `for` petlje.
B. Programski fragment ima grešku, jer kontrolna promenljiva `d` u zagradama `for` petlje ne može biti tipa `double`.

C. Pošto je uslov `d < 10` uvek tačan, `for` petlja je beskonačna.

D. 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);
```

A. `10` B. `11` C. `12` D. `13` E. `45`

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

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

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

A. Da B. Ne C. Zavisi

24. Da li je ova `for` petlja sintaksno ispravna?

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

A. Da B. Ne C. Zavisi

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);
   }
}
```

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

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

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

D. U programu je `for` petlja ekvivalentna ovoj petlji:
`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;
}
```

A. Da B. Ne C. Zavisi

27. Koju vrednost sadrži 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);
```

A. 5 B. 6 C. 7 D. 8

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

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

A. Da B. Ne C. Zavisi

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();
```

A. `20 19 18 17 16`

B. `20 15 10 5`

C. `15 10 5 0`

D. `15 10 5`

E. Ništa, zbog greške

30. Koja je sledeća naredba koja se izvršava nakon izvršavanja naredbe `break ponovo` u 6. redu u ovom programskom fragmentu?

```
1   ponovo:
2      for (int i = 1; i < 10; i++) {
3      dalje:
4         for (int j = 1; j < 10; j++) {
5            if (i * j > 50)
6               break ponovo;
7            System.out.println(i * j);
8         }
9      }
10  nastavak:
11     . . .
```

A. Naredba sa oznakom `ponovo` u 1. redu.

B. Naredba sa oznakom `dalje` u 3. redu.

C. Naredba sa oznakom `nastavak` u 10. redu.

D. Nijedna naredba, nego se program odmah završava.

31. Koja je sledeća naredba koja se izvršava nakon izvršavanja naredbe `continue ponovo` u 6. redu u ovom programskom fragmentu?

```
1   ponovo:
2      for (int i = 1; i < 10; i++) {
3      dalje:
4         for (int j = 1; j < 10; j++) {
5            if (i * j > 50)
6               continue ponovo;
7            System.out.println(i * j);
8         // Tačka 1
9         }
10     // Tačka 2
11     }
12  nastavak:
13     . . .
```

A. Kontrola se prenosi u tačku 2 u u 10. redu radi izvršavanja sledeće iteracije spoljašnje petlje sa oznakom `ponovo`.

B. Kontrola se prenosi u tačku 1 u 8. redu radi izvršavanja sledeće iteracije unutrašnje petlje sa oznakom `dalje`.

C. Naredba sa oznakom `nastavak` u 12. redu.

D. Nijedna naredba, nego se program odmah završava.

5.2 Rešenja zadataka

1. Uskrs je pokretni crkveni praznik koji uvek pada u nedelju. Tačan datum Uskrsa se određuje na osnovu prilično komplikovanih crkvenih i astronomskih pravila, a i razlikuje se postupak za izračunavanje pravoslavnog i katoličkog Uskrsa. Ipak, uz razumna ograničenja za vremenski period u kome se traži Uskrs i koristeći celobrojnu aritmetiku, izračunavanje datuma Uskrsa se može svesti na relativno jednostavan postupak. Tako, tačan datum pravoslavnog Uskrsa za datu godinu u periodu između 1900. i 2099. godine dobija se na osnovu sledećih vrednosti:

   a) A je ostatak od deljenja godine sa 4

   b) B je ostatak od deljenja godine sa 7

   c) C je ostatak od deljenja godine sa 19

   d) D je ostatak od deljenja $(19C + 15)$ sa 30

   e) E je ostatak od deljenja $(2A + 4B - D + 34)$ sa 7

   f) F je $D + E + 114$

   Zatim, da bi se dobio datum pravoslavnog Uskrsa po starom julijanskom kalendaru, poslednju vrednosti F treba podeliti sa 31:

   - mesec Uskrsa je rezultat celobrojnog deljenja F sa 31 (pri tome, 4 = april i 5 = maj)
   - dan Uskrsa je za jedan veći od ostatka celobrojnog deljenja F sa 31

   Na kraju, ovaj datum pravoslavnog Uskrsa po starom julijanskom kalendaru treba prevesti na novi gregorijanski kalendar dodavanjem 13 dana. Ovo prilagođavanje novom kalendaru zavisi od države do države, jer su različite države prešle na novi kalendar u različitim godinama. Srbija je, na primer, prešla na novi kalendar 1919. godine.

   Napisati program koji prikazuje datum pravoslavnog Uskrsa u Srbiji za datu godinu u periodu između 1900. i 2099. godine.

Program 5.1: Izračunavanje pravoslavnog Uskrsa u Srbiji

```
import java.util.*;

public class PravoslavniUskrs {

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

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

        System.out.println("Program izračunava datum pravoslavnog ");
        System.out.println("Uskrsa u Srbiji za datu godinu u ");
        System.out.println("periodu od 1900. do 2099. godine.");
        System.out.print("\nUnesite godinu za datum Uskrsa: ");
        int g = tastatura.nextInt();

        int a = g % 4;
        int b = g % 7;
        int c = g % 19;
        int d = (19 * c + 15) % 30;
        int e = (2 * a + 4 * b - d + 34) % 7;
        int f = d + e + 114;

        // Mesec i dan pravoslavnog Uskrsa po julijanskom kalendaru
        int mUskrs = f / 31;
        int dUskrs = f % 31 + 1;

        // Dodavanje 13 dana od 1919. godine kada je Srbija
        // prešla na računanje po gregorijanskom kalendaru
        if (g >= 1919) {
           dUskrs = dUskrs + 13;
           if (mUskrs == 3 && dUskrs > 31) {
              dUskrs = dUskrs % 31;
              mUskrs = 4;
           }
           else if (mUskrs == 4 && dUskrs > 30) {
              dUskrs = dUskrs % 30;
              mUskrs = 5;
           }
        }

        System.out.print("Pravoslavni Uskrs je u nedelju, ");
        System.out.println(dUskrs + "." + mUskrs + "." + g + ".");
    }
}
```

2. Tačan datum katoličkog Uskrsa u periodu između 1982. i 2048. godine dobija se na osnovu sledećeg postupka (pri tome, za ostatak pri celobrojnom deljenju se koristi kraći matematički izraz „po modulu”):

   a) A je godina po modulu 19

   b) B je godina po modulu 4

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

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

Program 5.2: Izračunavanje katoličkog Uskrsa za period 1982–2048

```
import java.util.*;

public class Uskrs {

   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);
   }
}
```

3. 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 se prikazuje datum katoličkog Uskrsa za datu godinu u periodu 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.)

Program 5.3: Izračunavanje katoličkog Uskrsa za period 1900–2099

```
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);
   }
}
```

4. 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.

Program 5.4: Simuliranje bacanja novčića

```
import java.util.*;

public class PismoGlava {

   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);
      }
   }
}
```

5. Napisati program koji simulira eksperiment bacanja dve kocke za igru i određuje koliko puta treba baciti dve kocke dok na obe kocke ne padne istovremeno broj 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.

Program 5.5: Simuliranje bacanja dve kocke za igru dok ne padnu obe jedinice

```
public class ZmijskeOči {

   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++;   // uračunati bacanje
         System.out.printf(
            "%4d. bacanje: kocka1 = %d, kocka2 = %d\n",
            brojBacanja, kocka1, kocka2);
      } while ((kocka1 != 1) || (kocka2 != 1));
   }
}
```

6. Niz brojeva za tzv. „$3n + 1$ problem" počinje od datog celog broja, a svaki sledeći broj niza se dobija sledeći način: ako je prethodni broj n paran, onda je sledeći broj niza jednak broju $n/2$; a ako je prethodni broj n neparan, onda je sledeći broj niza jednak broju $3n + 1$. Ovaj postupak za sledeći broj niza se ponavlja sve dok se na kraju ne dobije broj 1.

   Na primer, ako je početak niza dati broj 6, sledeći broj niza je $6/2 = 3$ pošto je prethodni broj 6 paran. U sledećem koraku, budući da je 3 neparan broj, sledeći broj niza je $3 \cdot 3 + 1 = 10$. Zatim, 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 6 jeste 6, 3, 10, 5, 16, 8, 4, 2, 1.

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

Program 5.6: Niz brojeva za „$3n + 1$ problem"

```
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.println(
              "Greška: početni broj niza mora biti pozitivan!");
           System.out.print("Unesite ponovo 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);
        }
     }
}
```

7. Napisati program kojim se izračunava najveći zajednički delilac dva cela broja tako što se za traženje njihovog zajedničkog delioca počinje od manjeg broja od datih brojeva.

Program 5.7: Izračunavanje NZD-a

```
import java.util.*;

public class NZD {

   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);
    }
}
```

8. 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.)

Program 5.8: Određivanje broja sa najviše delioca

```
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 brojNBD = 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;
                       brojNBD = k;
                }
            }

        System.out.print("Broj sa najvećim brojem delioca je ");
        System.out.println(brojNBD);
        System.out.print("Broj delioca tog broja je ");
        System.out.println(maxBrojDelioca);
    }
}
```

9. 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:

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

Program 5.9: Crtanje romba

```
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();
        }
     }
}
```

10. Napisati program koji učitava jedan red teksta i redom prikazuje njegove znakove brojeći pri tome broj slova i cifara koji se pojavljuju.

Program 5.10: Brojanje slova i cifara jednog reda teksta

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

11. Napisati program koji učitava jedan red teksta i deli ga po rečima. Na primer, ako je ulazni red

```
Profesor reče: "Student je položio ispit".
```

onda se na ekranu 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.

Program 5.11: Deljenje jednog reda teksta po rečima

```
import java.util.*;

public class ListaReci {

   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()));
   }
}
```

6

Metodi

6.1 Odgovori na pitanja

1. Ako metod ne vraća nijednu vrednost, koja se službena reč koristi za njegov tip rezultata u definiciji tog metoda?

   A. `void` B. `return` C. `public` D. `static`

2. Šta je potpis nekog metoda?

   A. Ime metoda B. Ime metoda i lista parametara C. Tip rezultata, ime metoda i lista parametara D. Lista parametara

3. Analizirajte sledeći metod:

```
public double nađi(int x, int y, boolean b) {
    .
    . // Telo metoda
    .
}
```

   A. Ime metoda je `public`.
   B. Ime metoda je `nađi`.
   C. Tip rezultata metoda je `int`.
   D. Tip rezultata metoda je `boolean`.
   E. Tip rezultata metoda je `double`.
   F. Broj parametara metoda je tri.
   G. Broj parametara metoda je šest.

4. Koji metod moraju imati svi Java programi?

A. `public static Main(String[] args)`
B. `public static Main(String args[])`
C. `public void main(String[] args)`
D. `public static void main(String[] args)`
E. `public static main(String[] args)`

5. Kako se navode argumenti nekog metoda u naredbi poziva tog metoda?

A. Unutar uglastih (srednjih) zagrada.
B. Unutar običnih (malih) zagrada.
C. Unutar vitičastih (velikih) zagrada.
D. Unutar dvostrukih apostrofa.
E. Unutar jednostrukih apostrofa.

6. Vrednosti argumenata u pozivu metoda se prenose odgovarajućim parametrima metoda. Kako se naziva ovaj način prenošenja argumenata?

A. Prenošenje po potrebi.
B. Prenošenje po vrednosti.
C. Prenošenje po referenci.
D. Prenošenje po imenu.

7. Koja se naredba koristi unutar tela nekog metoda za povratak iz tog metoda i vraćanje rezultata tog metoda?

A. `void` B. `return` C. `public` D. `static`

8. Da li naredba `return` u ovom metodu izaziva grešku?

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

A. Da B. Ne C. Zavisi

9. Da li poziv metoda `pow()` u ovom metodu izaziva grešku?

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

A. Da B. Ne C. Zavisi

10. Šta je rezultat poziva `nPrint("a",4)` ukoliko je metod `nPrint()` definisan na sledeći način?

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

A. Na ekranu se prikazuje `aaaaa`.
B. Na ekranu se prikazuje `aaaa`.
C. Na ekranu se prikazuje `aaa`.
D. Na ekranu se prikazuje `aa`.
E. Na ekranu se prikazuje `a`.

11. Neka 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);
```

A. 0 B. 1 C. 2 D. 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;
    }
}
```

A. Program na ekranu prikazuje `int` i zatim `5`.

B. Program na ekranu prikazuje `double` i zatim `5`.

C. Program na ekranu prikazuje `double`.

D. Program na ekranu prikazuje `int`.

E. 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(17));
    }

    public int m(int n) {
        return n;
    }

    public void m(int n) {
        System.out.println(n);
    }
}
```

A. Program ima grešku, jer se ne može odrediti koju verziju preopterećenog metoda `m()` treba pozvati.

B. Program ima grešku, jer je druga verzija preopterećenog metoda `m()` definisana ali se nigde ne poziva.

C. Program se normalno izvršava i prikazuje `17` jedanput.

D. Program se normalno izvršava i prikazuje `17` dvaput.

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

A. Globalna promenljiva B. Statička promenljiva C. Blokovska promenljiva D. Lokalna promenljiva

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

```
{
   int k = 1;
   System.out.println(k + 1);
}
```

A. 0 B. 1 C. 2 D. `k` ne postoji van bloka

16. Koje su od ovih rečenica o rekurzivnim metodima tačne?

A. Rekurzivni metodi su oni koji pozivaju sami sebe, bilo direktno ili indirektno.

B. Rekurzivni metodi se pozivaju drugačije od nerekurzivnih metoda.

C. Rekurzivni metodi rešavaju neki zadatak svođenjem polaznog problema na sličan prostiji problem (ili više njih).

D. Svaki rekurzivni metod mora imati *bazni slučaj* za najprostiji zadatak čije se rešenje ne dobija rekurzivnim pozivom.

17. Analizirajte sledeći rekurzivni metod:

```
public long fakt(int n) {
   return n * fakt(n - 1);
}
```

A. Rezultat poziva `fakt(3)` je 2.

B. Rezultat poziva `fakt(3)` je 3.

C. Rezultat poziva `fakt(3)` je 6.

D. Poziv `fakt(3)` izaziva grešku jer proizvodi beskonačan lanac poziva istog metoda `fakt()`.

18. 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);
      }
   }
}
```

A. Program proizvodi beskonačan lanac poziva metoda `rMetod()`.
B. Program na ekranu prikazuje `1 2 3`.
C. Program na ekranu prikazuje `3 2 1`.
D. Program na ekranu prikazuje `1 2`.
E. Program na ekranu prikazuje `2 1`.

19. 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);
      }
   }
}
```

A. Program na ekranu ne prikazuje ništa.
B. Program na ekranu prikazuje `1 2`.
C. Program na ekranu prikazuje `2 1`.
D. Program na ekranu beskonačno prikazuje `1 1 1 1 1 ....`
E. Program na ekranu beskonačno prikazuje `2 2 2 2 2 ....`

20. Analizirajte sledeći rekurzivni metod:

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

A. Pozivom `rMetod(5)` se isti metod `rMetod()` poziva još 3 puta.
B. Pozivom `rMetod(5)` se isti metod `rMetod()` poziva još 4 puta.
C. Pozivom `rMetod(5)` se isti metod `rMetod()` poziva još 5 puta.
D. Pozivom `rMetod(5)` se isti metod `rMetod()` poziva još 6 puta.

21. Analizirajte sledeći rekurzivni metod:

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

A. Rezultat poziva `rMetod(5)` je 5.

B. Rezultat poziva `rMetod(5)` je 10.

C. Rezultat poziva `rMetod(5)` je 15.

D. Rezultat poziva `rMetod(5)` proizvodi beskonačan lanac poziva istog metoda `rMetod()`.

22. Šta je bazni slučaj u ovom rekurzivnom metodu?

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

A. Slučaj kada je n jednako 1.

B. Slučaj kada je n manje od 1.

C. Slučaj kada je n veće od 1.

D. Nema baznog slučaja.

23. 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  }
```

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

B. `(s.charAt(0) != s.charAt(s.length()))`

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

D. `(s.charAt(1) != s.charAt(s.length()))`

24. 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  }
```

A. `palindrom(s)`
B. `palindrom(s, leviKraj, desniKraj)`
C. `palindrom(s, leviKraj + 1, desniKraj)`
D. `palindrom(s, leviKraj, desniKraj - 1)`
E. `palindrom(s, leviKraj + 1, desniKraj - 1)`

25. 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,
8         // kao i njegovog indeksa
9         int iMax = 0;
10        double max = niz[0];
11        for (int i = 1; i <= n; i++)
12           if (niz[i] > max) {
13              max = niz[i];
14              iMax = i;
15           }
16
17        // Zamena najvećeg i poslednjeg elementa niza
18        niz[iMax] = niz[n];
19        niz[n] = max;
```

```
20
21          // Sortiranje prvog dela niza
22          sortiraj(niz, n - 1);
23       }
24    }
```

A. `sortiraj(niz)`

B. `sortiraj(niz, niz.length)`

C. `sortiraj(niz, niz.length - 1)`

D. `sortiraj(niz, niz.length + 1)`

26. 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    }
```

A. `nađiBroj(niz, broj, sredina + 1, leviKraj)`

B. `nađiBroj(niz, broj, sredina - 1, leviKraj)`

C. `nađiBroj(niz, broj, desniKraj, sredina + 1)`

D. `nađiBroj(niz, broj, sredina + 1, desniKraj)`

6.2 Rešenja zadataka

1. Napisati metod `NZD()` kojim se izračunava najveći zajednički delilac dva cela broja koristeći Euklidov algoritam. Taj metod treba testirati pisanjem progra-

ma u kome se u metodu `main()` učitavaju dva cela broja i prikazuje njihov najveći zajednički delilac pozivom metoda `NZD()`.

Program 6.1: Euklidov algoritam za NZD

```
import java.util.*;

public class Euklid {

   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. Napisati metod `triN1()` kojim se prikazuje niz brojeva za „$3n + 1$ problem”

(videti 6. zadatak u poglavlju 5). Taj metod treba testirati pisanjem programa u kome se u metodu `main()` učitava početni broj niza i prikazuje ostatak tog niza pozivom metoda `triN1()`.

Program 6.2: Niz brojeva za „$3n + 1$ problem", druga verzija

```
import java.util.*;

public class Niz3n1 {

   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(
            "Greška: početni broj niza mora biti pozitivan!");
         System.out.print("Unesite ponovo 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. Napisati metod `kapitalizuj()` kojim se početno slovo svake reči datog stringa pretvara u veliko slovo. Taj metod treba testirati pisanjem programa u kome se u metodu `main()` učitava jedan red teksta i prikazuje njegova „kapitalizovana" verzija pozivom metoda `kapitalizuj()`.

Program 6.3: Pretvaranje početnog slova reči u veliko slovo

```
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. Heksadekadne „cifre” su dekadne cifre od 0 do 9 i slova A, B, C, D, E i F. U heksadekadnom sistemu ovi znakovi predstavljaju brojeve od 0 redom do 15. Heksadekadni broj je niz heksadekadnih cifara kao što su $34A7$, FF, $ABCD$ ili 172300.

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

   b) Napisati program kojim se učitava heksadekadni broj i prikazuje dekadna vrednost tog broja pozivom metoda `heksVrednost()`. Ako svi znakovi heksadekadnog broja nisu dozvoljene heksadekadne cifre, program treba da prikaže odgovarajuću poruku o grešci.

Program 6.4: Pretvaranje heksadekadnog broja u dekadni broj

```
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. Napisati program `ZbirDveKocke` kojim se simulira jedan eksperiment sa bacanjem dve kocke za igranje. Program treba da sadrži sledeća tri metoda:

   a) Metod `baciZaZbir()` simulira bacanje dve kocke za igru dok njihov zbir ne bude 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.

   b) Metod `prosekZaZbir()` koristi prethodni metod `baciZaZbir()` za ponavljanje 1000 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 1000 pokušaja.

c) Metod `main()` poziva prethodni metod `prosekZaZbir()` za svaki mogući zbir dve kocke 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
      .                       .
      .                       .
      .                       .
```

Program 6.5: Zbir dve kocke za igranje

```
public class ZbirDveKocke {

   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; // baciti prvu kocku
         kocka2 = (int)(Math.random()*6) + 1; // baciti 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;
    }
}
```

7

Klase

7.1 Odgovori na pitanja

1. Koja od sledećih programskih jedinica predstavlja šablon za konstruisanje objekata istog tipa?

   A. Paket B. Metod C. Promenljiva D. Klasa

2. Kakvi mogu biti članovi klase (polja i metodi)?

   A. Statički (klasni) i nestatički (objektni).
   B. Lokalni i globalni.
   C. Proceduralni i neproceduralni.
   D. Spoljašnji i unutrašnji.

3. Koja se službena reč koristi za definisanje klase?

   A. `method` B. `class` C. `main` D. `object`

4. Kako se naziva specijalni metod neke klase koji se poziva po konstruisanju svakog objekta te klase?

   A. Glavni metod B. Inicijalni metod C. Konstruktor klase
   D. Rekurzivni metod

5. Koje su od ovih rečenica o konstruktorima tačne?

   A. Podrazumevani konstruktor bez parametara se klasi automatski dodaje ukoliko u njoj nije eksplicitno definisan nijedan konstruktor.
   B. U klasi se mora eksplicitno definisati bar jedan konstruktor.
   C. Konstruktori nemaju tip rezultata, čak ni `void`.

D. Konstruktori moraju imati isto ime kao klasa u kojoj se definišu.

E. Konstruktori se pozivaju koristeći operator `new` kada se konstruiše objekat.

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

```
1   public class Test {
2      public static void main(String[] args) {
3         A a = new A();
4         a.prikaži();
5      }
6   }
7
8   class A {
9      String s;
10
11     public A(String s) {
12        this.s = s;
13     }
14     public void prikaži() {
15        System.out.println(s);
16     }
17  }
```

A. Program ima grešku, jer klasa `A` nije javna klasa.

B. Program ima grešku, jer klasa `A` nema podrazumevani konstruktor.

C. Program nema grešaka i normalno se izvršava ništa ne prikazujući na ekranu.

D. Program ima grešku koja se može ispraviti ukoliko se u trećem redu naredba `A a = new A();` zameni naredbom `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) {
      A a = new A(2);
   }
}

class A {
   int i;

   public void m(int j) {
      i = j;
```

```
    }
}
```

A. Program ima grešku, jer klasa `A` nije javna klasa.

B. Program ima grešku, jer klasa `A` nema podrazumevani konstruktor.

C. Program ima grešku, jer klasa `A` nema konstruktor sa parametrom tipa `int`.

D. Program nema grešaka i normalno se izvršava ništa ne prikazujući na ekranu.

8. Pod pretpostavkom da je data definicija `Krug k = new Krug()`, koja je od ovih rečenica najtačnija?

A. Promenljiva `k` sadrži celobrojnu vrednost.

B. Promenljivoj `k` se može dodeliti celobrojna vrednost.

C. Promenljiva `k` sadrži objekat klase `Krug`.

D. 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);
   }
}
```

A. Program ima grešku, jer promenljiva `t` nije inicijalizovana.

B. Program ima grešku, jer promenljiva `x` nije inicijalizovana.

C. Program ima grešku, jer klasa `Test` nema podrazumevani konstruktor.

D. 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`.

E. Program ima grešku, jer promenljiva `t` ima vrednost `null` kada se prikazuje polje `t.x`.

F. Program nema grešaka i normalno se izvršava ništa ne prikazujući na ekranu.

10. Koje su automatske početne vrednosti za polja logičkog, numeričkog i klasnog tipa svakog objekta, tim redom ?

A. `true`, 1, `null`
B. `false`, 0, `null`
C. `true`, 0, `null`
D. `false`, 1, `null`
E. `false`, 0, `void`

11. Koje su od ovih rečenica o promenljivim tačne?

A. Lokalne promenljive metoda ne dobijaju automatski početne vrednosti.
B. Globalne promenljive (polja) objekata dobijaju automatski početne vrednosti.
C. Promenljiva nekog primitivnog tipa sadrži vrednost tog primitivnog tipa.
D. Promenljiva nekog klasnog tipa ukazuje na memorijsku adresu u kojoj se nalazi objekat tog klasnog tipa.
E. Promenljivoj klasnog tipa može se dodeliti ceo broj koji predstavlja važeću memorijsku adresu.

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);
   }
}
```

A. Program ima grešku, jer promenljiva `prečnik` nije inicijalizovana.
B. Program ima grešku, jer je konstanta `PI` definisana unutar metoda.
C. Program ima grešku, jer konstanta `PI` ima previše decimala.
D. Program ima grešku, jer konstanta `PI` ima premalo decimala.
E. 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);
   }
}
```

A. Program ima grešku, jer se metod `System.out.println()` ne može koristiti u konstruktoru klase.

B. Program ima grešku, jer promenljiva `x` nije inicijalizovana.

C. Program ima grešku, jer klasa `Test` nema podrazumevani konstruktor.

D. Program ima grešku, jer se u nekoj klasi ne može konstruisati objekat te iste klase.

E. Program nema grešaka i normalno se izvršava prikazujući `0` na ekranu.

14. Koja je od ovih rečenica o objektima najtačnija?

A. Objektna promenljiva sadrži neki objekat.

B. Promenljiva klasnog tipa sadrži neki objekat.

C. Neki objekat može sadržati druge objekte.

D. Neki objekat može sadržati reference na druge objekte.

15. Koja polja su zajednička i jedinstvena za sve objekte neke klase?

A. Javna B. Privatna C. Objektna (instancna) D. Statička (klasna)

16. Da li se statičko polje neke klase može koristiti bez konstruisanja ijednog objekta te klase?

A. Da B. Ne C. Zavisi

17. U kojem od redova, 5 ili 9, u definiciji sledeće klase treba zameniti znakove `???` službenom rečju `static`?

```
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  }
```

A. Samo u 5. redu. B. Samo u 9. redu. C. U oba reda 5 i 9.
D. U nijednom redu.

18. Kako se naziva metod koji se pridružuje svakom pojedinačnom objektu neke klase?

A. Statički metod B. Klasni metod C. Objektni metod
D. Glavni metod

19. Koji od sledećih načina je ispravan za definisanje konstante `MAX_CENA` kao članice neke klase?

A. `final static MAX_CENA = 99.98;`
B. `final static float MAX_CENA = 99.98;`
C. `static double MAX_CENA = 99.98;`
D. `final double MAX_CENA = 99.98;`
E. `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++;
   }
}
```

A. Program ima grešku, jer metod `xMetod()` ne vraća nijednu vrednost.
B. Program ima grešku, jer metod `xMetod()` nije definisan da bude statički.
C. Program prikazuje `n je 1` na ekranu.
D. Program prikazuje `n je 2` na ekranu.
E. 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++;
   }
}
```

A. `t2.i je 1, t2.s je 1`
B. `t2.i je 1, t2.s je 2`
C. `t2.i je 2, t2.s je 2`
D. `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 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++;
    }
}
```

A. `t3.i je 1, t3.s je 1`

B. `t3.i je 1, t3.s je 2`

C. `t3.i je 1, t3.s je 3`

D. `t3.i je 3, t3.s je 1`

E. `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() {
    }
}
```

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

B. Program ima grešku, jer klasa `A` ima prazan podrazumevani konstruktor.

C. Program ima grešku, jer promenljiva `n` nije inicijalizovana.

D. 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;
   }
}
```

A. `b.n = 101` B. `b.n = 100` C. `b.n = 99` D. `b.n = 98`
E. `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;
    }
}
```

A. `k = 101` B. `k = 100` C. `k = 99` D. `k = 98` E. `k = 0`

26. Analizirajte sledeću klasu `Krug`:

```
public class Krug {

    private double prečnik;

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

A. Klasa `Krug` ima grešku, jer nema metod `main()`.

B. 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.

C. Klasa `Krug` ima grešku, jer se ne može pisati naredba dodele `prečnik = prečnik;` u konstruktoru.

D. 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()");
   }
}
```

A. `this.t()` u konstruktoru `Test(double x)` može se pojednostaviti i zameniti samo sa `t()`.

B. `this.x` u konstruktoru `Test(double x)` može se pojednostaviti i zameniti samo sa `x`.

C. `this(23)` u konstruktoru `Test()` mora se pisati pre naredbe `System.out.println("Podrazumevani konstruktor");`.

D. `this(23)` u konstruktoru `Test()` mora se zameniti tačnijim izrazom `this(23.0)`.

7.2 Rešenja zadataka

1. Napisati program koji prikazuje mesečni kalendar za dati mesec i datu godinu. (*Savet:* korisne klase za rad sa kalendarom su `Calendar` i `GregorianCalendar` iz paketa `java.util`.)

Program 7.1: Mesečni kalendar

```
import java.util.*;

public class MesečniKalendar {
```

```
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("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. 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.

Program 7.2: Tačka, kvadrat i krug u koordinatnom sistemu

```
public class TačkaKvadratKrug {

   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. Napisati klasu `KompleksniBroj` koja predstavlja kompleksni broj sa realnim i imaginarnim delom. Klasa `KompleksniBroj` treba da sadrži metode kojima se realizuju uobičajene operacije sa kompleksnim brojevima. Napisati jednostavan glavni metod `main()` u kojem se testiraju napisani metodi.

Program 7.3: Kompleksni broj

```
public class KompleksniBroj {

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

    // Konstruktor
    public KompleksniBroj(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 kompleksni broj:
// KompleksniBroj z = KompleksniBroj.konjugovanBroj(x)
public static KompleksniBroj konjugovanBroj(KompleksniBroj a) {
   return new KompleksniBroj(a.r, -a.i);
}

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

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

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

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

// Objektni metod za proizvod dva kompleksna broja:
// KompleksniBroj z = x.pomnoži(y);
public KompleksniBroj pomnoži(KompleksniBroj a) {
   return new KompleksniBroj(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) {

       KompleksniBroj x = new KompleksniBroj(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 = " +
          KompleksniBroj.zbir(x, x.konjugovanBroj()));

       System.out.println();

       KompleksniBroj y = new KompleksniBroj(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.pomnoži(y));
    }
}
```

4. Napisati klasu `RimskiBroj` koja predstavlja rimski broj. (Pretpostavite 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.

Program 7.4: Rimski broj

```
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 "";
    }

    // 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. Ponovo uraditi 5. zadatak iz poglavlja 6, ali tako da program sadrži posebnu klasu `KockaZaIgru` koja predstavlja kocku za igranje.

Program 7.5: Zbir dve kocke za igranje, druga verzija

```
public class ZbirDveKocke {

   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();   // baciti prvu kocku
         kocka2.baci();   // baciti drugu kocku
         brojBacanja++;   // uračunati bacanje

      } 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. 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, ...

Program 7.6: Bacanje novčića

```
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;
    }
}
```

8

Nizovi

8.1 Odgovori na pitanja

1. Koje je ime trećeg elementa u nizu pod nazivom `a`?

   A. `a[2]` B. `a(2)` C. `a[3]` D. `a(3)`

2. Koje su od ovih definicija niza `a` *pogrešne*?

   A. `int[] a = new int[2];`
   B. `int[] a = new int(2);`
   C. `int a = new int[2];`
   D. `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]`?

   A. `i`
   B. `(int)(Math.random() * 100)`
   C. `(int)(Math.random() * 100 + 1)`
   D. `Math.random() * 100`
   E. `i + 10`
   F. `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]);
        }
    }
```

A. Program ima grešku, jer je dužina niza `a` premala.

B. Program ima grešku, jer elementi niza `a` nisu inicijalizovani.

C. Program ima grešku, jer element `a[0]` nije definisan.

D. Program nema grešaka i normalno se izvršava prikazujući `a[0] je 0` na ekranu.

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

A. `int i = new int(30);`

B. `double[] d = new double[30];`

C. `int[] i = {3, 4, 3, 2};`

D. `char[] c = new char();`

E. `char[] c = new char{'a', 'b', 'c', 'd'};`

F. `char[] c = {'a', 'b'};`

6. Ako je data deklaracija `int[] a = {1, 2, 3, 4}`, koju vrednost sadrži polje `a.length`?

A. 0 B. 3 C. 4 D. 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] + " ");
    }
}
```

A. Program prikazuje `0 1 2 3 4` na ekranu.

B. Program prikazuje `4` na ekranu.

C. Program ima grešku, jer će se koristiti nepostojeći element `a[5]` u poslednjoj naredbi `print` u metodu `main`.

D. 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] + " ");
   }
}
```

A. Program prikazuje `120 200 16` na ekranu.
B. Program prikazuje `120 200 14` na ekranu.
C. Program prikazuje `120 200 22` na ekranu.
D. 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 = {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] + " ");
   }
}
```

A. `1 2 3` B. `1 1 1` C. `0 1 2` D. `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] + " ");
        }
    }
```

A. 1 2 3 B. 1 1 1 C. 0 1 2 D. 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] + " ");
        }
}
```

A. 1 2 3 4 B. 0 0 C. 0 0 3 4 D. 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] + " ");
        }
}
```

A. 1 2 3 4 B. 0 0 C. 0 0 3 4 D. 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] + " ");
        }
}
```

A. Program prikazuje `1 2 3 4` na ekranu.

B. Program prikazuje `0 0` na ekranu.

C. Program ima grešku kod naredbe `x = new int[2]`, jer je promenljiva `x` deklarisana da bude `final` i ne može se menjati.

D. 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];
```

A. Programski fragment ima grešku, jer se promenljiva `lista` ne može menjati nakon što joj se dodeli vrednost.

B. Programski fragment nema grešaka i drugom naredbom se novi niz dodeljuje promenljivoj `lista`.

C. Programski fragment ima grešku, jer se promenljivoj `lista` dodeljuje novi niz.

D. 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]);
   }
}
```

A. Program ima grešku kod naredbe `a = new int[2]`, jer se novi niz dodeljuje promenljivoj `a`.
B. Program ima grešku kod naredbe `println`, jer `a[1]` nije inicijalizovano.
C. Program na ekranu prikazuje `a[1] je 0`.
D. Program na ekranu prikazuje `a[1] je 1`.

16. Kojom naredbom se kopija niza `a` dodeljuje nizu `b`.

A. `b = Arrays.copyOf(a, a.length);`
B. `b = Arrays.copyOf(a);`
C. `Arrays.copyOf(b, a, a.length);`
D. `Arrays.copyOf(a, b);`

17. Kada se dati niz prenosi nekom metodu kao argument, šta se tačno prenosi tom metodu?

A. Kopija datog niza.
B. Kopija prvog elementa datog niza.
C. Dužina datog niza.
D. 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 uvećaj(int n) {
      n++;
   }
}
```

A. Poruka o grešci B. `1 1` C. `2 2` D. `2 1` E. `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;
   }
}
```

A. `1 2 3 4 5` B. `5 4 3 2 1` C. `5 4 1 2 3` D. `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);
   }
}
```

A. Program ima grešku, jer je u prvom pozivu metoda `xMetod()` nepravilno naveden argument `new double[]{3, 3}`.

B. Program ima grešku, jer je u drugom pozivu metoda `xMetod()` nepravilno naveden argument `new double[5]`.

C. Program ima grešku, jer je u trećem pozivu metoda `xMetod()` nepravilno naveden argument `new double[3]{1, 2, 3}`.

D. Program ima grešku, jer će sve vrednosti niza `a` imati vrednost `null` prilikom izvršavanja drugog poziva metoda `xMetod()`.

21. Kako se zove deo memorije u kojoj se smeštaju nizovi, kao i svi drugi objekti programa? U tom delu se, radi efikasnosti, ne vodi mnogo računa o redu po kojem se zauzima slobodna i oslobađa zauzeta memorija.

A. Stek memorija B. Hip memorija C. Keš memorija D. Virtuelna memorija

22. Kada se niz vraća kao rezultat nekog metoda, šta se tačno prenosi kao rezultat?

A. Kopija tog niza.

B. Kopija prvog elementa tog niza.

C. Dužina tog niza.

D. 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()`?

A. `return 1;`

B. `return {1, 2, 3};`

C. `return int[]{1, 2, 3};`

D. `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;
    }
}
```

A. 1 2 3 4 5 B. 5 4 3 2 1 C. 5 4 1 2 3 D. 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[] lista1 = {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;
    }
}
```

A. 1 2 3 4 5 B. 5 4 3 2 1 C. 5 4 1 2 3 D. 1 2 5 4 3

26. 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);
    }
}
```

A. Program na ekranu prikazuje `1 2 3 4 5`.

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

C. Program na ekranu prikazuje `5 4 3 2 1`.

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

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

A. Promenljiva `k` sadrži niz od 10 celobrojnih vrednosti.

B. Promenljiva `k` sadrži niz od 10 objekata klase `Krug`.

C. Promenljiva `k` sadrži referencu na niz od 10 promenljivih klasnog tipa `Krug`.

D. Promenljiva `k` sadrži objekat klase `Krug` prečnika 10.

28. Ako je zaglavlje glavnog metoda klase `Test` dato 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
```

A. `args[0]` B. `args[1]` C. `args[2]` D. `args[3]`

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

A. `public void prikaži(String... niska, double... broj)`

B. `public void prikaži(double... broj, String ime)`

C. `public void double... prikaži(double d1, double d2)`

D. `public void prikaži(double... broj)`

E. `public void prikaži(int n, double... broj)`

30. 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;
   }
}
```

A. Program ima grešku u prvoj naredbi `println`, jer je nepravilan poziv `prosek(d)`.

B. Program ima grešku u drugoj naredbi `println`, jer je nepravilan poziv `prosek(1, 2, 2, 1, 4)`.

C. Program ima grešku u trećoj naredbi `println`, jer je nepravilan poziv `prosek(new double[] {1, 2, 3})`.

D. Program ima grešku u četvrtoj naredbi `println`, jer je nepravilan poziv `prosek(1.0, 2.0, 2.0, 1.0)`.

E. Program se izvršava bez greške i prosek datih brojeva se tačno izračunava.

F. Program se izvršava bez greške, ali se prosek datih brojeva ne izračunava tačno.

31. Kojim od ovih poziva se sortira niz `lotoBrojevi` tipa `int[]`.

A. `Arrays(lotoBrojevi)`

B. `Arrays.sort(lotoBrojevi)`

C. `Arrays.sorts(lotoBrojevi)`

D. `Arrays.sortArray(lotoBrojevi)`

32. Ako je data deklaracija niza `int[] lotoBrojevi = {5, 8, 17, 23, 27, 33, 36}`, šta je rezultat poziva `Arrays.binarySearch(lotoBrojevi, 17)`?

A. 0 B. -1 C. 1 D. 2 E. -2

33. Koja je od ovih deklaracija ispravna?

A. `char[][] z = {'a', 'b'};`
B. `char[2][2] z = {{'a', 'b'}, {'c', 'd'}};`
C. `char[2][] z = {{'a', 'b'}, {'c', 'd'}};`
D. `char[][] z = {{'a', 'b'}, {'c', 'd'}};`

34. Ako je data deklaracija niza `double[][] d = new double[4][5]`, koje su vrednosti dužina `d.length` i `d[2].length`?

A. 4 i 4 B. 4 i 5 C. 5 i 4 D. 5 i 5

35. 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]);
   }
}
```

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

8.2 Rešenja zadataka

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.

Program 8.1: Eratostenovo sito

```
/**
 * 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. Napisati program koji simulira „igru života". Ova igra se sastoji od kolonije organizama koji žive u sopstvenim ćelijama u jednoj dvodimenzionalnoj matrici 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 odgovarajućoj strani uvek imaju prazne susedne ćelije.) Pravila za formiranje nove generacije organizama su:

   1. Živi organizam u ćeliji preživljava u sledećeoj generaciji ukoliko je broj njegovih suseda dva ili tri.
   2. Živi organizam u ćeliji umire u sledećeoj 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).

Program 8.2: Igra života

```
import java.util.*;

public class IgraZivota {

   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 ćelija:
                           //    true  = zauzeta ćelija
                           //    false = prazna ćelija

   // Konstruktor
   public Kolonija(int n) {
      this.n = n;
      m = new boolean[n+2][n+2]; // prazne ćelije oko ivice matrice
   }

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

   public void novaGen() {

      boolean[][] m1 = new boolean[n+2][n+2]; // novo stanje ćelija
      int bs;                                 // broj suseda ćelije

      for (int i = 1; i <= n; i++)
         for (int j = 1; j <= n; j++) {
            // Određivanje broja suseda ćelije 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 ćelije 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. Napisati program kojim se simulira igranje igre iks-oks sa računarom. Vrste i kolone table za igranje numerisane su brojevima 0, 1 i 2 radi lakšeg korišćenja polja table pomoću njihovih koordinata. Program treba da obaveštava korisnika o koordinatama svog izabranog polja kada je na potezu, a korisnik svoj potez zadaje koordinatama željenog polja. Posle svakog odigranog poteza programa (X) ili korisnika (O), na ekranu treba prikazati trenutni sadržaj table, na primer:

```
-------------
| X |   |   |
-------------
| O | X | O |
-------------
|   | O | X |
-------------
```

Program 8.3: Igranje iks-oks sa računarom

```
import java.util.*;

public class IksOks {

    static final char PRAZNO  = ' ';
    static final char RAČUNAR = 'X';
    static final char ČOVEK   = 'O';

    static final int ČOVEK_POBEDNIK   = 0;
    static final int NEREŠENO         = 1;
    static final int NEJASNO          = 2;
    static final int RAČUNAR_POBEDNIK = 3;
```

```
private char[][] tabla;  // tabla za igranje

// Konstruktor
public IksOks() {
   tabla = new char[3][3];
   obrišiTablu();
}

// Inicijalizacija table
public void obrišiTablu() {
   for (int i = 0; i < 3; i++)
      for (int j = 0; j < 3; j++)
         tabla[i][j] = PRAZNO;
}

// Prikazivanje table
public void prikažiTablu() {
   System.out.println();
   System.out.println("    0   1   2");
   System.out.println("  -------------");
   for (int i = 0; i < 3; i++) {
      System.out.print(i + " |");
      for (int j = 0; j < 3; j++)
         System.out.print(" " + tabla[i][j] + " |");
      System.out.println();
      System.out.println("  -------------");
   }
}

// Proveravanje ispravnosti datog poteza
public boolean ispravanPotez(int i, int j) {
   if (i < 0 || i >= 3 || j < 0 || j >= 3
             || tabla[i][j] != PRAZNO)
      return false;
   else
      return true;
}

// Proveravanje da li je dato polje prazno
public boolean praznoPolje(int i, int j) {
   return tabla[i][j] == PRAZNO;
}

// Proveravanje da li su sva polja popunjena
```

```
public boolean punaTabla() {
   for (int i = 0; i < 3; i++)
      for (int j = 0; j < 3; j++)
         if (praznoPolje(i, j))
            return false;
   return true;
}

// Određivanje da li je pozicija pobednička za igrača
public boolean pobeda(char igrač) {
   int i, j;

   // Pregledati vrste
   for (i = 0; i < 3; i++) {
      for (j = 0; j < 3; j++)
         if (tabla[i][j] != igrač)
            break;
      if (j >= 3)
         return true;
   }

   // Pregledati kolone
   for (j = 0; j < 3; j++) {
      for (i = 0; i < 3; i++)
         if (tabla[i][j] != igrač)
            break;
      if (i >= 3)
         return true;
   }

   // Pregledati obe dijagonale
   if (tabla[0][0] == igrač && tabla[1][1] == igrač
                            && tabla[2][2] == igrač)
      return true;
   if (tabla[0][2] == igrač && tabla[1][1] == igrač
                            && tabla[2][0] == igrač)
      return true;

   return false;
}

// Odigravanje jednog poteza (bez provere ispravnosti)
public void odigrajPotez(int i, int j, char igrač) {
   tabla[i][j] = igrač;
```

```
}

// Izračunavanje vrednosti trenutne pozicije
public int rezultatPozicije() {
   return pobeda(RAČUNAR) ? RAČUNAR_POBEDNIK :
          pobeda(ČOVEK)   ? ČOVEK_POBEDNIK :
          punaTabla()     ? NEREŠENO       : NEJASNO;
}

// Nalaženje najboljeg poteza igrača za trenutnu poziciju
public Potez izaberiNajboljiPotez(char igrač) {

   char protivnik;     // protivnički igrač
   Potez pp;           // najbolji potez protivnika
   int ni = 0, nj = 0; // koordinate najboljeg poteza
   int r;              // (privremeni) rezultat date pozicije

   r = rezultatPozicije();
   if (r != NEJASNO)   // završna pozicija
      return new Potez(r, -1, -1);

   if (igrač == RAČUNAR) {
      protivnik = ČOVEK;
      r = ČOVEK_POBEDNIK;
   }
   else {
      protivnik = RAČUNAR;
      r = RAČUNAR_POBEDNIK;
   }

   for (int i = 0; i < 3; i++)
      for (int j = 0; j < 3; j++)
         if (praznoPolje(i, j)) {
            odigrajPotez(i, j, igrač);  // pokušaj potez
            pp = izaberiNajboljiPotez(protivnik);
            odigrajPotez(i, j, PRAZNO); // vrati potez

            if ((igrač == RAČUNAR && pp.vrednost() > r) ||
                (igrač == ČOVEK && pp.vrednost() < r)) {
               // Zapamtiti najbolji dosadašnji rezultat
               r = pp.vrednost();
               ni = i;
               nj = j;
            }
```

```
            }
      return new Potez(r, ni, nj);
   }

   public static void main(String[] args) {

      int či, čj;     // koordinate čovekovog poteza
      int ri, rj;     // koordinate računarovog poteza
      char naPotezu;  // igrač koji je na potezu da igra
      int r;          // rezultat pozicije nakon odigranog poteza
      Potez potez;    // izabrani najbolji potez računara

      IksOks igra = new IksOks();
      Scanner tastatura = new Scanner(System.in);

      System.out.print("Hajde da igramo iks-oks: ja sam ");
      System.out.println(RAČUNAR + ", ti si " + ČOVEK + ".");
      System.out.print("Vrste i kolone table su numerisane ");
      System.out.println("brojevima 0,1,2,");
      System.out.print("a polja su predstavljena odgovarajućim ");
      System.out.println("koordinatama.");
      System.out.print("Na primer, polje u sredini ima ");
      System.out.println("koordinate 1 1.");

      igra.prikažiTablu();
      System.out.println();
      System.out.print("Ja igram prvi (d/n)? ");
      String odg = tastatura.next();
      if (odg.toUpperCase().equals("N"))
         naPotezu = ČOVEK;
      else
         naPotezu = RAČUNAR;

      while (true) {
         if (naPotezu == RAČUNAR) {
            potez = igra.izaberiNajboljiPotez(RAČUNAR);
            ri = potez.vrsta(); rj = potez.kolona();
            igra.odigrajPotez(ri, rj, RAČUNAR);
            System.out.println("Moj potez: " + ri + " " + rj);
            igra.prikažiTablu();
            System.out.println();
            r = igra.rezultatPozicije();
            if (r == RAČUNAR_POBEDNIK) {
               System.out.println("Ja sam pobedio!");
```

```
                break;
            }
            else if (r == NEREŠENO) {
                System.out.println("Nerešeno!");
                break;
            }
            naPotezu = ČOVEK;
        }

        if (naPotezu == ČOVEK) {
            do {
                System.out.print("Tvoj potez: ");
                či = tastatura.nextInt(); čj = tastatura.nextInt();
            } while (!igra.ispravanPotez(či, čj));
            igra.odigrajPotez(či, čj, ČOVEK);
            igra.prikažiTablu();
            System.out.println();
            r = igra.rezultatPozicije();
            if (r == ČOVEK_POBEDNIK) {
                System.out.println("Bravo, tvoja pobeda!");
                break;
            }
            else if (r == NEREŠENO) {
                System.out.println("Nerešeno!");
                break;
            }
            naPotezu = RAČUNAR;
        }
      }
   }
}

class Potez {

   private int i;  // vrsta poteza
   private int j;  // kolona poteza
   private int v;  // vrednost pozicije s tim potezom

   public Potez(int v, int i, int j) {
      this.i = i;
      this.j = j;
      this.v = v;
   }
```

```
    public int vrsta() {
        return i;
    }

    public int kolona() {
        return j;
    }

    public int vrednost() {
        return v;
    }
}
```

4. 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.

Program 8.4: Keš-memorija

```
import java.util.*;

public 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];

         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) {

         KešMemorija kešReči = new KešMemorija(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;
    }
}
```

5. 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.

Program 8.5: Često ponavljane reči u tekstu

```
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;
    }
}
```

6. 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 nema pacijenata koji čekaju za njegovu specijalnost, doktor je slobodan i odmara se sve 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 pre će biti primljeni) i doktori se odmaraju na fer način (oni koji su se duže odmarali pre će 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 pregledanog 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.

Program 8.6: Dom zdravlja

```
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();

            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.otvoriRecepciju(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();
    }
}
```

9

Nasleđivanje klasa

9.1 Odgovori na pitanja

1. Objektno orijentisano programiranje i Java omogućavaju da se nove klase pišu na osnovu postojećih klasa. Kako se naziva takva mogućnost?

   A. Enkapsulacija B. Nasleđivanje C. Polimorfizam D. Apstrakcija

2. Koje su od ovih rečenica tačne?

   A. Jedna klasa u Javi može *direktno* proširivati više klasa.

   B. Proširena klasa sadrži dodatna polja i metode u odnosu na svoju nasleđenu klasu.

   C. „Klasa `A` nasleđuje klasu `B`" znači da je `A` potklasa od `B`.

   D. Ako klasa `A` nasleđuje klasu `B`, tada objekti klase `A` sadrže sva polja i sve metode klase `B`.

   E. Ako klasa `A` nasleđuje klasu `B`, tada se svaki objekat klase `A` podrazumeva da je i objekat klase `B`.

3. Koja se od ovih modifikacija najbolje odnosi na klasu naslednicu u odnosu na postojeću klasu koju nasleđuje?

   A. Zamenjivanje B. Proširivanje C. Popravljanje D. Uklanjanje

4. 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.vozi();
6           v.brojVrata = 2;
7           Vozilo bmw = new Auto(2, 4);
8           bmw.vozi();
9           Auto audi = new Auto(4);
10          Auto.vozi();
11       }
12    }
13
14    class Vozilo {
15
16        public int brojTočkova;
17        public Vozilo(int t) {
18            brojTočkova = t;
19        }
20        public void vozi() {
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 vozi() {
33            System.out.println("Vožnja auta");
34        }
35    }
```

A. Greške su u redovima 7, 10 i 14.

B. Greške su u redovima 7, 9 i 25.

C. Greške su u redovima 6, 7 i 10.

D. Greške su u redovima 6, 9 i 10.

E. Greške su u redovima 6, 9 i 29.

F. Greške su u redovima 9, 10 i 29.

5. 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);
   }
}
```

A. Program ima grešku, jer klasa `Test` nema podrazumevani konstruktor `Test()`.

B. 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`.

C. Program ima grešku, ali bi radio bez greške ukoliko bi se klasi `A` eksplicitno dodao konstruktor bez parametara `A()`.

6. 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");
   }
}
```

A. Ništa.

B. `Pozvan podrazumevani konstruktor klase A`

C. `Pozvan podrazumevani konstruktor klase B`

D. `Pozvan podrazumevani konstruktor klase A` i u drugom redu `Pozvan podrazumevani konstruktor klase B`

E. `Pozvan podrazumevani konstruktor klase B` i u drugom redu `Pozvan podrazumevani konstruktor klase A`

7. Koja od ovih rečenica o službenoj reči `super` *nije* tačna?

   A. Službena reč `super` može poslužiti za pozivanje konstruktora nasleđene klase.

   B. Službena reč `super` može poslužiti za pozivanje zaklonjenog metoda nasleđene klase.

   C. Službena reč `super` ne može poslužiti za pozivanje zaklonjenog polja nasleđene klase.

8. 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);
    }
}
```

A. Program ima grešku, jer je metod `m()` nadjačan sa različitim potpisom u klasi `B`.

B. Program ima grešku, jer se `b.m(5)` ne može pozvati pošto je metod `m(int)` zaklonjen u klasi `B`.

C. Program ima grešku zbog `b.i`, jer je polje `i` nepristupačno iz klase `B`.

D. Program nema greške, jer metod `m()` nije nadjačan u klasi `B`, nego klasa `B` nasleđuje metod `m(int)` od klase `A` i u klasi `B` je definisan preopterećen metod `m(String)`.

9. Koje su od ovih rečenica o metodima tačne?

A. Metod se može preopteretiti u istoj klasi.

B. Metod se može nadjačati u istoj klasi.

C. Ako su metodi preopterećeni, njihovi potpisi moraju biti isti.

D. Ako jedan metod nadjačava drugi metod, njihovi potpisi moraju biti isti.

10. Koje su od ovih rečenica o polimorfizmu tačne?

A. 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.

B. Objekat tipa `B` se može preneti kao argument metodu na mesto parametera tipa `A` ukoliko je `B` klasa naslednica od `A`.

C. Za razrešavanje poziva preopterećenih metoda se primenjuje statičko vezivanje u fazi prevođenja programa.

D. Za razrešavanje poziva nadjačanih metoda se primenjuje dinamičko vezivanje u fazi izvršavanja programa.

11. 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 (produženi)");
```

```
    }
}
```

A. Izvršavanjem programa se na ekranu ništa ne prikazuje.

B. Izvršavanjem programa se na ekranu prikazuje tekst: `Pobednik je Crvena zvezda`.

C. Izvršavanjem programa se na ekranu prikazuje tekst: `Pobednik je Crvena zvezda (Beograd)`.

D. Izvršavanjem programa se na ekranu prikazuje tekst: `Pobednik je Partizan`.

E. Izvršavanjem programa se na ekranu prikazuje tekst: `Pobednik je Partizan (Beograd)`.

F. Izvršavanjem program se na ekranu prikazuje tekst: `Nerešeno`.

12. 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;
   }
}
```

A. Program ima grešku, jer naredbu `System.out.println(a)` treba zameniti naredbom `System.out.println(a.toString())`.

B. Program nema greške i poziva se metod `toString()` u klasi `Object` prilikom izvršavanja naredbe `System.out.println(a)`.

C. Program nema greške i poziva se metod `toString()` u klasi `A` prilikom izvršavanja naredbe `System.out.println(a)`.

D. Program nema greške i poziva se metod `toString()` u klasi `Object` prilikom izvršavanja naredbe `System.out.println(o)`.

13. 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?

A. `public boolean equals(String s)`

B. `public boolean equals(Object o)`

C. `public static boolean equals(Object o)`

D. `public boolean equals(String s1, String s2)`

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(Object o) {
      A a = (A)o;
      return this.x == a.x;
   }
}
```

A. Program ima grešku, jer se izrazom `a1.equals(a2)` proverava jednakost objekata `a1` i `a2` različitog tipa od `Object`.

B. Program ima grešku, jer se jednakost objekata `a1` i `a2` tipa `A` proverava izrazom `a1 == a2`.

C. Program se izvršava bez greške i prikazuje se `true` na ekranu.

D. Program se izvršava bez greške i prikazuje se `false` na ekranu.

15. 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;
   }
}
```

A. Program ima grešku, jer se izrazom `a1.equals(a2)` proverava jednakost objekata `a1` i `a2` različitog tipa od `Object`.

B. Program ima grešku, jer se jednakost objekata `a1` i `a2` tipa `A` proverava izrazom `a1 == a2`.

C. Program se izvršava bez greške i prikazuje se `true` na ekranu.

D. Program se izvršava bez greške i prikazuje se `false` na ekranu.

16. 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;
   }
}
```

A. Program ima grešku, jer se izrazom `a1.equals(a2)` proverava jednakost objekata `a1` i `a2` različitog tipa od `Object`.

B. Program ima grešku, jer se jednakost objekata `a1` i `a2` tipa `A` proverava izrazom `a1 == a2`.

C. Program se izvršava bez greške i prikazuje se `true` na ekranu.

D. Program se izvršava bez greške i prikazuje se `false` na ekranu.

17. 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";
   }
}
```

A. Greška je u pozivu `m(new MasterStudent())`.

B. Greška je u pozivu `m(new Student())`.

C. Greška je u pozivu `m(new Osoba())`.

D. Greška je u pozivu `m(new Object())`.

18. 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();
```

A. Rezultat izraza `c1 instanceof C1` je `true`.

B. Rezultat izraza `c2 instanceof C1` je `true`.

C. Rezultat izraza `c3 instanceof C1` je `true`.

D. Rezultat izraza `c4 instanceof C2` je `true`.

19. Analizirajte sledeći program:

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

A. Kada se vrednost promenljive `s` dodeljuje promenljivoj `o` u naredbi `Object o = s`, konstruiše se novi objekat.

B. Kada se konvertuje tip promenljiva `o` i njena vrednost dodeljuje promenljivoj `t` u naredbi `String t = (String)o`, konstruiše se novi objekat.

C. Kada se konvertuje tip promenljiva `o` i njena vrednost dodeljuje promenljivoj `t` u naredbi `String t = (String)o`, sadržaj promenljive `o` se menja.

D. Promenljive `s`, `o` i `t` ukazuju na isti objekat tipa `String`.

20. Ako je navedena deklaracija niza `Object[] a`, koje su od ovih naredbi ispravne?

A. `a = new char[100];`

B. `a = new int[100];`

C. `a = new double[100];`

D. `a = new String[100];`

E. `a = new java.util.Date[100];`

21. Koja se od ovih klasa *ne* može nasleđivati?

A. `class A { }`

B. `class A { private A(){ } }`

C. `final class A { }`

D. `class A { protected A(){ } }`

9.2 Rešenja zadataka

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. (Pretpostavite da objekti u elementima liste implementiraju metod `equals()` kojim se ispituje njihova jednakost.)

Program 9.1: Povezana lista

```
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.dužina());

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

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();
    }
}
```

2. Koristeći klasu `PovezanaLista` iz 1. zadatka napisati klasu `PoliLinija` kojom se poligonalna linija predstavlja u obliku liste njenih tačaka temena.

Program 9.2: Poligonalna linija

```
public class PoliLinija {

   private PovezanaLista listaTemena; // povezana lista temena

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

      listaTemena = new PovezanaLista();

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

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

      listaTemena = new PovezanaLista();

      for (Tačka t : temena)
         listaTemena.dodaj(t);
   }

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

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

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

   public String toString() {
      return listaTemena.toString();
```

```
    }

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

       // Formiranje niza koordinata tačaka temena
       double[][] koordTemena = {{2,2}, {1,2}, {-2,3},
                                 {3,-4}, {-1,-1}, {0,0}};
       PoliLinija pl = new PoliLinija(koordTemena);
       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.nađiTeme(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 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;
    }
}

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. 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.

Program 9.3: Povezana lista, proširena verzija

```
public class PovezanaLista {

    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 PovezanaLista() {}

    // 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;

    // Dodati novi element ispred tekućeg elementa
    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 ukloniTekući() {

    if (praznaLista()) return;

    // Odrediti prethodni element levo od tekućeg
    ElementListe pret = null;
    for (ElementListe elem = prvi; elem != tekući;
            elem = elem.getSled())
        pret = elem;

    // Ukloniti tekući element
    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 nađi(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
      PovezanaLista lis = new PovezanaLista();

      // 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.dužina());

// Dodavanje nekih elemenata u tekuću poziciju
lis.dodajTekući(51);
lis.tekućiPrvi();
lis.dodajTekući(3);
lis.tekućiSledeći(); lis.tekućiSledeći();
lis.dodajTekući(11);
lis.dodaj(0);

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

// Uklanjanje tekućih elemenata
lis.ukloniTekući();
lis.tekućiPrvi();
lis.ukloniTekući();
lis.tekućiSledeći(); lis.tekućiSledeći();
lis.ukloniTekući();

// 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();
    }
}
```