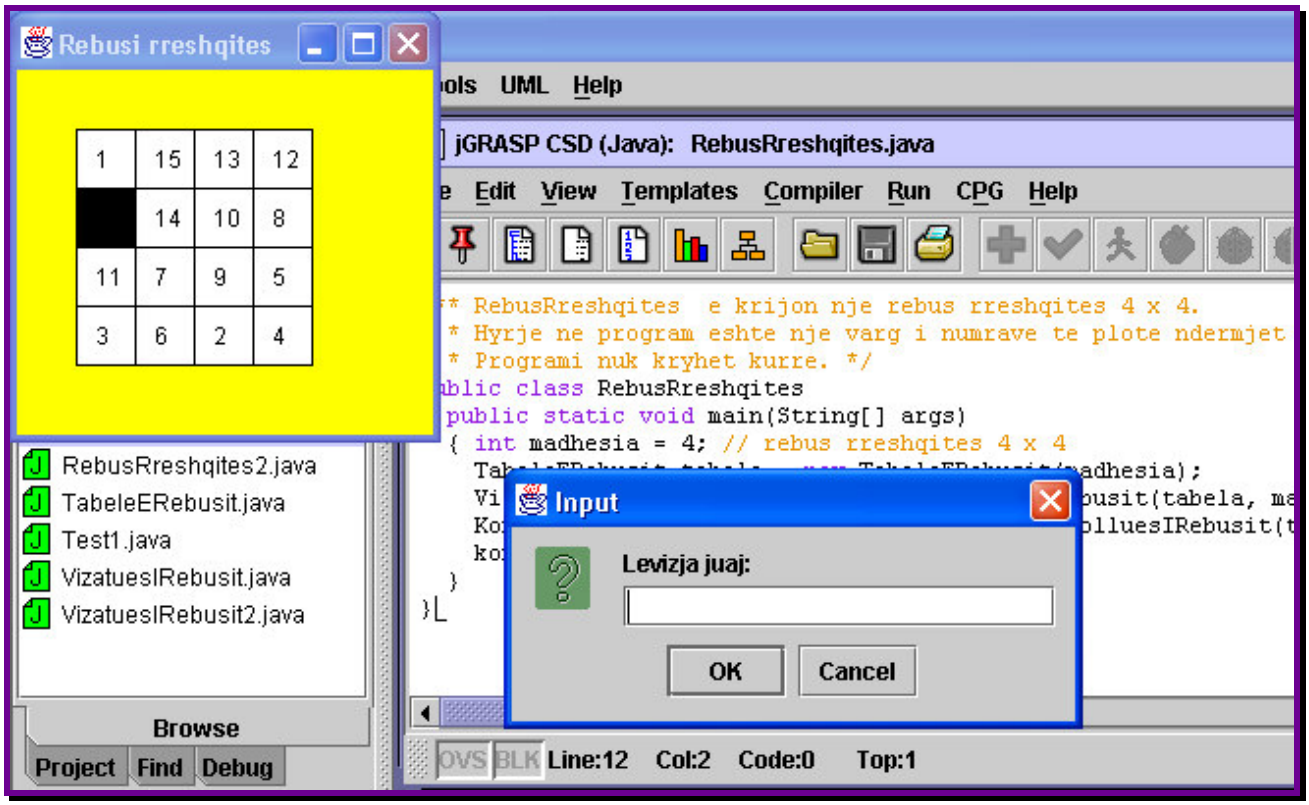
Principe të Programimit në Java: Arkitekturat dhe Interfejsat

Ushtrime
(Pjesa e dytë: kapitujt VII-VIII)

nga David Schmidt me Ridvan Bunjakun



Gjilan
2005

Përmbajtja

Hyrje

Në këtë tekst janë përkthyer dhe përshtatur ushtrimet nga kapitujt VII-VIII të librit të David Schmidt: "Programming Principles in Java: Architectures and Interfaces", që mund ta gjeni në www.cis.ksu.edu/santos/schmidt/ppj. Ky është vazhdimi i librit të parë ku janë ushtrimet nga kapitujt I-VI. E falënderoj autorin për lejimin e publikimit të këtij teksti.
Libri i Schmidt-it është i mrekullueshëm për fillestarët në programim dhe konkretisht në Java. Me anë të shembujve praktikë e shumë të afërt me cilindo lexues, autori e bën të këndshëm mësimin e Java-s dhe të teknikave të ndryshme të programimit.
Motivi për shkruarjen e këtij teksti në gjuhën shqipe lindi gjatë punës me studentë të cilët nuk e zotëronin mirë gjuhën angleze. Për shkak të pengesës së gjuhës ata nuk mund ta lexonin librin dhe t'i kuptonin kodet e ndryshme. Përshtatja ime e kodit në gjuhën shqipe (në masën sa është e mundur) doli të jetë e dobishme për ta.
Natyrisht, parapëlqehet që ky tekst të përdoret krahas librit të Schmidt-it. Poashtu, informata të dobishme mund të mirren edhe nga ueb-faqja e prof. Faton Berisha: fberisha.netfirms.com.

Për kë është ky tekst?

Ky tekst është shkruar për studentët që i ndjekin ushtrimet nga "Strukturat e të dhënave" në Universitetin e Prishtinës, FSHMN, Departamenti i Matematikës, Drejtimi i Shkencës Kompjuterike. Mirëpo mund ta përdorin si tekst për ushtrime të gjithë ata që duan ta mësojnë gjuhën Java bashkë me disa teknika bashkëkohore të programimit dhe filozofinë e programimit të orientuar kah objektet.

Gjuha e përdorur në tekst

Ky tekst nuk e ka për qëllim shmangien e mësimit të gjuhës angleze, por mësimin thelbësor të gjërave. Gjuha angleze, në të vërtetë, është kusht i domosdoshëm për Shkencën Kompjuterike.
I mirëpres të gjitha vërejtjet, këshillat dhe komentet rreth tekstit.

Ridvan Bunjaku, ing. mat.
ridvan.bunjaku@fshmn.uni-pr.edu

VII Shabllonat e Përsëritjeve: Iterimi dhe Rekursioni

(7.1 Përsëritja)

7.2 Urdhëri `while`

1. Çka do të afishojnë këto cikle-while ?

a.

```
String s = "";
while ( s.length() != 5 )
      { s = s + 'a';
        System.out.println(s);
      }
```

(Rikujto se veprimi `length` e kthen gjatësinë e stringut; për shembull,

```
 String s = "abcde";
 int i = s.length();
```

ia ndan vlerën 5 ndryshores `i`.)

Përgjigje:

```
a
aa
aaa
aaaa
aaaaa
```

b.

```
int teposhte = 10; // countdown
while ( teposhte != 0 )
      { System.out.println(i);
        teposhte = teposhte – 1;
      }
```

Përgjigje:

Ky kod i afishon numrat e plotë prej 10 deri në 1.

c.

```
int teposhte = 5;
int perpjete = -1; // countup
while ( teposhte > perpjete )
      { teposhte = teposhte - 1;
        System.out.println(teposhte + " " + perpjete);
        perpjete = perpjete + 1;
      }
```

Përgjigje:

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

d.

```
while ( false )
      { System.out.println("false"); }
```

Përgjigje:

Ky kod nuk afishon asgjë.

e.

```
String s = "";
while ( s.length() < 6 )
      { s = s + "a";
        if ( (s.length() % 2) == 1 )
           { System.out.println(s); }
      }
```

Përgjigje:

```
a
aaa
aaaaa
```

2. Shkruaje një cikël-while për t'i afishuar karakterët, prej `'a'` deri në `'z'`, nga një në rresht. (Udhëzim: rikujto se Java të lejon ta inicializosh një ndryshore si `char k = 'a'` dhe të bësh aritmetikë me të, p.sh. `k = k + 1` .)

Përgjigje:

```
char k = 'a'
while ( k <= 'z' )
      { System.out.println(k);
        k = k + 1;
      }
```

3. Shkruaje një cikël-while për t'i afishuar të gjithë plotpjesëtuesit e 1000shit që janë ndërmjet 2 e 50. (Rikujto se një numër i plotë, `i`, e plotpjesëton një tjetër, `j` , nëse `j % i` është 0. Udhëzim: fute një urdhër-if në trupin e ciklit.)

Përgjigje:

```
int n = 2;
while ( n <= 50 )
      { if ( 1000 % n == 0 )
           { System.out.println(n); }
        n = n + 1;
      }
```

7.3 Iterimi Definitiv

1. Implemento një aplikacion që i llogarit mesataret e provimit:

a. Së pari, vendose metodën `llogariteMesataren` në një klasë të re që do ta shkruash, të quajtur `StatistikatEProvimit`. Pastaj shkruaje një metodë `main` me këtë algoritëm:

1. konstrukto një objekt `StatistikatEProvimit`;
2. konstrukto një dialog që kërkon nga shfrytëzuesi ta shkruajë një numër pozitiv për numrin e provimeve
3. thirre metodën `llogariteMesataren` në objektin `StatistikatEProvimit`.
4. konstrukto një dialog që e tregon rezultatin e kthyer nga `llogariteMesataren`.

Përgjigje:

```
import javax.swing.*;
/** StatistikatEProvimit permban metoda
  * per llogaritjen e statistikave te provimit */
public class StatistikatEProvimit
{ public double llogariteMesataren(int sa)
  { ... }

  public static void main(String[] args)
  { StatistikatEProvimit llog = new StatistikatEProvimit();
    String s = JOptionPane.showInputDialog("Shkruaje sasine e provimeve:");
    int sasia = new Integer(s).intValue();
    double mesatarja = llog.llogariteMesataren(sasia);
    JOptionPane.showMessageDialog(null, "Rezultati mesatar: " + mesatarja);
  }
}
```

b. Ndryshoje metodën `llogariteMesataren` ashtu që nëse argumenti i saj është nr. i plotë jopozitiv, metoda tregon një dialog që e shpall një gabim dhe kthen `0.0` si përgjigje.

Përgjigje:

```
double llogariteMesataren(sasia)
{ double pergjigjja = 0.0;
  if ( sasia < 1 )
       { JOptionPane.showMessageDialog(null,
            "Gabim te StatistikatEProvimit: sasia e provimeve - jopozitive)");
       }
  else { double total_pike = 0.0; // totali i krejt pikeve te testeve
         int numri = 0; // numri i testeve te lexuara deri tash
         while ( numri != sasia )
               { String s = JOptionPane.showInputDialog(
                   "Rezultati i provimit + nr. " + (numri + 1) );
                 r = new Integer(s).intValue();
                 total_pike = total_pike + r;
               }
         pergjigjja = total_pike / sa;
       }
  return pergjigjja;
}
```

c. Pastaj, shkruaje një metodë: `llogariteRezultMax`:

```
  /** llogariteRezultMax e lexon nje varg te rezultateve te testeve
    * te dhene nga shfrytezuesi dhe e kthen rezultatin me te larte
    * te lexuar
    * @param sasia - sasia e rezultateve te testeve qe lexohen;
    *                duhet te jete jonegative
    * @return (kthen) rezultatin maksimal */
  public double llogariteRezultMax(int sasia)
```

Shtoja këtë metodë klasës `StatistikatEProvimit` dhe modifikoje metodën `main` (që e shkrove më herët) për ta thirrur `llogariteRezultMax` në vend të `llogariteMesataren`.

Përgjigje:

```
import javax.swing.*;
public class StatistikatEProvimit
{ public int llogariteRezultMax(int sasia)
  { int rezult_max = 0;
    if ( sasia < 1 )
         { JOptionPane.showMessageDialog(null,
             "Gabim te StatistikatEProvimit: sasia e provimeve jopozitive");
         }
    else { int numeruesi = 0;  // sasia e testeve te lexuara deri tash
           while ( numeruesi != sasia )
                  // ne cdo iteracion: rezult_max  e mban maksimumin e
                  //   test_1, test_2, ..., test_numri te lexuara deri tash:
                 { String hyrja = JOptionPane.showInputDialog
                                          ("Shkruaje rezult. e testit tjeter:");
                   int piket = new Integer(hyrja).intValue();
                   if ( piket > rezult_max )
                      { rezult_max = piket; }
                   numeruesi = numeruesi + 1;
                 }
         }
    return rezult_max;
  }

  public static void main(String[] args)
  { StatistikatEProvimit llog = new StatistikatEProvimit();
    String s = JOptionPane.showInputDialog("Numri i provimeve:");
    int sasia_e_pr = new Integer(s).intValue();
    int max = llog.llogariteRezultMax(sasia_e_pr);
    JOptionPane.showMessageDialog(null, "Rezultati me i larte: " + max);
  }
}
```

d. Shkruaje këtë metodë dhe përfshije brenda klasës `StatistikatEProvimeve`:

```
/** llogariteMesatarenPakMeTeMire e llogarit mesataren e rezultateve
  * te testeve te dhena nga shfrytezuesi _por e perjashton_ rezultatin
  * me te ulet ne llogaritje.
  * @param sa - sasia e rezultateve te provimeve qe lexohen;
  *             duhet te jete > 1
  * @return (kthen) mesataren e (sa - 1) rezultateve me te mira */
```

Përgjigje:

```
public double llogariteMesatarenPakMeTeMire(int sa)
{ double pergjigjja = 0.0;
  if ( sa < 1 )
       { JOptionPane.showMessageDialog(null,
           "Gabim te StatistikatEProvimit: sasia e testeve jopozitive");
       }
  else { double total_pike = 0.0;  // totali i krejt pikeve te testeve
         int piket_min = 0;  // piket me te uleta te lexuara deri tash
         int numeruesi = 0;  // sasia e testeve te lexuara deri tash
         while ( numeruesi != sa )
               { String hyrja = JOptionPane.showInputDialog
                                   ("Shkruaje rezult. e testit tjeter:");
                 int piket = new Integer(hyrja).intValue();
                 total_pike = total_pike + piket;
                 // provo mos  piket  jane piket e reja minimale:
                 if ( piket < piket_min  ||  numeruesi == 0 )
                    { piket_min = piket; }
                 count = count + 1;
               }
          pergjigjja = (total_pike - piket_min) / (sa - 1);
        }
  return pergjigjja;
}
```

2. Shkruaje një shteg (gjurmë) të ekzekutimit për këtë shembull:

```
int t = 4;
int numeruesi = 2;
while ( numeruesi <= 4 )
      { t = t * 2;
        numeruesi = numeruesi + 1;
      }
```

Përgjigje:

```
>int t = 4; int numeruesi = 2; while ( numeruesi <= 4 ) { t = t * 2; numeruesi =
numeruesi+1; }

=> int t ==| 4 |
   >int numeruesi = 2; while ( numeruesi <= 4 ) {t = t * 2; numeruesi =
numeruesi+1;}

=> int t ==| 4 |  int numeruesi ==| 2 |
   >while ( numeruesi <= 4 ) { t = t * 2; numeruesi = numeruesi+1; }

=> int t ==| 4 |  int numeruesi ==| 2 |
   >while ( 2 <= 4 ) { t = t * 2; numeruesi = numeruesi+1; }

=> int t ==| 4 |  int numeruesi ==| 2 |
   >while ( true ) { t = t * 2; numeruesi = numeruesi+1; }

=> int t ==| 4 |  int numeruesi ==| 2 |
   while ( true ) { >t = t * 2; numeruesi = numeruesi+1; }
```

```
=> int t ==| 4 |  int numeruesi ==| 2 |
   while ( true ) { >t = 8; numeruesi = numeruesi+1; }

=> int t ==| 8 |  int numeruesi ==| 2 |
   while ( true ) { ... >numeruesi = numeruesi+1; }

=> int t ==| 8 |  int numeruesi ==| 3 |
   while ( true ) { ... > }
```

Në këtë pikë, cikli përsëritet:

```
=> int t ==| 8 |  int numeruesi ==| 3 |
   >while ( numeruesi <= 4 ) { t = t * 2; numeruesi = numeruesi+1; }

=> int t ==| 8 |  int numeruesi ==| 3 |
   >while ( true ) { t = t * 2; numeruesi = numeruesi+1; }

=> int t ==| 8 |  int numeruesi ==| 3 |
   while ( true ) { >t = t * 2; numeruesi = numeruesi+1; }
```

Dy hapa më vonë kemi si vijon:

```
=> int t ==| 16 |  int numeruesi ==| 4 |
   while ( true ) { ... > }

=> int t ==| 16 |  int numeruesi ==| 4 |
   >while ( numeruesi <= 4 ) { t = t * 2; numeruesi = numeruesi+1; }

=> int t ==| 16 |  int numeruesi ==| 4 |
   >while ( true ) { t = t * 2; numeruesi = numeruesi+1; }

=> int t ==| 16 |  int numeruesi ==| 4 |
   while ( true ) { >t = t * 2; numeruesi = numeruesi+1; }

=> int t ==| 32 |  int numeruesi ==| 5 |
   while ( true ) { ... > }
=> int t ==| 32 |  int numeruesi ==| 5 |
   >while ( numeruesi <= 4 ) { t = t * 2; numeruesi = numeruesi+1; }

=> int t ==| 32 |  int numeruesi ==| 5 |
   >while ( false ) { t = t * 2; numeruesi = numeruesi+1; }

=> int t ==| 32 |  int numeruesi  ==| 5 |
    ... >
```

3. Përdore strukturën për iteracion definitiv për ta shkruar një cikël që e afishon këtë dalje, krejt në një rresht: `88 77 66 55 44 33 22 11`

Përgjigje:

```
int nr = 8;
while (nr >= 1)
      { System.out.print("" + nr + nr + " ");
        nr = nr - 1;
      }
System.out.println();
```

ose:

```
int nr = 8;
while ( nr >= 1 )
      // deri tash jane afishuar:  88, 77 ...deri ne... (nr+1)*11
      { System.out.print(nr*11 + " ");
        nr = nr - 1;
      }
System.out.println();
```

4. Shkruaje një metodë `main` që bën si vijon:

a) e përdor një cikël për të kërkuar nga shfrytëzuesi që t'i shkruajë katër fjalë, një nga një, p.sh.

```
Une
e
dua
kompjuterin
```

b) e tregon një dialog që i afishon fjalët të listuara në renditje të kundërt në një rresht, p.sh.

```
kompjuterin dua e Une
```

Përgjigje:

```
import javax.swing.*;
public class Test4
{ public static void main(String[] args)
  { int pika_ndalese = 4;
    int numeruesi = 0;
    String fjalia = "";
    while ( numeruesi != pika_ndalese)
          { String hyrja = JOptionPane.showInputDialog
                           ("Shkruaje nje fjale:");
            fjalia = hyrja + " " + fjalia;
            numeruesi = numeruesi + 1;
          }
    JOptionPane.showMessageDialog(null, fjalia);
  }
}
```

5. Shumë përkufizime standarde matematikore janë llogaritur me cikle të iteracionit definitiv. Për secilin nga përkufizimet që vijojnë, shkruaje një metodë që e llogarit përkufizimin:

a. shuma e një numri të plotë jonegativ, *i*, përkufizohet kështu:

shuma(*i*) = 0 + 1 + 2 + ... + *i*

Për shembull, shuma(4) është 0 + 1 + 2 + 3 + 4 = 10. Pra, shkruaje një metodë që e llogarit shumën, ballina e të cilës duket kështu:

```
public int shuma(int i)
```

Përgjigje:

```
public int shuma(int n)
{ int totali = 0;
  int numeruesi = 0;  // numeron deri ne n
  while ( numeruesi != n )
        // ne secilin iteracion:  totali == 0+1+...deri te...+numeruesi
        { numeruesi = numeruesi + 1;
          totali = totali + numeruesi;
        }
        // totali == 0+1+...deri ne...+n
  return totali;
}
```

b. Prodhimi i iteruar i dy numrave të plotë jonegativë, *a* dhe *b*, është:

prodhimi(*a*, *b*) = *a* * (*a*+1) * (*a*+2) * (*a*+3) * ... * *b*

(Vërejtje: nëse *b* > *a* është e vërtetë, atëherë përkufizo prodhimi(*a*, *b*) = 1.)
Për shembull prodhimi(3, 6) është 3 * 4 * 5 * 6 = 360.

Përgjigje:

```
/** prodhimi  e llogarit prodhimin e iteruar  a*(a+1)*(a+2)*...*b
  * @param  a - numri i plote fillues i prodhimit
  * @param  b - numri i plote mbarues i prodhimit
  * @return  (kthen) a*(a+1)*(a+2)*...*b, nese  a <= b;
  *  return (kthen) 1,  nese a > b   */
public int prodhimi(int a, int b)
{ int totali = 1;
  if ( a <= b )
     { totali = a;
       int numeruesi = a;
       while ( numeruesi != b )
             // totali == a * (a+1) * ...deri ne... * i
             { numeruesi = numeruesi + 1;
               totali = totali * numeruesi;
             }
       // totali == a * (a+1) * (a+2) * ... * b == product(a, b)
     }
  return total;
}
```

c. Një variant i famshëm i prodhimit të iteruar është faktorieli; për një numër jonegativ, *m*, fatorieli, *m*!, përkufizohet si vijon:

0! = 1
n! = 1 * 2 * 3 * ... * *n*, për *n* pozitiv

Për shembull, 5! është 1 * 2 * 3 * 4 * 5 = 120.
(Vërejtje e rëndësishme: Për shkak se vlerat e m! rriten të mëdha shpejt me rritjen e m, përdore Java-tipin e të dhënave `long` ("long integer" - nr. i plotë i gjatë) në vend të `int` për argument dhe përgjigje të këtij funksioni.)

Përgjigje:

```
/** faktoriel  e llogarit n! per n ne rangun 0..20
  * (verejtje: vlerat per n>20 jane shume te medha per llogaritje)
  * @param  n - duhet te jete ne rangun 0..20
  * @return (kthen) n!, nese  n  ne 0.20; perndryshe kthen -1 */
public long faktoriel(int n)
{ long fakt = -1; // fakt - faktorieli
  if ( n >= 0  &&  n <= 20 )
     { int numeruesi = 0;  f = 1;
       while ( numeruesi != n )
             // fakt == 1*2*...deri te...*numeruesi
             { numeruesi = numeruesi + 1;
               fakt = fakt * numeruesi;
             }
       // fakt == faktoriel(n)
     }
  return fakt;
}
```

d. Nëse të pëlqen t'i përdorësh sinusin dhe kosinusin, atëherë përdore metodën faktoriel në ushtrimin e kaluar për t'i implementuar këto metoda klasike:

```
/** sinus e llogarit sinusin e argumentit te vet,
  * duke e perdorur formulen
  * sin(x) = x - (x^3/3!) + (x^5/5!) - (x^7/7!) + ... - (x^19/19!)
  * @param x - vlera, ne radiane, sinusi i te ciles deshirohet
  * (p.sh. sinus(0)=0, sinus(pi/2)=1, sinus(pi)=0, sinus(3pi/2)=-1 etj.)
  * @return (kthen) sinusin e llogaritur nga formula */
public double sinus(double x)
```

(Vërejtje. përdore `Math.pow(a,b)` për të llogaritur a^b.) Krahaso përgjigjet që i jep metoda jote me ato që i jep metoda `Math.sin(...)`.

```
/** kosinus e llogarit kosinusin e vleres se parametrit te vet,
  * duke e perdorur formulen
  * cos(x) = 1 - (x^2/2!) + (x^4/4!) - (x^6/6!) + ... (x^20/20!)
  * @param x - vlera, ne radiane, kosinusi i te ciles deshirohet
  * @return (kthen) kosinusin e llogaritur nga formula
public double kosinus(double x)
```

Përgjigje:

```
public double sinus(double x)
{ double sin = x;
  int numeruesi = 3;
  while ( numumeruesi <= 19 )
        // sin == x - (x^3/3!) +...deri te...(x^(numeruesi-2)/(numeruesi-2)!)
        { double d = Math.pow(x, numeruesi) / factoriel(numeruesi);
          if ( (numeruesi % 4) == 3 )
               { sin = sin - d; }
          else { sin = sin + d; }
          numeruesi = numeruesi + 2;
        }
  return sin;
}
```

(7.3.1 Shembull i Iterimit Definitiv: Vizatimi i Syrit të Demit)

7.4 Divergjenca

1. Shkruaje këtë metodë, që është e dizajnuar që të ekzekutohet përgjithmonë:

```
/** afishoReciproket afishon vlerat decimale te thyesave,
  * 1/2, 1/3, 1/4, ... , nje nga nje: Kur thirret,
  * e tregon nje dialog te mesazhit me informaten, "1/2 = 0.5",
  * dhe kur shfrytezuesi e shtyp OK, e tregon nje tjeter dialog te mesazhit
  * qe thote, "1/3 = 0.3333333333", dhe kur shfrytezuesi e shtyp OK,
  * e tregon edhe nje dialog te mesazhit, "1/4 = 0.25", etj. ...*/
public void afishoReciproket()
```

Përgjigje:

```
public class Test
public static void main(String[] args)
{ int emeruesi = 2;
  while ( true )  // nje menyre e thjeshte qe cikli te punoje pergjithmone
        { double thyesa = 1.0 / emeruesi;
          JOptionPane.showMessageDialog(null,
                          "1/" + emeruesi + " = " + thyesa);
          emeruesi = emeruesi + 1;
        }
}
```

Tash shkruaje një aplikacion që e thirr këtë metodë. Testoje aplikacionin derisa të mërzitesh.

7.5 Iterimi Indefinitiv: Përpunimi i hyrjes

1. Shpjego se çka bën kjo metodë:

```
public static void main(String[] args)
{ boolean duke_perpunuar;
  int totali = 0;
  while ( duke_perpunuar )
        { String s = JOptionPane.showInputDialog("Jepe nje numer te plote:");
          int i = new Integer(s);
          if ( i < 0 )
             { duke_perpunuar = false; }
          else { totali = totali + i; }
        }
  JOptionPane.showMessageDialog(null, totali);
}
```

Përgjigje:

I mbledh disa numra të plotë dhe ndalet kur në hyrje i jipet një numër negativ.

2. Shkruaje një aplikacion që i lexon aq rreshta të tekstit sa don përdoruesi. Përdoruesi i shkruan rreshtat, një nga një, në dialogje hyrëse. Kur përdoruesi e shtyp Cancel ose kur përdoruesi e shtyp vetem tastin Enter (pa tekst të shkruar), programi i afishon në dritaren komanduese të gjithë rreshtat që i ka shkruar përdoruesi dhe ndalet. (Udhëzim: Mund t'i ruash rreshtat e plotë të tekstit në një ndryshore `String` si vijon:

```
String s = "";
...
s = s + nje_rresht_i_tekstit + "\n";
```

Rikujto se `'\n'` është karakteri për rresht të ri.

Përgjigje:

```
public static void main(String[] args)
{ String teksti = "";
  boolean duke_perpunuar = true;
  while ( duke_perpunuar )
        { String hyrja = JOptionPane.showInputDialog("Shkruaj ca tekst:");
          if ( hyrja == null          // a u shtyp Cancel?
             || hyrja.equals("") )    // a eshte futur rresht i zbrazet?
                                      // (vetem Enter)
               { duke_perpunuar = false; }
          else { teksti = teksti + hyrja + "\n"; }
        }
  System.out.println(teksti);
}
```

3. Rishkruaje menaxherin e kontove të bankës nga kapitulli 6 ashtu që të përdorë cikël-while për transakcionet e veta hyrëse.
(Udhëzim: Rishkruaje `perpunoTransaksionet (processTransactions)` në klasën `KontrolluesILlogarise (AccountController)` në Figurën 16, Kapitulli 6.)

Përgjigje:

Metoda `perpunoTransaksionet` tani duket kështu:

```
...
  /** perpunoTransaksionet i perpunon komandat e perdoruesit derisa te jepet Q*/
  public void perpunoTransaksionet()
  { boolean duke_perpunuar = true;
    while ( duke_perpunuar )
          // deri tash, te gjitha transaksionet jane perpunuar korrektesisht
          { char komanda = lexuesi.lexoKomanden("Komanda (D,T,Q) dhe sasia:");
            if ( komanda == 'Q' )  // sinjali per t'u ndalur
                 { duke_perpunuar = false; }
            else if ( komanda == 'D' )  // depozitim ?
                      { ... shih Figuren 12, Kapitulli 6 ne liber... }
                   else if ( komanda == 'T' )  // terheqje?
                      { ... shih Figuren 12, Kapitulli 6 ne liber... }
                   else { shkruesi.tregoTransaksionin("Komande e parregullt"); }

          }
  }
}
```

7.6 Urdhëri `for`

1. Rishkruaje metodën `llogariteMesataren` në Figurën 1 ashtu që të përdorë urdhër-for.

Përgjigje:

```
public double llogariteMesataren(int sa)
{ double total_pike = 0.0;  // totali i te gjitha pikeve
  int numri = 0;  // sasia e testeve te lexuara deri tash
  for ( int numri = 0;  numri != sa;  numri = numri + 1 )
      // ne cdo iteracion: total_pike == prov_1 + prov_2 + ... + prov_numri
      { String hyrja = JOptionPane.showInputDialog("Piket e prov. tjeter:");
        int piket = new Integer(hyrja).intValue();
        total_pike = total_pike + piket;
        System.out.println("numri = " + numri + "; totali = " + total_pike);
      }
  return (total_pike / sa);
}
```

2. Përdore një urdhër-for për ta shkruar një funksion, `'ktheje'`, që e pranon një parametër string, `s`, dhe e kthen si rezultat të vetin stringun që duket si `s` e kthyer mbrapsht, p.sh. `ktheje("abcd")` kthen `"dcba"`.

Përgjigje:

```
/** ktheje e kthen mbrapsht nje argument string
  * @param s - stringu qe do te kthehet
  * @return (kthen) stringun e kthyer mbrapsht,
  *         p.sh. kthen "cba" kur  s  eshte "abc" */
public String ktheje(String s)
{ String pergjigjja = "";
  for ( int i = 0;  i != s.length();  i = i+1 )
      // invariant: pergjigjja i mban  i  karakteret e pare te  s
      // ne renditje te kundert
      { pergjigjja = s.charAt(i) + pergjigjja; }
  return pergjigjja;
}
```

3. Rishkruaje metodën në Figurën 2 që ta përdorë një urdhër-for.

4. Ciklet-while që i ke shkruar në përgjigjet e ushtrimeve te "Iterimi definitiv" - tash rishkruaji si cikle-for.

7.7 Ciklet e Ndërfutura

1. Shkruaji ciklet e ndërfutura që e japin tabelën e mbledhjes për 0+0 deri në 5+5.

Përgjigje:

```
for (int i = 0; i <= 5; i = i + 1 )
    { for ( int j = 0; j <= 5; j = j + 1 )
          { System.out.print(i + "+" + j + "=" + (i+j) + " "); }
      System.out.prinln();
    }
```

2. Shkruaje këtë metodë:

```
/** hiqiShkronjatEDyfishta e konstrukton nje string qe i përmban
  * shkronjat e njejta si argumenti i vet, por te gjitha shkronjat
  * e dyfishta (duplikatet) hiqen,
  * p.sh., per argument "fluturat", rezultati eshte "flutra"
  * @param s - stringu argument
  * @return (kthen) nje string qe duket si s por pa duplikate
public String hiqiShkronjatEDyfishta(String s)
```

Përgjigje:

Cikli i ndërfutur gjendet në metodën ndihmëse, `nukGjendetNe`:

```
/** hiqiShkronjatEDyfishta e konstrukton nje string qe i permban
  * shkronjat e njejta si argumenti i vet, por te gjitha shkronjat
  * e dyfishta (duplikatet) hiqen,
  * p.sh., per argument "fluturat", rezultati eshte "flutra"
  * @param s - stringu argument
  * @return (kthen) nje string qe duket si s por pa duplikate
public String hiqiShkronjatEDyfishta(String s)
{ String pergjigjja = "";
  for ( int i = 0; i != s.length(); i = i + 1 )
      // invariant: pergjigjja e mban saktesisht nga nje kopje
      // te cdo karakteri qe paraqitet ne nenstringun
      // s.charAt(0) ..deri ne.. s.charAt(i-1)
      { if ( nukGjendetNe(pergjigjja, s.charAt(i)) )
           { pergjigjja = pergjigjja + s.charAt(i); }
      }
  return pergjigjja;
}

/** nukGjendetNe(String a, char c)
{ boolean nukgjendet = true;
  for ( int j = 0; j != a.length(); j = j + 1 )
      // invariant: nukgjendet tregon se a mungon c
      // ne nenstringun a.charAt(0)..deri ne.. a.charAt(j-1)
      { nukgjendet = nukgjendet && (c != a.charAt(j)); }
  return nukgjendet;
}
```

3. Shkruaji ciklet e ndërfutura që e afishojnë këtë model:

```
0 0
1 0   1 1
2 0   2 1   2 2
3 0   3 1   3 2   3 3
```

Përgjigje:

```
for ( int i = 0; i <= 3; i = i + 1 )
      for ( int j = 0; j <= i; j = j + 1 )
          { System.out.print(i + " " + j); }
        System.out.println();
    }
```

4. Shkruaji ciklet e ndërfutura që e afishojnë këtë model:

```
0 3   0 2   0 1   0 0
1 3   1 2   0 1
2 3   2 2
3 3
```

Përgjigje:

```
for ( int i = 0; i <= 3; i = i + 1 )
    { for ( j = 3; j >= i; j = j + 1 )
          { System.out.print(i + " " + j + "   "); }
      System.out.println();
    }
```

(7.8 Shkruarja dhe Testimi i Cikleve)

7.9 Case Study: Animim i Topave Kërcyes

Qe klasët e animimit:

```
// Fajli TopLevizes.java

/** TopLevizes e modelon nje top qe leviz */
public class TopLevizes
{ private int x_poz;  // x-pozita e qendres se topit
  private int y_poz;// y-pozita e qendres se topit
  private int rrezja;  // rrezja e topit

  private int x_shpejtesia = +5;  // shpejtesia horizontale; pozitive = djathtas
  private int y_shpejtesia = +2;  // shpejtesia vertikale; pozitive = teposhte

  private Kuti mbajtesi;      // mbajtesi ne te cilin udheton topi

  /** Konstruktori TopLevizes e konstrukton topin.
    * @param x_fillues - qendra horizontale e pozites filluese te topit
    * @param y_fillues - qendra vertikale e pozites filluese te topit
    * @param r - rrezja e topit
    * @param kutia - permbajetesi ne te cilin udheton topi */
  public TopLevizes(int x_fillues, int y_fillues, int r, Kuti kutia)
  { x_poz = x_fillues;
    y_poz = y_fillues;
    rrezja = r;
    mbajtesi = kutia;
  }

  /** xPozita e kthen poziten horizontale momentale te topit */
  public int xPozita()
  { return x_poz; }

  /** yPozita e kthen poziten vertikale momentale te topit */
  public int yPozita()
  { return y_poz; }

  /** rrezjaE e kthen rrezen e topit */
  public int rrezjaE()
  { return rrezja; }

  /** levize e leviz topin
    * @param njesi_kohe - sasia e kohes prejse eshte levizur topi
    *                     heren e fundit */
  public void leviz(int njesi_kohe)
  { x_poz = x_poz + (x_shpejtesia * njesi_kohe);
    if ( mbajtesi.neKontaktHorizontal(x_poz) )
       { x_shpejtesia = -x_shpejtesia; } // ktheje drejtimin horizontal prapa
    y_poz = y_poz + (y_shpejtesia * njesi_kohe);
    if ( mbajtesi.neKontaktHorizontal(y_poz) )
       { y_shpejtesia = -y_shpejtesia; } // ktheje drejtimin vertikal prapa
  }

  // vazhdon ne faqen tjeter...
```

```
  //...vazhdon nga faqja e përparshme

  public static void main(String[] args)
  { Kuti kutia = new Kuti(50); // me madhesi 50 piksela me 50 piksela
    TopLevizes topi = new TopLevizes(25, 25, 10, kutia); // rrezja = 10 piksela
    while ( true )
          { topi.leviz(1); // 1 njesi te kohes se kaluar
            System.out.println("x = " + topi.xPozita()
              + "; y = " + topi.yPozita());
          }
  }
}

------------------------------

// Fajli Kuti.java

/** Kuti e modelon nje kuti ne te cilen jetojne objektet levizese */
public class Kuti
{ private int madhesia_e_kutise;

  /** Konstruktori Kuti e nderton kutine
    * param madhesia - madhesia e kutise */
  public Kuti(int madhesia)
  { madhesia_e_kutise = madhesia; }

  /** neKontaktHorizontal pergjigjjet:
    *  a ka kontaktuar pozita momentale ndonje mur horizontal?
    * @param x_pozita - pozita qe testohet
    * @return (kthen) true, nese x_pozita eshte baraz ose i kalon pozitat
    * e mureve horizontale; perndryshe return (kthen) false */
  public boolean neKontaktHorizontal(int x_pozita)
  { return ( x_pozita <= 0 ) || (x_pozita >= madhesia_e_kutise); }

  /** neKontaktVertikal pergjigjet: a ka kontaktuar pozita ndonje mur vertikal
    * @param y_pozita - pozita e testuar
    * return (kthen) true, nese y_pozita eshte baraz ose i tejkalon pozitat
    *   e mureve vertikale; perndryshe, return (kthen) false */
  public boolean neKontaktVertikal(int y_pozita)
  { return (y_pozita <= 0) || (y_pozita >= madhesia_e_kutise); }

  /** madhesiaE e kthen madhesine e kutise */
  public int madhesiaE()
  { return madhesia_e_kutise; }
}
```

```
// Fajli VizatuesAnimacioni.java

import java.awt.*;
import javax.swing.*;
/** VizatuesAnimacioni e paraqet nje kuti me nje top ne te. */
public class VizatuesAnimacioni extends JPanel
{ private VizatuesKutie vizatuesi_i_kutise; // pamja dalese e kutise
  private VizatuesTopi vizatuesi_i_topit;   // pamja dalese e topit ne kuti

  /** Konstruktori VizatuesAnimacioni e konstrukton pamjen e kutise dhe te topit
    * @param k - vizatuesi i kutise
    * @param t - vizatuesi i topit
    * @param madhesia - madhesia e kornizes */
  public VizatuesAnimacioni(VizatuesKutie k, VizatuesTopi t, int madhesia)
  { vizatuesi_i_kutise = k;
    vizatuesi_i_topit = t;
    JFrame korniza_ime = new JFrame();
    korniza_ime.getContentPane().add(this);
    korniza_ime.setTitle("Gjuajtja e Topit");
    korniza_ime.setSize(madhesia, madhesia);
    korniza_ime.setVisible(true);
  }

  /** paintComponent paints the box and ball
    * @param g - the graphics pen */
  public void paintComponent(Graphics g)
  { vizatuesi_i_kutise.vizato(g);
    vizatuesi_i_topit.vizato(g);
  }

  /** main - vetem per testim */
  public static void main(String[] args)
  { Kuti k = new Kuti(200);
    TopLevizes t = new TopLevizes(50, 50, 10, k);
    VizatuesKutie vk = new VizatuesKutie(k);
    VizatuesTopi vt = new VizatuesTopi(t, Color.red);
    new VizatuesAnimacioni(vk, vt, 50);
  }
}
```

```
// Fajli VizatuesKutie.java

import java.awt.*;
/** VizatuesKutie e paraqet nje kuti */
public class VizatuesKutie
{ private Kuti kutia; // objekti kuti qe do te paraqitet

  /** Konstruktori VizatuesKutie e paraqet kutine
    * @param k - kutia qe paraqitet */
  public VizatuesKutie(Kuti k)
  { kutia = k; }

  /** vizato e vizaton kutine
    * @param g - the penda grafike e perdorur per ta vizatuar kutine */
  public void vizato(Graphics g)
  { int madhesia = kutia.madhesiaE();
    g.setColor(Color.white);
    g.fillRect(0, 0, madhesia, madhesia);
    g.setColor(Color.black);
    g.drawRect(0, 0, madhesia, madhesia);
  }
}

-------------------------------------------

// Fajli VizatuesTopi.java

import java.awt.*;
/** VizatuesTopi e paraqet ne ekran nje top levizes */
public class VizatuesTopi
{ private TopLevizes topi; // objekti top i paraqitur (adresa e vet)
  private Color ngjyra_e_topit;

  /** Konstruktori VizatuesTopi
    * @param x - topi qe do te paraqitet
    * @param n - ngjyra e tij */
  public VizatuesTopi(TopLevizes x, Color n)
  { topi = x;
    ngjyra_e_topit = n;
  }

  /** vizato e vizaton topin ne pamje (ekran)
    * @param g - penda grafike e perdorur per ta vizatuar topin */
  public void vizato(Graphics g)
  { g.setColor(ngjyra_e_topit);
    int rrezja = topi.rrezjaE();
    g.fillOval(topi.xPozita() - rrezja, topi.yPozita() - rrezja,
               rrezja * 2, rrezja * 2);
  }
}
```

```
// Fajli GjuajtjeKontrollues.java

/** GjuajtjeKontrollues e kontrollon nje top levizes brenda nje kutie. */
public class GjuajtjeKontrollues
{ private TopLevizes topi; // objekti model
  private VizatuesAnimacioni vizatuesi; // objekti i pamjes-dalese

  /** Konstruktori GjuajtjeKontrollues e nis kontrolluesin
    * @param t - objekti model
    * @param v - objekti i pamjes dalese */
  public GjuajtjeKontrollues(TopLevizes t, VizatuesAnimacioni v)
  { topi = t;
    vizatuesi = v;
  }

  /** niseAnimacionin e nis animacionin sipas ores se brendshme */
  public void niseAnimacionin()
  { int njesia_e_kohes = 1; // njesia e kohes per secilin hap te animacionit
    int vonesa_e_vizatimit = 20; // sa te vonohet ndermjet dy rivizatimeve
    while ( true )
    { prit(vonesa_e_vizatimit);
      topi.leviz(njesia_e_kohes);
      System.out.println(topi.xPozita() + ", " + topi.yPozita());
      vizatuesi.repaint(); // rishfaqe kutine dhe topin
    }
  }

  /** prit e ndal ekzekutimin per 'sa' milisekonda */
  private void prit(int sa)
  { try { Thread.sleep(sa); }
    catch (InterruptedException e) { }
  }
}

-----------------------------------

// Fajli GjuajeTopin.java

import java.awt.*;
/** GjuajeTopin i konstrukton dhe i nis objektet e animacionit. */
public class GjuajeTopin
{ public static void main(String[] args)
  { // konstrukto objektet e modelit:
    int madhesia_e_kutise = 200;
    int rrezja_e_topit = 6;
    Kuti kutia = new Kuti(madhesia_e_kutise);
    // vendose topin jo mu ne qender te kutise, po afer 3/5 e pozites:
    TopLevizes topi = new TopLevizes((int)(madhesia_e_kutise / 3.0),
                                     (int)(madhesia_e_kutise / 5.0),
                                     rrezja_e_topit, kutia);
    VizatuesTopi vizatuesi_i_topit = new VizatuesTopi(topi, Color.red);
    VizatuesKutie vizatuesi_i_kutise = new VizatuesKutie(kutia);
    VizatuesAnimacioni vizatuesi
            = new VizatuesAnimacioni(vizatuesi_i_kutise, vizatuesi_i_topit,
                                     madhesia_e_kutise);
    // konstruktoje kontrolluesin dhe nise:
    new GjuajtjeKontrollues(topi, vizatuesi).niseAnimacionin();
  }
}
```

1. Ekzekutoje animimin e topave kërcyes. Nganjëherë, do ta vëreni se si topi kthehet nga muri "shumë vonë", dmth. topi e ndërron drejtimin pasi të hyjë në mur. Gjeje burimin e këtij problemi dhe jep mënyra se si të rregullohet animacioni.

Përgjigje:

Metoda `leviz` brenda klasës `TopLevizes` së pari e lëviz topin, e pastaj e kontrollon pozitën e topit. Kështu, topi mund t'i depërtojë muret e kutisë. Për ta parandaluar këtë, `'leviz'` duhet ta kontrollojë pozitën e topit para se ta lëvizë topin dhe, nëse është e nevojshme, ta lëvizë topin pikërisht afër murit, dhe jo më tej.

2. Rregulloje animacionin e topit kërcyes ashtu që të jenë dy topa në kuti.
(Mos u brengos për përplasjen e dy topave.)

Përgjigje:

Për ta bërë animacionin me dy topa, krijo dy objekte `TopLevizes` dhe futi në kuti.
Qe ku janë klasët që i marrin ndërrimet e thjeshta:

```
import java.awt.*;
/** GjuajeTopin i konstrukton dhe i nis objektet ne animacion. */
public class GjuajeTopin2
{ public static void main(String[] args)
  { // konstrukto objektet e modelit:
    int madhesia_e_kutise = 200;
    int rrezja_e_topit = 6;
    Kuti kutia = new Kuti(madhesia_e_kutise);
    TopLevizes topi1 = new TopLevizes((int)(madhesia_e_kutise / 2.0),
                                      (int)(madhesia_e_kutise / 5.0),
                                       rrezja_e_topit, kutia);
    TopLevizes topi2 = new TopLevizes((int)(madhesia_e_kutise / 5.0),
                                      (int)(madhesia_e_kutise / 2.0),
                                       rrezja_e_topit, kutia);

    // konstrukto objektet e pamjes dalese:
    VizatuesTopi vizatuesi_i_topit1 = new VizatuesTopi(topi1, Color.red);
    VizatuesKutie vizatuesi_i_kutise = new VizatuesKutie(kutia);
    VizatuesTopi vizatuesi_i_topit2 = new VizatuesTopi(topi2, Color.blue);

    VizatuesAnimacioni2 vizatuesi
             = new VizatuesAnimacioni2(vizatuesi_i_kutise,
                                       vizatuesi_i_topit1, vizatuesi_i_topit2,
                                       madhesia_e_kutise);
    // konstruktoje kontrolluesin dhe nise:
    new GjuajtjeKontrollues2(topi1, topi2, vizatuesi).niseAnimacionin();
  }
}
```

```
import java.awt.*;
import javax.swing.*;
/** VizatuesAnimacioni2 e paraqet nje kuti me dy topa ne te. */
public class VizatuesAnimacioni2 extends JPanel
{ private VizatuesKutie vizatuesi_i_kutise;  // pamja dalese e kutise
  private VizatuesTopi vizatuesi_i_topit1;   // pamja dalese e topit1 ne kuti
  private VizatuesTopi vizatuesi_i_topit2;   // pamja dalese e topit2 ne kuti

  /** Konstruktori VizatuesAnimacioni e konstrukton pamjen e kutise dhe topit
    * @param k - vizatuesi i kutise
    * @param t1 - vizatuesi i topit te pare
    * @param t2 - vizatuesi i topit te dyte
    * @param madhesia - madhesia e kornizes */
  public VizatuesAnimacioni2(VizatuesKutie k,
                             VizatuesTopi t1,
                             VizatuesTopi t2,
                             int madhesia)
  { vizatuesi_i_kutise = k;
    vizatuesi_i_topit1 = t1;
    vizatuesi_i_topit2 = t2;
    JFrame korniza_ime = new JFrame();
    korniza_ime.getContentPane().add(this);
    korniza_ime.setTitle("Gjuaje");
    korniza_ime.setSize(madhesia, madhesia);
    korniza_ime.setVisible(true);
  }

  /** paintComponent paints the box and ball
    * @param g - the graphics pen */
  public void paintComponent(Graphics g)
  { vizatuesi_i_kutise.vizato(g);
    vizatuesi_i_topit1.vizato(g);
    vizatuesi_i_topit2.vizato(g);
  }

  /** main - vetem per testim */
  public static void main(String[] args)
  { Kuti k = new Kuti(200);
    TopLevizes t = new TopLevizes(50, 50, 10, k);
    VizatuesKutie vk = new VizatuesKutie(k);
    VizatuesTopi vt = new VizatuesTopi(t, Color.red);
    new VizatuesAnimacioni(vk, vt, 50);
  }
}
```

```
/** GjuajtjeKontrollues i kontrollon dy topa levizes brenda nje kutie. */
public class GjuajtjeKontrollues2
{ private TopLevizes topi1; // objekt model
  private TopLevizes topi2; // objekt model
  private VizatuesAnimacioni2 vizatuesi; // objekti i pamjes-dalese

  /** Konstruktori GjuajtjeKontrollues e nis kontrolluesin
    * @param t1 - top
    * @param t2 - top tjeter
    * @param v - objekti i pamjes dalese */
  public GjuajtjeKontrollues2(TopLevizes t1, TopLevizes t2,
                              VizatuesAnimacioni2 v)
  { topi1 = t1;
    topi2 = t2;
    vizatuesi = v;
  }

  /** niseAnimacionin e nis animacionin sipas ores se brendshme */
  public void niseAnimacionin()
  { int njesia_e_kohes = 1; // njesia e kohes per secilin hap te animacionit
    int vonesa_e_vizatimit = 20; // sa te vonohet ndermjet dy rivizatimeve
    while ( true )
    { prit(vonesa_e_vizatimit);
      topi1.leviz(njesia_e_kohes);
      topi2.leviz(njesia_e_kohes);
      // System.out.println(topi.xPozita() + ", " + topi.yPozita());
      vizatuesi.repaint(); // rishfaqe kutine dhe topin
    }
  }

  /** prit e ndal ekzekutimin per 'sa' milisekonda */
  private void prit(int sa)
  { try { Thread.sleep(sa); }
    catch (InterruptedException e) { }
  }
}
```

3. Nëse e ke kryer ushtrimin e kaluar, ripunoje animacionin ashtu që përplasja e dy topave të shkaktojë që të dy topat t'i kthejnë mbrapsht drejtimet horizontale dhe vertikale.

4. Vendose një pengesë të vogël në qendër të kutisë ashtu që topat të refuzohen nga pengesa.

(7.10 Rekursioni)

7.11 Numërimi me Rekursion

1. Shkruaje një aplikacion që i afishon vlerat e 3!, 6!, 9!, ... , 18! .

Përgjigje:

```
public class TestFaktoriel
{
  public static void main(String[] args)
  { for ( int i = 3;  i <= 20;  i = i+3 )
    // invariant: jane afishuar faktorialet prej 3 ..deri ne.. (i-1)
    { System.out.println( i + "! = " +  faktoriel(i) ); }
  }

  public static long faktoriel(int n)
  { ... }
}
```

2. Largoje nga metoda `'faktoriel'` urdhërin e parë kushtëzues, atë që e teston `n<0 || n>20`. Pastaj provo ta llogaritësh `faktoriel(20)`, `faktoriel(21)`, `faktoriel(99)`, dhe `faktoriel(-1)`.

Përgjigje:

Do ta marrësh këtë dalje:

```
20! = 2432902008176640000
21! = -4249290049419214848
99! = 0
Exception in thread "main" java.lang.StackOverflowError
        at TestFactorial.factorial(Compiled Code)
        at TestFactorial.factorial(Compiled Code)
         ...
```

Përgjigjja për 21! është qartas jokorrekte dhe paraqitet për shkak se ndryshoret `long` mbajnë numra prej rreth 20 shifrash të shumtën. Kur futet një numër më i madh, paraqitet teprimi (overflow) dhe disa shifra të rëndësishme të numrit cungohen (hiqen). (Në shembullin e mësipërm, përgjigjja e gjymtuar duket si numër negativ.) Një shembull edhe më ekstrem i këtij fenomeni ndodh te 99! Përfundimisht, tentimi për të llogaritur -1! shkakton një sasi të pakufizuar të thirrjeve rekursive, që e mbush plotësisht memorjen e kompjuterit me kopje të papërfunduara të `'faktoriel'`-it. Kjo shkakton gabim: `StackOverflowError`.

3. Me pak të menduar, mund ta përdorësh një cikël-while për t'i programuar barazimet rekurzive për faktoriel. Bëje këtë.

Përgjigje: Shih përgjigjen në ushtrimin 5 në njësinë "Iterimi definitiv".

4. Implemento Funksionin e Fibonaçit me një metodë që i përmban dy rekursione, dhe provoje funksionin tënd me parametra të vërtetë (aktualë) 20; 30; 35; 40. Pse llogaritja shkon aq ngadalë për këto raste të testimit?
(Meqë ra fjala, Funksioni i Fibonaçit ishte propozuar në shekullin e 13-të për t'i numëruar numrin e çifteve të lepujve të prodhuar nga një çift fillestar, duke supozuar se një çift i lepujve vonohet një muaj të piqet dhe prej atëherë e tutje çifti prodhon nga një çift të lepujve çdo muaj!)

Përgjigje:

```
/** fibonacci  e llogarit funksionin fibonacci */
public static int fibonacci(int n)
{ int pergjigjja;
  if ( n == 0  ||  n == 1 )
       { pergjigjja = 1; }
  else { pergjigjja = fibonacci(n-1) + fibonacci(n-2); }
  System.out.println("fibonacci(" + n + ") = " + pergjigjja);
  return pergjigjja;
}
```

Mund të llogarisim dhe të shohim se fibonacci(20) është 10946, fibonacci(30) është 1346269, fibonacci(35) është 14930352, dhe fibonacci(40) është 165580141. Këta nuk janë numra të mëdhenj, por llogaritja shkon ngadalë sepse funksioni rregullisht i rillogarit përgjigjet e njëjta. (Urdhëri System.out.println në metodë do ta tregojë këtë qartas!) Për shembull, fibonacci(20) dërgon mesazhe për llogaritjen e fibonacci(19) dhe fibonacci(18), pastaj fibonacci(19) dërgon mesazhe për t'u llogaritur fibonacci(18) dhe fibonacci(17), edhe pse fibonacci(18) tashmë është duke u llogaritur. Këto rillogaritje eksplodojnë derisa parametrat e vërtetë (aktualë) zvogëlohen në vlerë.
Në kapitullin tjetër mësojmë si t'i përdorim *vargjet* për t'iu ikur llogaritjeve të tepërta.

5. Qe një algoritëm rekursiv për ta llogaritur prodhimin e një vargu të numrave të plotë jonegativë:

```
prodhimi(a, b) = 1, kur b < a
prodhimi(a, b) = prodhimi(a, b-1) * b, kur b >= a
```

Shkruaje një metodë të përkufizuar në mënyrë rekursive që e llogarit prodhimin.

Përgjigje:

```
/** prodhimi e llogarit prodhimin a*(a+1)*(a+2)*...*b
  * @param  a - numri i plote fillues pwr prodhimin
  * @param  b - numri i plotw mbarues pwr prodhimin
  * @return (kthen) a*(a+1)*(a+2)*...*b, nese  a <= b;
  *  return (kthen) 1,  nese  a > b   */
public int prodhimi(int a, int b)
{ int totali;
  if ( b < a )
       { totali = 1; }
  else { totali = prodhimi(a, b-1) * b; }
  return totali;
}
```

Krahasoje këtë zgjidhje me atë të Ushtrimit 5 në pjesën "Iterimi definitiv".

6. Edhe një aktivitet i thjeshtë si mbledhja e dy numrave të plotë jonegativë mund të zgjidhet në mënyrë rekursive vetëm sipas "-1" dhe "+1":

```
mbledh(0, b) = b
mbledh(a, b) = mbledh(a-1, b) + 1, kur a > 0
```

a. Shkruaje një metodë të përkufizuar në mënyrë rekursive që e llogarit këtë përkufizim.
b. Në stil të ngjashëm, përkufizoje shumëzimin, `'shumëzo'`, të dy numrave të plotë jonegativë sipas "-1" dhe `'mbledh'`.
c. Përkufizoje fuqizimin, `'fuqi'`, të dy numrave të plotë jonegaticvë sipas "-1" dhe `'shumezo'`.

Përgjigje:

```
/** mbledh e llogarit a + b, per numrat jonegative  a dhe b  */
public int mbledh(int a, int b)
{ int rezultati;
  if ( a == 0 )
       { rezultati = b; }
  else { rezultati = mbledh(a-1, b) + 1; }
  return rezultati;
}

/** shumezo e llogarit a * b, per numrat jonegative  a dhe b  */
public int shumezo(int a, int b)
{ int rezultati;
  if ( a == 0 )
       { rezultati = 0; }
  else { rezultati = mbledh(shumezo(a-1, b), b); }
  return rezultati;
}

/** fuqi e llogarit  a^b, per numrat jonegative  a dhe b  */
public int fuqi(int a, int b)
{ int rezultati;
  if ( b == 0 )
       { rezultati = 1; }
  else { rezultati = shumezo(a, fuqi(a, b-1)); }
  return rezultati;
}
```

7. Funksioni i Akermanit është edhe një përkufizim i famshëm rekursiv:

```
A(0, n) = n + 1
A(m, 0) = A(m-1, 1), kur m>0
A(m, n) = A(m-1, A(m, n-1)), kur m > 0 dhe n > 0
```

Shkruaje një metodë të përkufizuar në mënyrë rekursive mbi këtë përkufizim. Funksioni është i famshëm sepse rekursionet e tij shkaktojnë që vlerat e parametrit të dytë të rriten derisa parametri i parë zvogëlohet ngadalë, por megjithatë funksioni kryhet gjithmonë.

Përgjigje:

```
public static int ackermann(int m, int n)
{ int rezultati;
  if ( m == 0 )
       { rezultati = n+1; }
  else if ( n == 0 )
       { rezultati = ackermann(m-1, 1); }
  else { rezultati = ackermann(m-1, ackermann(m, n-1)); }
  System.out.println("ackermann(" + m + ", " + n + ") = " + rezultati);
  return rezultati;
}
```

Provoje këtë metodë në shembuj të vegjël së pari, p.sh., `ackermann(2, 2)`. Pastaj, përdore orën tënde për ta matur se sa kohë po i duhet kompjuterit tënd që ta llogarisë `ackermann(3, 5)`.

8. Themi se një fjalë është *e sortuar* nëse të gjitha shkronjat e saj janë të ndryshme dhe janë të renduara në radhitje alfabetike, p.sh. "bdez" është e sortuar por "ba" dhe "bbd" nuk janë. Shkruaje një algoritëm rekursiv që e llogarit numrin e fjalëve të sortuara të gjatësisë n ose më të vogël që dikush mund t'i krijojë nga n shkronja të ndryshme.

Përgjigje:

```
numri_i_fjaleve_te_sortuara(1) = 1
```

Një shkronjë e vetme është fjalë e sortuar.

```
numri_i_fjaleve_te_sortuara(n) = q + 1 + q,
   ku q = numri_i_fjaleve_te_sortuara(n-1)
```

Qe arsyetimi: E zëmë se i kemi shkruar q fjalë të ndryshme të sortuara me n-1 shkronja të ndryshme, dhe e zëmë se '*z*' është shkronjë e re që është e ndryshme nga të gjitha shkronjat e përdorura deri tash. Fjalët e sortuara që mund t'i shkruajmë tash përfshijnë *(i)* ato që tashmë i kemi shkruar; *(ii)* '*z*' vetëm; *(iii)* të gjitha fjalët e reja të sortuara që mund t'i ndërtojmë nga fjalët e vjetra të sortuara plus '*z*'. Në rastin e fundit, janë saktësisht q fjalë të tilla të sortuara, që konstruktohen duke e futur '*z*' në pozitën e vet të duhur brenda secilës fjalë të sortuar më parë.

7.12 Vizatimi i Figurave Rekursive

1. Rishkruaje klasën `VizatuesRekursivFigurash` duke e hequr kufirin përreth secilës figurë; pastaj, lëvizi vezët ashtu që ato të rrinë në këndin e majtë të figurës, e jo në të djathtin.

Përgjigje:

Ripunoje metodën `vizatojeFiguren`:

```
  private void vizatojeFiguren(int kufiri_djathtas, int poshte,
                               int madhesia_e_vezes,
                               Graphics g)
  { int madhesia_e_vezes_prapa = (int)(madhesia_e_vezes * SHKALLA);
    if ( madhesia_e_vezes_prapa > 0 )  // a ia vlen te vizatohet prapavija
       { vizatojeFiguren((int)(kufiri_djathtas * SHKALLA),
                         (int)(poshte * SHKALLA),
                         madhesia_e_vezes_prapa,
                         g);
       }
    // vizatoje nje veze perpara
    vizatojeNjeVeze(0, poshte, madhesia_e_vezes, g);
  }
```

2. Shkruaje një klasë që e krijon një rreth me diametër prej 200 pikselash, në të cilin është një tjetër rreth sa 0.8 e madhësisë së të parit, në të cilin është një rreth tjetër sa 0.8 e madhësisë së të dytit, ..., derisa rrethi zvogëlohet në madhësinë 0.

Përgjigje:

```
import java.awt.*;
import javax.swing.*;
/** VizatuesIRratheve e vizaton nje koleksion te rratheve bashkeqendrore  */
public class VizatuesIRratheve extends JPanel
{ private double SHKALLA = 0.8;  // madhesia e figures prapa ne relacion
                                 // me figuren perpara
  private int GJERESIA = 300;
  private int DIAMETRI = 200;

  /** Konstruktori VizatuesIRratheve e krijon dritaren */
  public VizatuesIRratheve()
  { JFrame korniza_ime = new JFrame();
    korniza_ime.getContentPane().add(this);
    korniza_ime.setTitle("Rrathet");
    korniza_ime.setSize(GJERESIA, GJERESIA);
    korniza_ime.setBackground(Color.white);
    korniza_ime.setVisible(true);
  }
```

```
  /** paintComponent i vizaton rrathet
    * @param g - ''penda grafike'' */
  public void paintComponent(Graphics g)
  { vizatoRrathet(DIAMETRI, g); }

  private void vizatoRrathet(int madhesia, Graphics g)
  { if ( madhesia > 0 )
       { int pozita_e_kendit = (GJERESIA / 2) - (madhesia / 2);
         g.drawOval(pozita_e_kendit, pozita_e_kendit, madhesia, madhesia);
         vizatoRrathet((int)(madhesia * 0.8), g);
       }
  }

  public static void main(String[] args)
  { new VizatuesIRratheve(); }
}
```

3. Për ta kuptuar më mirë klasën `VizatuesIVezëve`, vepro si vijon. Së pari, zëvendësoje metodën `vizatoNjëVezë` me këtë version, që e bën një vezë të tejdukshme:

```
private void vizatojeNjeVeze(int skaji_majtas, int fundi,
                             int gjeresia, Graphics g)
{ int lartesia = (2 * gjeresia) / 3;
  int lart = poshte - lartesia;
  g.setColor(Color.black);
  g.drawOval(skaji_majtas, lart, gjeresia, lartesia);
}
```

Tash provoje përsëri `VizatuesIVezes`.

VIII Struktura e të Dhënave: Vargjet

8.1 Pse na Duhen Vargjet

1. Të themi se e deklarojmë këtë varg:

```
int r = new int[4];
```

Çka afishojnë secili nga këto cikle? (Udhëzim: do të jetë e dobishme të vizatohet një vizatim i vargut `r`, si ato që janë në këtë paragraf në libër, dhe të rifreskohet vizatimi përderisa kalon nëpër ekzekutimin e secilit cikël.)

(a)

```
for ( int i = 0; i < 4; i = i + 1 )
    { System.out.println(r[i]); }
```

Përgjigje:

```
0
0
0
0
```

(b)

```
int i = 1;
r[i] = 10;
r[i + 2] = r[i] + 2;
for ( int i = 0; i < 4; i = i + 1 )
{ System.out.println(r[i]); }
```

Përgjigje:

```
0
10
0
12
```

(c)

```
for ( int i = 3; i >= 0; i = i - 1 )
    { r[i] = i * 2; }
for ( int i = 0; i < 4; i = i + 1 )
    { System.out.println(r[i]); }
```

Përgjigje:

```
0
2
4
6
```

(d)

```
r[0] = 10;
for ( int i = 1; i != 4; i = i + 1 )
    { r[i] = r[i - 1] * 2; }
for ( int i = 0; i < 4; i = i + 1 )
    { System.out.println(r[i]); }
```

Përgjigje:

```
10
20
40
80
```

2. Deklaro një varg, `fuqite_e_dyshit`, që i mban 10 numra të plotë; shkruaje një cikël-for që e vendos në `fuqite_e_dyshit[i]` vlerën e 2^i, për të gjitha vlerat e `i` në intervalin prej 0 deri në 9.

Përgjigje:

```
int[] fuqite_e_dyshit = new int[10];
for ( int i = 0; i != 10; i = i + 1 )
{ fuqite_e_dyshit[i] = Math.pow(2, i); }
```

ose

```
int [] fuqite_e_dyshit = new int[10]
fuqite_e_dyshit[0] = 1;
for ( int i = 1; i != 10; i = i + 1 )
   { fuqite_e_dyshit[i] = 2 * fuqite_e_dyshit[i-1]; }
```

3. Deklaro një varg, `shkronja`, që i mban 26 karakterë. Shkruaje një cikël-for që e inicializon vargun me karakterët nga `'a'` deri në `'z'`. Pastaj, shkruaje një cikël që e lexon përmbajtjen e vargut `'shkronja'` dhe e afishon, dmth., shkronjat në alfabetin anglez, të gjitha në një rresht, në radhitje të kundërt.

Përgjigje:

```
char[] shkronja = new char[26];
for ( char c = 'a'; c <= 'z'; c = c + 1 )
    { shkronja[int(c-'a')] = c; }
for ( int i = 25; i >= 0; i = i - 1 )
    { System.out.print(shkronja[i]); }
System.out.println();
```

ose

```
char[] shkronja = new char[26];
for ( int i = 0; i != 26; i = i + 1 )
    { shkronja[i] = 'a' + i; } // Java na lejon t'i shtojme nr
                               // te plote karakterit per te
                               // formuar karakter tjeter
for ( int i = 25; i >= 0; i = i - 1 )
    { System.out.print(shkronja[i]); }
System.out.println();
```

4. Deklaro një varg, `reciproket`, që i mban 10 `double`; shkruaje një cikël-for që e vendos në `reciproket[i]` vlerën e `1.0 / i`, për të gjitha vlerat e i në intervalin prej 1 deri në 9. (Çfarë vlere mbetet në `reciproket[0]`?)

Përgjigje:

```
double[] reciproket = new double[10];
for ( int i = 1; i != 10; i = i + 1 )
    { reciproket[i] = 1.0 / i; }
```

Në `reciproket[0]` mbetet vlera `0.0`.

8.2 Mbledhja e të Dhënave Hyrëse në Vargje

Qe klasa `NumeruesIVotave`:

```
import javax.swing.*;
/** NumeruesIVotave i shenon votat per kandidatet e zgjedhjeve.
  * hyrja: nje varg votash, i perfunduar me  -1
  * dalja: lista e kandidateve dhe votat e tyre te shenuara */
public class NumeruesIVotave
{ public static void main(String[] args)
  { int num_kandidateve = 4; // sa kandidate
    int[] votat = new int[num_kandidateve]; // i mban votat
    // mbledhi votat:
    boolean duke_perpunuar = true;
    while ( duke_perpunuar )
          // invariant: te gjitha votat e lexuara jane numeruar ne vargun  votat
          { int v = new Integer(JOptionPane.showInputDialog
                        ("Vota per (0,1,2,3):")).intValue();
            if ( v >= 0 && v < votat.length ) // a eshte vote e rregullt
                 { votat[v] = votat[v] + 1; }
            else { duke_perpunuar = false; } // dil nese ka vote te parregullt
          }
    // afisho totalet:
    for ( int i = 0;  i != votat.length;  i = i + 1 )
        // totalet per votat[0]..votat[i-1] jane afishuar
        { System.out.println("Kandidati" + i + " i ka " + votat[i] + " vota"); }
  }
}
```

1. Modifikoje klasën `NumeruesIVotave` në Figurën 1 ashtu që të afishojë numrin total të votave të marrura dhe "emrin" e kandidatit fitues.

Përgjigje:

Pjesa e programit që e kryen atë që kërkohet është:

```
    int totali = 0; // totali i te gjitha votave
    int indeksi_max = 0; // indeksi i kandidatit me më së shumti vota
    for (int i = 0; i != num_kandidateve; i = i + 1)
        { totali = totali + votat[i];
          if ( votat[i] > votat[indeksi_max])
             { indeksi_max = i; }
        }
    System.out.println("Gjithsej vota: " + totali);
    System.out.println("Kandidati fitues: Kandidati" + indeksi_max);
```

Por kjo zgjidhje nuk punon mirë për rastin kur dy kandidatë kanë numër të njëjtë votash. Qe një zgjidhje që e merr parasysh edhe atë rast (shtesat që e dallojnë këtë kod nga shembulli i kaluar janë shënuar me (*)):

```
import javax.swing.*;
/** NumeruesIVotave i shenon votat per kandidatet e zgjedhjeve.
  * hyrja: nje varg votash, i perfunduar me  -1
  * dalja: lista e kandidateve dhe votat e tyre te shenuara */
public class NumeruesIVotave3
{ public static void main(String[] args)
  { int num_kandidateve = 4; // sa kandidate
    int[] votat = new int[num_kandidateve]; // i mban votat
    int numri_i_votave = 0;  // (*) e mban n'mend numrin e votave te marrura
    // mbledhi votat:
    boolean duke_perpunuar = true;
    while ( duke_perpunuar )
       // invariant: te gjitha votat e lexuara jane numeruar ne vargun  votat
          { int v = new Integer(JOptionPane.showInputDialog
                          ("Vota per (0,1,2,3):")).intValue();
            if ( v >= 0 && v < votat.length ) // a eshte vote e rregullt
                 { votat[v] = votat[v] + 1; }
            else { duke_perpunuar = false; } //dil nese ka vote te parregullt
          }
    System.out.println("Gjithsej vota te marrura: " + numri_i_votave); //*
    // afisho totalet:
    for ( int i = 0;  i != votat.length;  i = i + 1 )
        // totalet per votat[0]..votat[i-1] jane afishuar
        { System.out.println("Kandidati" + i + " i ka " + votat[i] + " vota"); }
    int fituesi = 0; // (*) e mban n'mend fituesin e zgjedhjeve
    boolean ka_barazim = false; // (*) e mban n'mend se a jane zgjedhjet baraz
    for (int i = 1; i != num_kandidateve; i = i + 1) // (*)
        { if ( votat[i] > votat[fituesi] )
             { fituesi = i;
               ka_barazim = false;
             }
          else if ( votat[i] == votat[fituesi] )
                   { ka_barazim = true; }
        }

  // vazhdon ne faqen tjeter
```

```
    // vazhdon nga faqja e meparshme

    if ( ka_barazim )  // (*)
         { System.out.println("Zgjedhjet jane baraz!"); }
    else { System.out.println("Fitues eshte Kandidati " + fituesi); }
  }
}
```

2. Modifikoje klasën `NumeruesIVotave` ashtu që aplikacioni së pari pyet për numrin e kandidatëve në zgjedhje. Pasi të jepet numri, atëherë votat mirren si zakonisht dhe afishohen rezultatet.

Përgjigje:

Ndërrimi kyç është ndërrimi i urdhërit, `int num_kandidateve = 4`, me

```
int num_kandidateve = new Integer(JOptionPane.showInputDialog
                        "Jepe numrin e kandidateve: ").intValue();
```

3. Modifiko klasën `NumeruesIVotave` ashtu që aplikacioni së pari i kërkon emrat e kandidatëve. Pasi të shkruhen emrat, votat mirren si zakonisht dhe rezultatet afishohen me emrin e secilit kandidat dhe votat përkatëse.
(Udhëzim: Përdore një varg të tipit `String[]` për t'i mbajtur emrat.)

Përgjigje: Në fillim të programit futi këta urdhëra, që i lexojnë emrat e kandidatëve:

```
String[] emri = new String[num_kandidateve]; // (*) i mban emrat e kandidateve
for ( int i = 0;  i != num_kandidateve;  i = i+1 )
    { emri[i] = JOptionPane.showInputDialog("Shkruaje emrin e Kandidatit " + i);
}
```

Në fund të programit, ndërroje ciklin që i afishon totalët e kandidatëve si vijon:

```
for ( int i = 0;  i != num_kandidateve;  i = i + 1 )
    { System.out.println("Kandidati " + emri[i] + " i ka "
                          + votat[i] + " vota");
    }
```

4. Shkruaje një aplikacion që e lexon një varg të numrave të plotë në intervalin prej 1 deri në 20. Hyrja kryhet me një numër të plotë që nuk bie në këtë interval. Për dalje të tij, aplikacioni e afishon numrin e plotë që është paraqitur më së shpeshti në hyrje, numrin e plotë që është paraqitur më së rralli në hyrje dhe mesataren e të gjitha hyrjeve.

Përgjigje: Zgjidhja e këtij ushtrimi është në thelb e njëjtë me zgjidhjen e Ushtrimit 1.

8.3 Tabelat Përkthyese

1. Shkruaje aplikacionin e plotë që e merr një numër amë dhe një rresht të fjalëve si hyrje dhe e prodhon një varg të kodeve (numra të plotë) si dalje.

Përgjigje:

```
import javax.swing.*;
public class Perkthe
{ public static void main(String[] args)
  { int[] kodi = new int[27]; // kjo është tabela përkthyese:
                              // kodi[0] e mban kodin për ' ',
                              // kodi[1] e mban kodin për 'a',
                              // kodi[2] e mban kodin për 'b', etj.
    int k = new Integer(JOptionPane.showInputDialog
                        ("Shkruaje amën (numër i plotë): ")).intValue();
    kodi[0] = k;
    for ( int i = 1; i != kodi.length; i = i+1 )
        { kodi[i] = (kodi[i-1] * 2) + 1; }
    String rreshti_hyres = JOptionPane.showInputDialog
                           ("Shkruaje fjalinë që do të kodohet:");
    for ( int j = 0; j != rreshti_hyres.length(); j = j+1 )
        { char c = rreshti_hyres.charAt(j);
          if ( c == ' ' )
               { System.out.println(kodi[0]); }
          else if ( c >= 'a' && c <= 'z' )
               { int indeksi = (c - 'a') + 1;
                 System.out.println(kodi[indeksi]);
               }
          else { System.out.println("gabim: karakter i keq në hyrje"); }
        }
  }
}
```

2. Shkruaje programin përkatës dekodues, që e merr numrin amë dhe hyrjen e tij të parë dhe pastaj e lexon një varg numrash të plotë, që dekodohen në karakterë dhe afishohen.

Përgjigje:

```
import javax.swing.*;
public class Dekodo
{ public static void main(String[] args)
  { int[] kodi = new int[27]; // kjo është tabela e përkthimit
                              // kodi[0] e mban kodin për ' ',
                              // kodi[1] e mban kodin për 'a',
                              // kodi[2] e mban kodin për 'b', etj.
    int k = new Integer(JOptionPane.showInputDialog
                        ("Shkruaje amën (numër i plotë): ")).intValue();
    kodi[0] = k;
    for ( int i = 1; i != kodi.length; i = i+1 )
        { kodi[i] = (int)((kodi[i-1] * 1.3) + 1); } //tjeter sistem i kodimit
    String pergjigjja = "";
    boolean duke_perpunuar = true;
    while ( duke_perpunuar )
          { String hyrja = JOptionPane.showInputDialog
                           ("Shkruaje një numër të plotë për ta dekoduar" +
                            "(ose asgjë, për të përfunduar): ");
            if ( hyrja.equals("") )
                 { duke_perpunuar = false; }
            else { char c = dekodo(kodi, new Integer(hyrja).intValue());
                   pergjigjja = pergjigjja + c;
                 }
          }
    System.out.println(pergjigjja);
  }

  /** dekodo  e përkthen një kod (numër i plotë) në karakter
    * @param kodi - vargu që i mban kodet për ' ', 'a', 'b', ..., 'z'
    * @param i - numri i plotë që do të dekodohet
    * @return (kthen) karakterin përkatës, ose '*' nëse numri nuk dekodohet */
  private static char dekodo(int[] kodi, int i)
  { char c = '*';
    boolean u_gjet = false;
    int indeksi = 0;
    while ( !u_gjet && indeksi != kodi.length )
          { if ( kodi[indeksi] == i )
                 { u_gjet = true;
                   if ( indeksi == 0 )
                        { c = ' '; }
                   else { c = (char)(indeksi + 'a' - 1); }
                 }
            else { indeksi = indeksi + 1; }
          }
    return c;
  }
}
```

3. Shkruaji urdhërat që e konstuktojnë një tabelë përkthyese, `fuqitë e dyshit`, dhe i japin vlerat tabelës `fuqitë e dyshit`, dmth.,

$$\texttt{fuqitë\_e\_dyshit[i]} = 2^{i}$$

për të gjitha vlerat e `i`-së në intervalin 0-9.

Përgjigje:

```
int[] fuqitë_e_dyshit = new int[10];
fuqitë_e_dyshit[0] = 1;
for ( int i = 1; i != 10; i = i + 1 )
    { fuqitë_e_dyshit[i] = fuqitë_e_dyshit[i - 1] * 2; }
```

4. Tabelat përkthyese e kanë një lidhje të veçantë me barazimet e përkufizuara në mënyrë rekursive, si ato që i kemi parë në Kapitullin e mëparshëm. Për shembull, ky përkufizim rekursiv i shumës:

```
shuma(0) = 0
shuma(n) = n + shuma(n-1), nëse n > 0
```

e përkufizon tabelën përkthyese vijuese:

```
int[] shuma = new int[...];
shuma[0] = 0;
for (int n = 1; n != shuma.length; n = n + 1 )
    { shuma[n] = n + shuma[n-1]; }
```

Shkruaji aplikacionet që i ndërtojnë tabelat (vargje me nga 20 elemente) për përkufizimet vijuese rekursive; ndërto aplikacionet që do ta afishojnë përmbajtjen e secilës tabelë, në renditje të kundërt.

(a) Funksioni faktoriel:

```
0! = 1
n! = n * (n-1)!, kur n është pozitiv
```

(b) Funksioni i Fibonaccit:

```
Fib(0) = 1
Fib(1) = 1
Fib(n) = Fib(n-1) + Fib(n-2), kur n >= 2
```

Ky shembull është veçanërisht interesant, sepse është shumë më efikase ta llogarisim krejt tabelën përkthyese për një interval të vlerave të Fibonaccit dhe ta konsultojmë tabelën vetëm një herë, se sa ta llogarisim vetëm një vlerë të Fibonaccit me metodën e përkufizuar në mënyrë rekursive. Pse ndodh kështu?

Përgjigje:

(a)

```
int[] faktorieli = new int[20];
faktorieli[0] = 1;
for ( int n = 1; n != faktorieli.length; n = n + 1 )
    { faktorieli[n] = n * faktorieli[n-1]; }
```

(b)

```
int[] fib = new int[20];
fib[0] = 1;
fib[1] = 1;
for ( int n = 2; n != fib.length; n = n + 1 )
    { fib[n] = fib[n – 1] + fib[n – 2]; }
```

Siç kemi vërejtur në zgjidhjen e ushtrimit të ngjashëm në Kapitullin e mëparshëm, një përkufizim rekursiv i funksionit të Fibonnaci-t kryen shumë thirrje rekursive me të njëjtën vlerë të parametrit aktual. Llogaritja ciklike (me urdhërin for) e llogarit përgjigjen për secilën vlerë saktësisht një herë.

(8.4 Struktura e Brendshme e Vargjeve Njëdimensionale)

8.5 Vargjet e Objekteve

1. Qe një klasë, `Numerues`, që mund ta mbajë n'mend një numërim:

```
public class Numerues
{ private int n;
  public Numerues(int v) { n = v; }
  public void rrite() { n = n + 1; }
  public int merreNumrin() { return n; }
}
```

Të themi se e ndërrojmë deklarimin e votave në aplikacionin për numërimin e votave `NumeruesIVotave` në Figurën 1 ashtu që të kemi:

```
Numerues[] votat = new Numerues[num_kandidateve];
```

Rregulloje aplikacionin për numërimin e votave që të punojë me këtë varg.

Përgjigje:

Urdhërat e ndërruar janë shënuar me (*):

```
import javax.swing.*;
public class NumeruesIVotave4
{ public static void main(String[] args)
  { int num_kandidatëve = 4;
    Numerues[] votat = new Numerues[num_kandidatëve]; //(*) i mban votat
    // (*) krijo objektet Numerues dhe vendosi në varg:
    for ( int i = 0; i != num_kandidatëve; i = i + 1 ) // (*)
        { votat[i] = new Numerues(0); }
    boolean duke_perpunuar = true;
    while ( duke_perpunuar )
          { int v = new Integer(JOptionPane.showInputDialog
                       ("Vota për (0,1,2,3): ")).intValue();
            if ( v >= 0  &&  v < votat.length )
                 { votat[v].rrite(); }         // (*)
            else { duke_perpunuar = false; }
          }
    for ( int i = 0; i != num_kandidatëve; i = i + 1 )
        { System.out.println("Kandidati" + i + " i ka "
                             + votat[i].merreNumrin() + " vota"); //(*)
        }
  }
}
```

2. Qe një klasë që e modelon bankën e përshkruar në këtë paragraf (në libër):

```
/** Bankë e modelon nje grup te llogarive te bankes */
public class Bankë
{ LlogariEBankes[] banka; // vargu qe e mban llogarinë
  int llogaria_max; // numri maksimal i llogarise
  /** Konstruktori Banka e nis (inicializon) banken
    * @param sa - numri maksimal i llogarive te bankes */
  public Bankë(int sa)
  { llogaria_max = sa;
    banka = new LlogariEBankes[sa];
  }

  /** shtoLlogariTëRe e shton një llogari të re në bankë
    * @param numri_id - numri identifikues i llogarisë;
    *                   duhet të jetë në intervalin:
    *                   0..numri_maksimal_i_llogarisë-1
    * @param llogaria - objekti i ri i llogarisë së bankës
    * return (kthen) false nëse numri_id është i parregullt */
  public boolean shtoLlogariTëRe(int numri_id,
                                 LlogariEBankes llogaria)
  { boolean rezultati = false;
    if ( numri_id >= 0 && numri_id < llogaria_max
         && banka[numri_id] == null ) // a është numri_id numër i rregullt?
       { banka[numri_id] = llogaria;
         rezultati = true;
       }
    return rezultati;
  }
```

```
  /** gjejeLlogarinë e gjen nje llogari te bankës
    * @param numri_id - numri identifikues i llogarisë së dëshiruar
    * @return (kthen) llogarinë e bankës me identifikuesin numri_id;
    * nese nuk ka llogari me numri_id, return (kthen) null */
  public LlogariEBankes gjejeLlogarinë(int numri_id)
  { //...
  }
  /** fshijeLlogarinë e fshin një llogari te bankës
    * @param numri_id - numri identifikues i llogarisë që do të fshihet
    * @return (kthen) true, nëse fshirja ishte e suksesshme;
    * return (kthen) false, nëse asnjë llogari nuk e ka numrin identifikues */
  public boolean fshijeLlogarinë(int numri_id)
  { //...
  }
}
```

Do të mund ta përdornim klasën si vijon:

```
Bankë b = new Bankë(500);
b.shtoLlogariTëRe(155, new LlogariEBankes(100));
...
LlogariEBankes a = b.gjejeLlogarinë(155);
... a.depozito(200) ...
```

Shkruaji dy metodat që mungojnë për klasën `Bankë`.

Përgjigje:

```
  public LlogariEBankes gjejeLlogarinë(int numri_id)
  { LlogariEBankes rezultati = null;
    if ( numri_id >= 0  &&  numri_id < llogaria_max )
       { rezultati = banka[numri_id]; }
    return rezultati;
  }

  public boolean fshijeLlogarinë(int numri_id)
  { boolean rezultati = false;
    if ( numri_id >= 0  &&  numri_id < llogaria_max )
       { if ( banka[numri_id] != null )
         { banka[numri_id] = null;
           rezultati = true;
         }
       }
    return rezultati;
  }
```

3. Përdore klasën `Bankë` në shembullin e mëparshëm për ta shkruar një aplikacion që e lejon përdoruesin të konstruktojë llogari të reja të bankës, të bëjë depozitime dhe tërheqje, dhe të afishojë balanse.

8.6 Case Study: Bazat e të Dhënave

Qe klasët `BazëETëDhënave`, `Rrokje` dhe `Çelës`:

```
/** BazeETeDhenave e implementon nje baze te rrokjeve */
public class BazeETeDhenave
{ private Rrokje[] baza;    // bashkesia e rrokjeve
  private int NUK_U_GJET = -1;  // numer i plote qe perdoret
                                // kur nuk gjendet nje rrokje

  /** Konstruktori  BazeETeDhenave  e inicializon bazen
    * @param madhesia_fillestare - madhesia e bazes */
  public BazeETeDhenave(int madhesia_fillestare)
  { if ( madhesia_fillestare > 0 )
        { baza = new Rrokje[madhesia_fillestare]; }
    else { baza = new Rrokje[1]; }
  }

  /** gjejeVendin  eshte metode ndihmese qe e kerkon neper baze nje rrokje
    * qe e ka celesin k.  Nese gjendet, kthehet indeksi i rrokjes,
    * perndryshe kthehet NUK_U_GJET. */
  private int gjejeVendin(Celes c)
  { int rezultati = NUK_U_GJET;
    boolean u_gjet = false;
    int i = 0;
    while ( !u_gjet && i != baza.length )
          { if ( baza[i] != null && baza[i].merreCelesin().barazMe(c) )
                    { u_gjet = true;
                      rezultati = i;
                    }
               else { i = i + 1; }
          }
    return rezultati;
  }

  /** gjeje  e gjen nje rrokje ne baze, sipas celesit
    * @param celesi - celesi i rrokjes se deshiruar
    * @return (kthen) rrokjen e deshiruar (adresen e saj) */
  public Rrokje gjeje(Celes celesi)
  { Rrokje pergjigjja = null;
    int indeksi = gjejeVendin(celesi);
    if ( indeksi != NUK_U_GJET )
       { pergjigjja = baza[indeksi]; }
    return pergjigjja;
  }

  // vazhdon ne faqen tjeter
```

```
  // vazhdon nga faqja e meparshme

  /** fute  e fut nje rrokje te re ne baze.
    * @param rr - rrokja
    * @return (kthen) true, nese shtohet rrokja; return (kthen) false
    *  nese rrokja nuk shtohet sepse nje rrokje tjeter me celes te njejte
    *  tashme eshte ne baze */
  public boolean fute(Rrokje rr)
  { boolean suksesi = false;
    if ( gjejeVendin(rr.merreCelesin()) == NUK_U_GJET ) // rr nuk eshte ne baze?
       { // gjeje nje element te zbrazet ne baze, per ta futur rr-ne:
         boolean u_gjet_vend = false;
         int i = 0;
         while ( !u_gjet_vend && i != baza.length )
               // deri tash, baza[0]..baza[i-1] jane te zena
                  { if ( baza[i] == null ) // a eshte ky element i zbrazet ?
                              { u_gjet_vend = true; }
                     else { i = i + 1; }
                  }
         if ( u_gjet_vend )
               { baza[i] = rr; }
         else { // vargu eshte plot!
                    // Prandaj, krijoje nje tjeter qe mban me shume rrokje:
                    Rrokje[] perk = new Rrokje[baza.length * 2]; // i perkohshem
                    for ( int j = 0; j != baza.length; j = j + 1 )
                        { perk[j] = baza[j]; }
                    perk[baza.length] = rr;//fute rr ne elementin e pare te lire
                    baza = perk;
                }
         suksesi = true;
       }
    return suksesi;
  }

  /** fshije  e heq nje rrokje ne baze sipas celesit
    * @param celesi - celesi i rrokjet (identifikimi)
    * @return (kthen) true, nese rrokja gjendet dhe fshihet; perndryshe false */
  public boolean fshije(Celes celesi)
  { boolean rezultati = false;
    int indeksi = gjejeVendin(celesi);
    if ( indeksi != NUK_U_GJET )
       { baza[indeksi] = null;
         rezultati = true;
       }
    return rezultati;
  }
}
```

```
/** Rrokje e modelon nje llogari te bankes me nje celes te identifikimit */
public class Rrokje
{ private int balansi;      // balansi i llogarise
  private Celes celesi;       // celesi identifikues

  /** Konstruktori Rrokje e inicializon llogarine
    * @param sasia_fillestare - balansi fillestar i llogarise, numer jonegativ
    * @param id - celesi identifikues i llogarise */
  public Rrokje(int sasia_fillestare, Celes id)
  { balansi = sasia_fillestare;
    celesi = id;
  }

  /** depozito shton para ne llogari.
    * @param sasia - sasia e parave qe do te shtohen, numer i plote jonegativ */
  public void depozito(int sasia)
  { balansi = balansi + sasia; }

  /** merreBalansin e tregon balansin momental te llogarise
    * @return balansin */
  public int merreBalansin() { return balansi; }

  /** merreCelesin e kthen celesin e llogarise
    * @return (kthen) celesin */
  public Celes merreCelesin() { return celesi; }
}

-----------------------------------------------

/** Celes e modelon nje celes ne trajte te numrit te plote */
public class Celes
{ private int c;   // celesi - numer i plote

  /** Konstruktori Celes e konstrukton celesin
    * @param n - numri i plote qe e perkufizon celesin ne menyre unike */
  public Celes(int n) { c = n; }

  /** barazMe  e krahason se a eshte i barabarte ky Celes me nje tjeter
    * @param tjetri - celesi tjeter
    * @return (kthen) true, nese ky celes eshte i barabarte me c-ne,
    *  perndryshe false */
  public boolean barazMe(Celes tjetri)
  { return ( c == tjetri.merreNumrin() ); }

  /** merreNumrin e kthen vleren e plote qe e mban ky celes */
  public int merreNumrin() { return c; }
}
```

1. Shkruaje një aplikacion që e përdor klasën `BazeETeDhenave` dhe klasët `Rrokje` dhe `Celes` për t'iu ndihmuar përdoruesve që të konstruktojnë llogari të reja të bankës dhe të bëjnë depozitime në to.

Përgjigje:

Ndërrimet (në klasën `BazeETeDhenave`) janë të mërzitshme por të thjeshta:

```
/** Të gjitha paraqitjet e  Rrokje  zëvendësohen me  LlogariEBankes1
  * Të gjitha paraqitjet e  Celes  zëvendësohen me  int
  * Te gjejeVendin zëvendëso  barazMe  me  == */

public class BazeETeDhenave1
{ private LlogariEBankes1[] baza;   // koleksion i rrokjeve
   ...

  /** pozitaE e kthen indeksin ne baze ku paraqitet
    * nje rrokje me  k */
  private int pozitaE(int k)
  { int rezultati = NUK_U_GJET;
     ...
          if ( baza[i] != null  &&  baza[i].merreCelesin() == k )
             { ... }
     ...
  }
}
```

Qe një program testues:

```
public class Test1
{
  public static void main(String[] args)
  { BazeETeDhenave1 b = new BazeETeDhenave1(5);

    LlogariEBankes1 r1 = new LlogariEBankes1(100, 1);
    System.out.println("Insertimi 1 eshte  " + (b.fute(r1) ? "mire" : "keq");
    LlogariEBankes1 r2 = new LlogariEBankes1(200, 2);
    System.out.println("Insertimi 2 eshte " + (b.fute(r2) ? "mire" : "keq");

    System.out.println("Fshirja e 2 eshte " + (b.fshije(2) ? "mire" : "keq");

    LlogariEBankes1 r = b.gjeje(1);
    System.out.println( "Gjetja e 1 eshte "
       + ( b.gjeje( r1.merreCelesin() ) ? "mire" : "keq" ) );
  }
}
```

8.7 Case Study: Lojë Kartash

Qe kodi i klasëve `Karte` dhe `PakoEKartave`:

```
// Fajli Karte.java

/** Karte  e modelon nje karte per lojera */
public class Karte
{ // perkufizimet qe dikush mund t'i perdore per ta pershkruar vleren nje karte:
  public static final String GJETHE = "gjethe";
  public static final String ZEMER = "zemer";
  public static final String RRUSH = "rrush";
  public static final String ROMB = "romb";

  public static final int PIKE = 1;
  public static final int XHANDAR = 11;
  public static final int QIKE = 12;
  public static final int MBRET = 13;

  public static final int MADHESIA_E_NJE_NGJYRE = 13; // sa karta ne nje ngjyre

  // Keto jane atributet (vetite) e kartes:
  private String ngjyra;
  private int vlera;

  /** Konstruktori Karte e cakton ngjyren dhe vleren.
    * @param n - ngjyra
    * @param v - vlera */
  public Karte(String n, int v)
  { ngjyra = n;
    vlera = v;
  }

  /** merreNgjyren e kthen ngjyren e kartes */
  public String merreNgjyren()
  { return ngjyra; }

  /** merreVleren e kthen vleren e kartes */
  public int merreVleren()
  { return vlera; }
}
```

```
// Fajli PakoEKartave.java

/** PakoEKartave e modelon nje pako te kartave */
public class PakoEKartave
{
  private int nr_kartave; // sa karta mbeten ne pako
  private Karte[] pakoja = new Karte[4 * Karte.MADHESIA_E_NJE_NGJYRE];
    // invariantë: elementet pakoja[0]..pakoja[nr_kartave-1] i mbajne kartat */

  /** Konstruktori  PakoEKartave  e krijon nje pako te re te kartave
    * me te gjitha kartat e veta */
  public PakoEKartave()
  { krijoNgjyren(Karte.GJETHE);
    krijoNgjyren(Karte.ZEMER);
    krijoNgjyren(Karte.RRUSH);
    krijoNgjyren(Karte.ROMB);
  }

  /** karteERe  e merr nje karte te re nga pakoja.
    * @return (kthen) nje karte qe s'eshte perdore me heret,
    * ose kthen null, nese s'kane mbete me karta */
  public Karte karteERe()
  { Karte karta_tjeter = null;
    if (nr_kartave == 0)
         { System.out.println("Gabim te PakoEKartave: nuk ka me karta"); }
    else { int indeksi = (int)(Math.random() * nr_kartave); //zgjedhe rastesisht
           karta_tjeter = pakoja[indeksi];
           // meqe karta u nxjerr nga pakoja, zhvendosi kartat tjera
           // per ta mbushe zbraztesine:
           for ( int i = indeksi+1; i != nr_kartave; i = i + 1 )
               // deri tash, kartat nga indeksi+1 deri ne i-1
               // jane zhvendosur majtas ne varg per nje pozite
               { pakoja[i - 1] = pakoja[i]; }
           nr_kartave = nr_kartave - 1;
         }
    return karta_tjeter;
  }

  /** kaKarta  tregon se a ka pakoja karta per te dhene.
    * @return (kthen) se a eshte pakoja joboshe */
  public boolean kaKarta()
  { return (nr_kartave > 0); }

  /** krijoNgjyren e krijon nje ngjyre kartash per nje pako te re te kartave */
  private void krijoNgjyren(String cila_ngjyre)
  { for ( int i = 1; i <= Karte.MADHESIA_E_NJE_NGJYRE; i = i + 1 )
       { pakoja[nr_kartave] = new Karte(cila_ngjyre, i);
         nr_kartave = nr_kartave + 1;
       }
  }
}
```

1. Shkruaje një aplikacion testues që e krijon një kartë të re, qikë-zemër, dhe pastaj e pyet kartën se çfarë ngjyre dhe vlere ka. Programi i afishon përgjigjet që i pranon në dritaren komanduese. A afishon programi yt `Qikë` apo `12`? Çfarë zgjidhje propozon që të afishohet e para (`Qikë`)?

Përgjigje:

```
public class Test1
{
  public static void main(String[] args)
  {
    Card m = new Card(Card.HEARTS, Card.QUEEN);
    String ngjyra = q.suitOf();
    int vlera = q.countOf();
    System.out.println("Karta është " + vlera + " - " + ngjyra);
  }
}
```

Natyrisht, ky shembull afishon: `Karta është 12 - zemer`, që është joelegante. Ndoshta zgjidhja më e mirë është një metodë përkthyese brenda klasës `Karte`; shtoja këtë metodë asaj:

```
  /** neFjale e afishon ngjyren dhe vleren e kartes me fjale
    * @return paraqitjen me fjale (si String) te kartes */
  public String neFjale
  { String s = " me " + ngjyra;
    if ( numri == PIKE ) { s = "pike" + s; }
    else if ( numri == XHANDAR ) { s = "xhandar" + s; }
    else if ( numri == QIKE ) { s = "qike" + s; }
    else if ( numri == MBRET ) { s = "mbret" + s; }
    else    { s = vlera + s; }
    return s;
  }
}
```

Pastaj, në programin testues mund të shkruajmë:
`System.out.println("Karta është " + q.neFjale()).`

2. Shkruaje një aplikacion që e krijon një pako të re të kartave dhe kërkon nga pakoja që t'i shpërndajë ('shkepë') 53 karta. (Kjo është një kartë më shumë se që i mban pakoja!) Derisa pakoja i kthen kartat një nga një, afisho në dritaren komanduese vlerën dhe ngjyrën e secilës kartë.

Përgjigje:

```
public class Test2
{
  public static void main(String[] args)
  { PakoEKartave pakoja = new PakoEKartave();
    for ( int i = 1;  i <= 53;  i = i+1 )
        { Karte k = pakoja.karteERe();
          System.out.println(k.merreVleren() + " me " + k.merreNgjyren());
        }
  }
}
```

3. Shkruaje një aplikacion që e lejon përdoruesin t'i kërkojë kartat një nga një, derisa përdoruesi të thotë "mjaft".

Përgjigje:

```
import javax.swing.*;
public class Test3
{
  public static void main(String[] args)
  { PakoEKartave pakoja = new PakoEKartave();
    boolean duke_perpunuar = true;
    while ( duke_perpunuar )
        { String kerkesa =
            JOptionPane.showInputDialog("Edhe një kartë? (Apo, mjaft):");
          if ( kerkesa.equals("mjaft") )
               { duke_perpunuar = false; }
          else { if ( !pakoja.kaKarta() )
                      { System.out.println("Me vjen keq – pakoja boshe"); }
                 else { Karte k = pakoja.karteERe();
                        System.out.println(k.merreVleren()
                                           + " me " + k.merreNgjyren());
                      }
               }
        }
  }
}
```

8.8 Vargjet Dy-dimensionale

1. Krijoje një varg dy-dimensional të numrave të plotë, `m`, me 12 rreshta e 14 shtylla dhe inicializoje ashtu që çdo `m[i][j]` e mban vlerën `i * j` .

Përgjigje.

```
int[][] m = new int[12][14];
for ( int i = 0; i != 12; i = i + 1 )
    { for ( int j = 0; j != 14; j = j + 1 )
          { m[i][j] = i * j; }
    }
```

2. Krijoje një varg dy-dimensional të Stringjeve, `m`, me 7 rreshta e 15 shtylla dhe inicializoje ashtu që çdo `m[i][j]` e mban stringun `"Elementi " + i + " " + j`.

Përgjigje.

```
String[][] m = new String[7][15];
for ( int i = 0; i != 7; i = i + 1 )
    { for ( int j = 0; j != 15; j = j + 1 )
          { m[i][j] = "Elementi " + i + " " + j; }
```

3. Është dhënë vargu, `int[][] r = new int[4][4]`, dhe ky cikël-for që e afishon përmbajtjen e r-së:

```
for ( int i = 0; i != 4; i = i + 1 )
    { for ( int j = 0; j != 4; j = j + 1 )
          { System.out.print( r[i][j] + " " ); }
      System.out.println();
    }
```

shkruaji ciklet-for që e inicializojnë `r`-në ashtu që të afishohen modelet vijuese:

(a)
```
1 0 0 0
1 2 0 0
1 2 3 0
1 2 3 4
```

Përgjigje.

```
for ( int i = 0; i != 4; i = i + 1 )
    { for ( int j = 0; j != 4; j = j + 1 )
          { if ( j <= i )
                { r[i][j] = j + i; }
            else { r[i][j] = 0; }
          }
    }
```

(b)
```
1 2 3 4
0 3 4 5
0 0 5 6
0 0 0 7
```

Përgjigje.

```
for ( int i = 0; i != 4; i = i + 1 )
    { for ( int j = 0; j != 4; j = j + 1 )
          { if ( j >= i )
                { r[i][j] = i + j + 1; }
            else { r[i][j] = 0; }
    }
```

(c)
```
1 0 1 0
0 1 0 1
1 0 1 0
0 1 0 1
```

Përgjigje.

```
for ( int i = 0; i != 4; i = i + 1 )
    { for ( int j = 0; j != 4; j = j + 1 )
          { r[i][j] = (i + j + 1) % 2; }
    }
```

(8.9 Struktura e Brendshme e Vargjeve Dy-dimensionale)

8.10 Case Study: Loja e Rebusit Rrëshqitës

Qe klasët e rebusit:

```
// Fajli PjeseERebusit.java

/** PjeseERebusit  e perkufizon nje pjese (katror) te rebusit rreshqites */
public class PjeseERebusit
{ private int vlera; // vlera e shkruar ne siperfaqen e pjeses (numri)

  /** Konstruktori PjeseERebusit e krijon nje pjese
    * @param v - vlera qe paraqitet ne siperfaqen e pjeses */
  public PjeseERebusit(int v)
  { vlera = v; }

  /** vleraE  e kthen vleren e pjeses */
  public int vleraE()
  { return vlera; }
}

-------------------------------

// Fajli TabeleERebusit.java

/** TabeleERebusit e modelon nje rebus rreshqites. */
public class TabeleERebusit
{ private int madhesia;   // madhesia e tabeles
  private PjeseERebusit[][] tabela; // vargu qe i mban pjeset

  // nje pozite ne tabele duhet te jete vend i zbrazet (bosh):
  private int bosh_rreshti;
  private int bosh_shtylla;

  // invariantë e reprezentimit: tabela[bosh_rreshti][bosh_shtylla] == null

  /** Konstruktori TabeleERebusit e konstrukton rebusin fillestar, që i ka
    *   pjesët të vendosura në radhitje rënëse numerike.
    * @param s - madhesia e rebusit, numër i plotë pozitiv (p.sh., s==4 dmth.
    *  se rebusi është 4 x 4 dhe do t'i ketë pjesët të numëruara 15,14,...,1)*/
  public TabeleERebusit(int s)
  { madhesia = s;
    tabela = new PjeseERebusit[madhesia][madhesia];
    // krijo pjeset vec e vec dhe vendosi ne tabele ne radhitje te kundert:
    for ( int num = 1; num != madhesia*madhesia; num = num + 1)
        { PjeseERebusit p = new PjeseERebusit(num);
          int rr = num / madhesia; // rreshti
          int sh = num % madhesia; // shtylla
          // cakto  p  ne "poziten inverse" ne tabele:
          tabela[madhesia-1-rr][madhesia-1-sh] = p;
        }
  // vazhdon ne faqen tjeter...
```

```
// vazhdon nga faqja e mëparshme

  // mbaje n'mend vendin ne tabele kur fillimisht nuk ka pjese:
  bosh_rreshti = madhesia - 1;
  bosh_shtylla = madhesia - 1;
}

/** permbajtja  e kthen gjendjen momentale te rebusit
  * @return (kthen) matrice qe i permban adresat e pjeseve */
public PjeseERebusit[][] permbajtja()
{ PjeseERebusit[][] pergjigjja = new PjeseERebusit[madhesia][madhesia];
  for ( int i = 0; i != madhesia; i = i + 1 )
      { for ( int j = 0; j != madhesia; j = j + 1 )
            { pergjigjja[i][j] = tabela[i][j]; }
      }
  return pergjigjja;
}

/** leviz  e leviz nje pjese ne hapesiren e zbrazet, duke u siguruar qe eshte
  *  levizje e lejueshme.
  * @param v - vlera ne faqen e pjeses qe dikush don ta levize
  * @return (kthen) true, nese pjesa e shenuar me  v  eshte levizur
  *    ne vendin bosh;
  *  return(kthen) false nese pjesa nuk mund te levizet ne vendin bosh */
public boolean leviz(int v)
{ int NUK_U_GJET = -1;
  int rreshti = NUK_U_GJET;  // rreshti dhe shtylla do ta mbajne n'mend
  int shtylla = NUK_U_GJET;  // ku jeton pjesa  v
  // mundohu ta gjesh  w  ngjitas me vendin bosh ne tabele
  if ( gjendet(v, bosh_rreshti - 1, bosh_shtylla) )      // lart
     { rreshti = bosh_rreshti - 1;
       shtylla = bosh_shtylla;
     }
  else if ( gjendet(v, bosh_rreshti + 1, bosh_shtylla) ) // poshte
     { rreshti = bosh_rreshti + 1;
       shtylla = bosh_shtylla;
     }
  else if ( gjendet(v, bosh_rreshti, bosh_shtylla - 1) ) // majtas
     { rreshti = bosh_rreshti;
       shtylla = bosh_shtylla - 1;
     }
  else if ( gjendet(v, bosh_rreshti, bosh_shtylla + 1) ) // djathtas
     { rreshti = bosh_rreshti;
       shtylla = bosh_shtylla + 1;
     }

   if ( rreshti != NUK_U_GJET ) // dmth. u gjet
      { // levize pjesen ne vendin e zbrazet:
        tabela[bosh_rreshti][bosh_shtylla] = tabela[rreshti][shtylla];
        // shenoje vendin e ri te zbrazet ne tabele:
        bosh_rreshti = rreshti;
        bosh_shtylla = shtylla;
        tabela[bosh_rreshti][bosh_shtylla] = null;
      }
  return  rreshti != NUK_U_GJET;
}

// vazhdon ne faqen tjeter...
```

```
  // vazhdon nga faqja e mëparshme
  /** gjendet pergjigjet se a e ka numrin  v  pjesa ne poziten  rr, sh
    *   dmth a gjendet numri v ne poziten (rr,sh) */
  private boolean gjendet(int v, int rr, int sh)
  { boolean pergjigjja = false;
    if ( rr >= 0 && rr < madhesia && sh >= 0 && sh < madhesia )
       { pergjigjja = ( tabela[rr][sh].vleraE() == v ); }
    return pergjigjja;
  }
}
```

```
// Fajli KontrolluesIRebusit.java

import javax.swing.*;
/** KontrolluesIRebusit i kontrollon levizjet e nje rebusi rreshqites */
public class KontrolluesIRebusit
{ private TabeleERebusit tabela;       // modeli
  private VizatuesIRebusit vizatuesi;  // pamja dalese

  /** Konstruktori KontrolluesIRebusit e nis kontrolluesin
    * @param t - modeli, tabela e rebusit
    * @param v - pamja dalese (vizatuesi) */
  public KontrolluesIRebusit(TabeleERebusit t, VizatuesIRebusit v)
  { tabela = t;
    vizatuesi = v;
  }

  /** luaj e lejon perdoruesin qe te luaje ne rebus */
  public void luaj()
  { while ( true )
          { vizatuesi.paraqiteRebusin();
            int i = new Integer
                     (JOptionPane.showInputDialog("Levizja juaj:")).intValue();
            boolean rezultat_i_mire = tabela.leviz(i);
            if ( !rezultat_i_mire )
               { vizatuesi.afishoGabimin(
                     "Levizje e parregullt - rebusi nuk nderron"); }
          }
  }
}
```

```
// Fajli RebusRreshqites.java

/** RebusRreshqites  e krijon nje rebus rreshqites 4 x 4.
  * Programi nuk kryhet kurre. */
public class RebusRreshqites
{ public static void main(String[] args)
  { int madhesia = 4; // rebus rreshqites 4 x 4
    TabeleERebusit tabela = new TabeleERebusit(madhesia);
    VizatuesIRebusit vizatuesi = new VizatuesIRebusit(tabela, madhesia);
    KontrolluesIRebusit kontrolluesi =
      new KontrolluesIRebusit(tabela, vizatuesi);
    kontrolluesi.luaj();
  }
}
```

```
// Fajli VizatuesIRebusit.java

import java.awt.*; import javax.swing.*;
/** VizatuesIRebusit  e paraqet permbajtjen e nje rebusi rreshqites */
public class VizatuesIRebusit extends JPanel
{ private TabeleERebusit tabela; // tabela qe paraqitet
  private int madhesia;               // madhesia e tabeles
  private int madhesia_e_pjeses = 30;
                // madhesia ne piksela e nje pjese(katrori)
  private int gjeresia_e_panelit; // gjeresia dhe lartesia e panelit
  private int lartesia_e_panelit;

  /** Konstruktori VizatuesIRebusit e nderton dritaren grafike.
    * @param t - rebusi rreshqites qe paraqitet
    * @param m - madhesia e rebusit, p.sh., 5 dmth 4 x 4 */
  public VizatuesIRebusit(TabeleERebusit t, int m)
  { tabela = t;
    madhesia = m;
    gjeresia_e_panelit = madhesia_e_pjeses * madhesia + 100;
    lartesia_e_panelit = madhesia_e_pjeses * madhesia + 100;
    JFrame korniza_ime = new JFrame();
    korniza_ime.getContentPane().add(this);
    korniza_ime.setTitle("Rebusi rreshqites");
    korniza_ime.setSize(gjeresia_e_panelit, lartesia_e_panelit);
    korniza_ime.setVisible(true);
  }

  /** vizatoPjesen  e vizaton pjesen  p  ne poziten i,j  ne dritare */
  private void vizatoPjesen(Graphics g, PjeseERebusit p, int i, int j)
  { int zhvendosja = madhesia_e_pjeses;
    int x_poz = zhvendosja + (madhesia_e_pjeses * j);
    int y_poz = zhvendosja + (madhesia_e_pjeses * i);
    if ( p != null )
         { g.setColor(Color.white);
           g.fillRect(x_poz, y_poz, madhesia_e_pjeses, madhesia_e_pjeses);
           g.setColor(Color.black);
           g.drawRect(x_poz, y_poz, madhesia_e_pjeses, madhesia_e_pjeses);
           g.drawString(p.vleraE() + "", x_poz + 10, y_poz + 20);
         }
    else { g.setColor(Color.black);
           g.fillRect(x_poz, y_poz, madhesia_e_pjeses, madhesia_e_pjeses);
         }
  }

  /** paintComponent  e paraqet rebusin ne kornize. */
  public void paintComponent(Graphics g)
  { g.setColor(Color.yellow);
    g.fillRect(0, 0, gjeresia_e_panelit, lartesia_e_panelit);
    PjeseERebusit[][] r = tabela.permbajtja();
    for ( int i = 0; i != madhesia; i = i + 1 )
        { for ( int j = 0; j != madhesia; j = j + 1 )
              { vizatoPjesen(g, r[i][j], i, j); }
        }
  }

  // vazhdon ne faqen tjeter...
```

```
  // vazhdon nga faqja e mëparshme

  /** paraqiteRebusin  e paraqet gjendjen momentale te rebusit rreshqites. */
  public void paraqiteRebusin()
  { this.repaint(); }

  /** afishoGabimin  e paraqet nje porosi te gabimit.
    * @param s - porosia e gabimit */
  public void afishoGabimin(String s)
  { JOptionPane.showMessageDialog(null, "Gabim te VizatuesIRebusit: " + s ); }
}
```

1. Testoje klasën `TabelëERebusit` duke e krijuar një objekt, `tabela`, prej saj dhe duke e thirrur menjëherë metodën e vet, `përmbajtja`. Afishoje `tabelën` me këto cikle:

```
PjeseERebusit[][] r = tabela.permbajtja();
for ( int i = 0; i != r.length; i = i + 1 )
    { for ( int j = 0; j != r[i].length; j = j + 1 )
          { if (r[i][j] == null )
                { System.out.print("X "); }
            else { System.out.print( r[i][j].vleraE() + " "); }
          }
      System.out.println();
    }
```

Pastaj, përdore metodën, `lëviz`, të objektit për të kërkuar nga fusha t'i lëvizë disa numra. Afishoje tabelën që del nga secila lëvizje.

Përgjigjje.

```
public class Test1
{ public static void main(String[] args)
  { TabeleERebusit tabela = new TabeleERebusit(4);

    PjeseERebusit[][] r = tabela.permbajtja();
    afishoPermbajtjen(r);

    tabela.leviz(1);
    afishoPermbajtjen(tabela.permbajtja());

    tabela.leviz(4);
    afishoPermbajtjen(tabela.permbajtja());

    tabela.leviz(5);
    afishoPermbajtjen(tabela.permbajtja());
  }

  // vazhdon ne faqen tjeter...
```

```
  /** afishoPermbajtjen i paraqet pjeset e rebusit ne tabelen r */
  private static void afishoPermbajtjen(PjeseERebusit[][] r)
  { for ( int i = 0; i != r.length; i = i+1 )
        { for (int j = 0; j != r[i].length; j = j+1 )
              { if ( r[i][j] == null )
                     { System.out.print("\tX"); }  // '\t' eshte Tab
                else { System.out.print( "\t" + r[i][j].vleraE() ); }
              }
          System.out.println();
        }
    System.out.println();
  }
}
```

2. Përdori ciklet-for nga ushtrimi i kaluar në një implementim alternativ (tjetër) të klasës `VizatuesIRebusit` që e afishon rebusin në dritaren komanduese (konsolë) (e jo në dritare grafike).

Përgjigje.

```
public class VizatuesIRebusit2
{ private TabeleERebusit tabela;      // tabela qe do te afishohet
  private int madhesia;               // madhesia e tabeles

  /** Konstruktori VizatuesIRebusit2 e nis vizatuesin.
    * @param t - rebusi rreshqites qe afishohet
    * @param m - madhesia e rebusit, p.sh. 4 dmth. 4 x 4 */
  public VizatuesIRebusit2(TabeleERebusit t, int m)
  { tabela = t;
    madhesia = m;
  }

  /** afishoRebusin e afishon gjendjen momentale te rebusit */
  public void afishoRebusin()
  { PjeseERebusit[][] r = tabela.permbajtja(); // merre permb. e tab.
    for ( int i = 0; i != r.length; i = i+1 )
        { for (int j = 0; j != r[i].length; j = j+1 )
              { if ( r[i][j] == null )
                     { System.out.print("\tX"); }  // '\t' eshte Tab
                else { System.out.print( "\t" + r[i][j].vleraE() ); }
              }
          System.out.println();
        }
    System.out.println();
  }

  /** afishoGabimin e afishon nje porosi te gabimit.
    * @param s - porosia e gabimit */
  public void afishoGabimin(String s)
  { System.out.println("Gabim te VizatuesIRebusit2: " + s); }
}
```

• •

* * * * * artridesign * * * * *

• •