Dokumentacija za Izpit

Rok Kos

verzija: 30. januar 2017

Kazalo

Slike

1 Input/Output

1.1 Input

```java
import java.util.Scanner;
Scanner in = new Scanner(System.in);
byte b = in.nextByte();
int i = in.nextInt();
long l = in.nextLong();
double d = in.nextDouble();
String s = in.next();  // Vrne naslednji string
String line = in.nextLine();  // Prebere celotno vrstico in skoci v novo

// Branje do konca inputa
while(in.hasNextInt()){
        int a = in.nextInt();
}
// Namesto hasNextInt bi lahko bilo tudi:
// -hasNext(), ne dela vedno, mislim da bere tudi \n new line znake, raje ne uporabljati
// -hasNextDouble()
// -hasNextLong
// -hasNextLine
```

1.2 Output

```java
System.out.println(dnevi + ". dan: " + prej + " -> " + d + " (prehodil " + p + ")");
System.out.print("Ne gre v naslednjo vrsico" + 5);

System.out.printf("%d. dan: %d -> %d (prehodil %d)%n", dnevi, prej, d, p);
System.out.format("%1\$+020.10f", Math.PI);  // Enako kot print pri println in print(ne gre v novo vrstico)
// "1 dolar kateri argument"
// + pomeni predznacen, 0 pomeni da naj bodo spredaj vodilne 0
// 20.10 pomeni 20 mest spredaj in na 10 decimalk
// FORMATER
// %d - int, long, byte, %f - double, float, %s -string, %n - newline
// + -> predznak, - -> levo poravnan, 010.5 -> vodilne nicle 10 mest z 5 decimlkami
```

2 Delo s števili

2.1 Parsing

```java
String s = "123456789";
int a = Integer.parseInt(s);
long l = Long.parseLong(s);
double d = Double.parseDouble(s);

// Dodatna moznost se spreminja iz razlicnih sistemov
int deset = Integer.parseInt("10101010", 2);  // Pretvori iz dvojiskega v desetisko
String trojisko = Integer.toString(15, 3);  // Pretvori iz desetiskega v trojisko

String besedilo = "Neka dolga poved v kateri mores dobit vsako besedo";
String[] besede = besedilo.split(" ");  // Lahko tudi po , . !, ali celo po regex
for (int i = 0; i < besede.length; ++i) {
        System.out.println(besede[i]);
}
```

2.2 Math knjiznica

```java
double pi = Math.PI;  // Math.E
double a = abs(a);  // Tudi za int, long in float
double max = Math.max(int, int)  // lahko tudi double, float in long
double min = Math.min(int, int)  // lahko tudi double, float in long

// Trigonometricne funkcije
double kot = asin(val) / Math.PI * 180;  // acos, atan vrne vrednost v PI radianih
double val = sin(kot * Math.PI / 180);  // cos, tan
// Mozna pretvorba tudi z toDegrees ali toRadians

// Hiperbolicne funkcije
double h = sinh(val);  // cosh, tanh

// Zaokrozevanje
```

```
15  double navzgor = Math.ceil(decimalka);
16  double navzdol = Math.flor(decimalka);
17  long zaokrozi = Math.round(decimalka);  // vrne celo stevilo(lahko tudi int), 2.5 bo 3, 2.4 bo 2
18
19  // Korenjenje
20  double kvadratni = Math.sqrt(koren);
21  double kubicni = Math.cbrt(koren);
22
23  // Eksponetna funkcija
24  double potenca = Math.pow(osnova, eksponent);
25  double eNaEks = Math.exp(naDecimalko);
26  double obratno = Math.log(naravni);  // lahko tudi z desetisko osnovo (log10(a))
27
28  // Random
29  double r = Math.random();  // vrne od 0.0 do 1.0
```

2.3 Int/Long class

```
1  int M = Integer.MAX_VALUE;  // MIN_VALUE
2  // Enako za byte, short, long, double
3
4  // Pretvorba velja tudi za long
5  String s = i.toString();
6  String b = i.toBinary();
7  String h = i.toHex();
8  String o = i.toOctal();
```

3 Delo z besedili

3.1 String class

```
1  char a = besedilo.charAt(index);
2  int l = besedilo.length();
3  String s = str1.concat(str2);  // Doda str2 nakoncu str1
4
5  // Primerjanje
6  boolean enaka = string1.equals(string2);  // NUJNO UPORABLJAJ TO ZA PRIMERJANJE
7  int pred = string1.compareTo(string2);  // Vrne neg. st ce je str1 pred str2 in poz. st. obratno, 0 ce sta enaka
8  // compareToIgnoreCase in equalsIgnoreCase je tudi na voljo
9
10 // Manipulacija stringov
11 String sub = str1.substring(zacetek, konec);
12 String[] s = str1.split(' ');  // Split po nekem znaku ali regex pravilu
13
14 String rep = str1.replace(kateriChar, zKaterimChar);  // zamenja vse pojavitve
15 String rep = str1.replaceAll(regex, sCim);  // zamnenja vse pojavitve, ki ustrezajo regex(lahko tudi normalen string
16 String rep = str1.replaceFirst(regex, sCim);  // zamnenja prvo pojavitev, ki ustreza regexu
17
18 String lower = str1.toLowerCase();
19 String upper = str1.toUpperCase();
20
21 boolean match = str1.matches(regex);
22
23 // Iskanje po stringu
24 int index = str1.indexOf(chr, fromIndex);  // Namesto chr lahko tudi String
25 int index = str1.lastIndexOf(chr, fromIndex);  // Namesto chr lahko tudi String
26
27 boolean seZacne = str1.startsWith(str2, fromIndex);
28 boolean seKonca = str1.startsWith(str2);
```

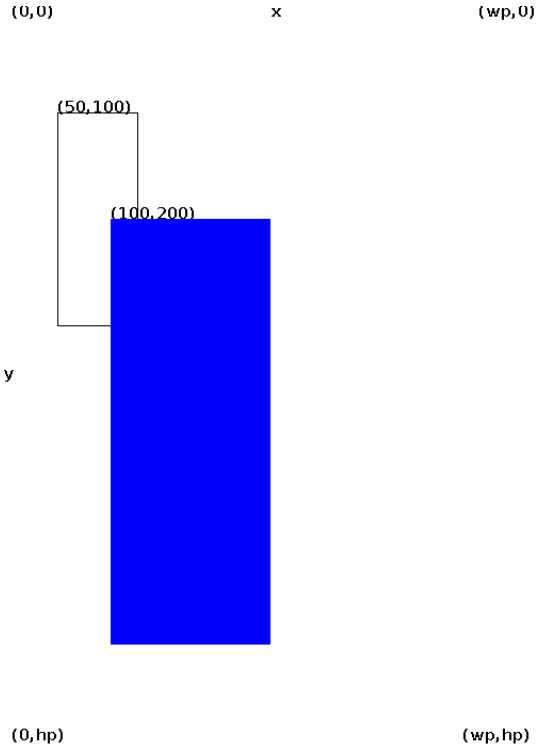
4 Delo z grafiko

4.1 Pravokotniki

Glej sliko 1

```
1  // Samo oznacevanje
2  g.setColor(Color.BLACK);
3  g.drawString("(0,0)",7,10);
```

```
4   g.drawString("(0,hp)",7,ri(hp-10));
5   g.drawString("(wp,0)",ri(wp)-40,10);
6   g.drawString("(wp,hp)",ri(wp)-50,ri(hp)-10);
7   g.drawString("x", ri(wp / 2), 10);
8   g.drawString("y", 0, ri(hp / 2));
9
10  // Risanje pravokotnikov
11  int levoZgorajX = 50;
12  int levoZgorajY = 100;
13  int dolzina = 75;
14  int visina = 200;
15  // Zaznamek za levi zgornji kot
16  g.drawString(String.format("(%d,%d)",levoZgorajX, levoZgorajY),levoZgorajX,levoZgorajY);
17  g.drawRect(levoZgorajX, levoZgorajY, dolzina, visina);
18  // Zaznamek za levi zgornji kot
19  g.drawString(String.format("(%d,%d)",levoZgorajX * 2, levoZgorajY * 2),levoZgorajX * 2,levoZgorajY * 2);
20  g.setColor(Color.BLUE);
21  g.fillRect(levoZgorajX * 2, levoZgorajY * 2, dolzina * 2, visina * 2);
```



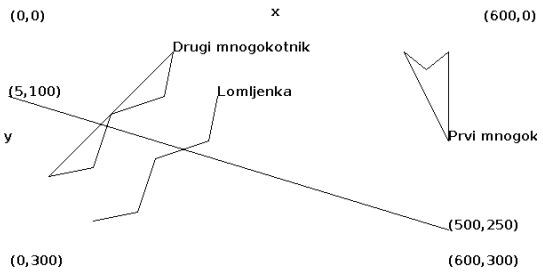
Slika 1: Pravokotniki

4.2 Crte

Glej sliko 2

```
1   // Risanje crt
2   int x1 = 5, y1 = 100, x2 = 500, y2 = 250;
3   g.drawString(String.format("(%d,%d)",x1, y1),x1,y1);
4   g.drawString(String.format("(%d,%d)",x2, y2),x2,y2);
5   g.drawLine(x1, y1, x2, y2);
6
7   // Risanje mnogokotnikov
8   // Desni na sliki
9   Polygon p = new Polygon();
10  p.addPoint(450, 50);
11  p.addPoint(475, 70);
12  p.addPoint(500, 50);
13  p.addPoint(500, 125);
14  p.addPoint(500, 150);
```

```
15  g.drawPolygon(p);
16
17  g.drawString("Prvi mnogokotnik",500, 150);
18
19
20  // Levi na sliki
21  int[] xKordinateOgljisc = new int[] {50, 100, 120, 180, 190};
22  int[] yKordinateOgljisc = new int[]  {190, 180, 120, 100, 50};
23
24  g.drawPolygon(xKordinateOgljisc, yKordinateOgljisc, xKordinateOgljisc.length);
25
26  g.drawString("Drugi mnogokotnik",190, 50);
27
28  for (int i = 0; i < xKordinateOgljisc.length; ++i) {
29          xKordinateOgljisc[i] += 50;
30          yKordinateOgljisc[i] += 50;
31  }
32  // Na enak nacin lahko narisemo lomljeno crto
33  g.drawPolyline(yKordinateOgljisc, xKordinateOgljisc, xKordinateOgljisc.length);
34
35  g.drawString("Lomljenka",240, 100);
```



Slika 2: Crte

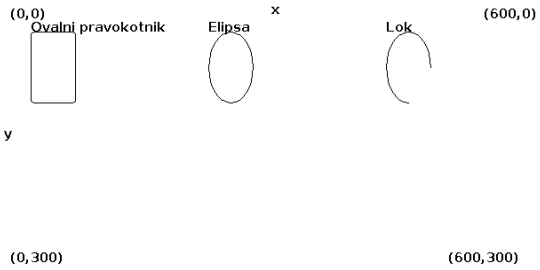
4.3 Ovalne oblke

Glej sliko 3

```
1   // Risanje ovalnih oblik
2   int x = 30, y = 30, sirina = 50, visina = 80;
3   int hPolmerKotnegaLoka = 10;  // horizontalno
4   int vPolmerKotnegaLoka = 5;  // vertikalno
5   // Ovalni pravokotnik
6   g.drawRoundRect(x, y, sirina, visina, hPolmerKotnegaLoka, vPolmerKotnegaLoka);
7   g.drawString("Ovalni pravokotnik",x, y);
8
9   x += sirina * 4;
10
11  // Elipsa
12  g.drawOval (x, y, sirina, visina);
13  g.drawString("Elipsa",x, y);
14
15  x += sirina * 4;
16
17  int zacetniKot = 0;
18  int kot = 270;
19  g.drawArc(x,y, sirina, visina, zacetniKot, kot);
20  g.drawString("Lok",x, y);
```

4.4 Ostalo

```
1   // Ostale uporabne funkcije
2   Color color = g.getColor();  // Dobi trenurno barvo
3   Font font = g.getFont();  // Dobi trenurni font
4   // Dobi velikost in ostale stvari o trenutnem fontu
5   FontMetrics metrics = g.getFontMetrics();  // Lahko ma kot argument font
6   g.setColor(color);
```



Slika 3: Ovalne oblike

```
7   g.setFont(font);
8   int dolzinaStringa = metrics.stringWidth(str);
9
10  // Generalna funkcija za draw
11  // Tako naredimo 2D pravokotnik
12  g.draw (new Rectangle2D.Double(x, y, sirina, visina));  // vsi parametri so double
13  g.draw (new RoundRectangle2D.Double(x, y, sirina, visina));
14  g.fill (new Ellipse2D.Double(x,y, sirina, visina));
15  // Drugi dve monosti sta Arc2D.OPEN, Arc2D.CHORD
16  g.fill (new Arc2D.Double(x,y, sirina, visina, zacKot, kot, Arc2D.PIE));
17
18  // fill lahko naredimo tudi na ostalih metodah
19  g.fillRect(x,y,sirina,visina)
20  g.fillRoundRect(x,y,sirina,visina,rH,rV)
21  g.fill3DRect(x,y,sirina,visina,dvig)
22  g.fillOval(x,y,sirina,visina)
23  g.fillPolygon(p)
24  g.fillPolygon(x,y,n)
25  g.fillArc(x,y,sirina,visina,zacKot,kot)
```

5 Razredi

5.1 Primer

```
1   public class Primer {
2           private int skrito;
3           private static final int skritoSamoEnoKoncno = 1;
4           protected int polaPola;  // Vidno razredom, ki dedujejo ta class, ostalim ne
5           public int vsiVidijo;
6
7           // Constructor
8           public Primer () {
9                   this.skrito = 0;
10                  this.polaPola = 0;
11          }
12          public Primer (int _skrito, int _polaPola) {
13                  this.skrito = _skrito;
14                  this.polaPola = _polaPola;
15          }
16
17          protected int Metoda(){                 // da lahko funkcijo Metoda klicemo  iz podrazredov mora biti VSAJ pr
18                  return 0;                  // po stopnjah dostopnosti private<default<protected<public
19                                             // vec info na https://www.tutorialspoint.com/java/java_access_modifiers.htm
20          }
21
22  }
23
24  public class PodPrimer extends Primer {
25          private int samoOdTega;
26
27          public PodPrimer(int _skrito, int _polaPola, int _samoOdTega) {
28                  super(_skrito, _polaPola);  // Klic contruktorja od Primer
29                  this.samoOdTega = _samoOdTega;
30          }
31          @Override
32          public int Metoda() {
33                  super.Metoda();  // Klic metode Primer
34                  return 1;  // Mogoce je narobe
35          }
36
```

```
}

public abstract class AbstraktenPrimer {
        // Enak kot primer samo da so v njem definirane metode in spremenljivke, ki jih kasneje
        // drugi razredi podeduje, kot nek modelcek po katerem se dela ostale clase
        // PAZI: ce podedujes tak class moras napisati definicije za vse njegove abstraktne metode

        // Ce imamo abstrakno metodo hocemo, da imajo vsi, ki se dedujejo iz tega to metodo
        // ampak jo vsak po svoje implementira
        public abstract int Metoda();

        public int get1(){        // abstrakten class ima lahko tudi navadne metode in spremenljivke, razlika med ab
                return 1;
        }
}

public interface interfacePrimer {
        // V interfacu samo specificiramo katere metode imamo(vse so abstrakne) in
        // tudi class sam je abstrakten, ce ga podedujemo modramo definirati vse njegove
        // metode

        public void Metoda();
}

public class interfacePodPrimer implements interfacePrimer {          // Lahko bi implementiral se en interface takol
                                                                    // JAVA NE podpira extendanja 2 classov; npr. Ovca e
        public void Metoda() {
                return 0;
        }

}

public static void main(String[] args) {
        Primer[] t = Primer[3];
        t[0] = new Primer();
        t[1] = new Primer(1,2);
        t[2] = new PodPrimer(1,2,3);
}


// Sortiranje objektov
public class Primerjava implements Comparable<Primerjava> {
        private int a;
        @Override
        public int compareTo (Primerjava other) {
                if (this.a < other.a) {
                        return -1;
                } else if (this.a > other.a) {
                        return 1;
                }
                return 0;
        }
}

// Drug Primer
import java.util.Comparator;

private class Obj {
        public int c;
}

private static class PrimerjajObj implements Comparator<Obj> {
        @Override
        public int compare (Obj a, Obj b) {
                if (a.c < b.c) {
                        return -1;
                } else if (a.c > b.c) {
                        return 1;
                }
                return 0;
        }
}

boolean jeTegaTipa = obj instanceof Primer;
//pomembno pri dedovanju je, da ce imamo class Macek extends Animal, potem: Macek a = new Macek(); a instanceof Anim
```

6 Podatkovne strukture

6.1 Array

```
import java.util.Arrays;

final int SIZE = 10;
// definicija(lahko je kateregakoli tipa, tudi class)
int[] seznam = new int[SIZE];  // to ga ze nastavi na default vrednost (0)

// napolni z 0 odKje(inkluzivno) do doKje(eksluzivno), ce spustimo ta dva argumenta
// bo napolnilo celoten array
Arrays.fill(seznam, 5, odKje, doKje);

// zgornja metoda je ekvivalenta temu
for (int i = 0; i < SIZE; ++i) {
        seznam[i] = 5;
}

Arrays.sort(seznam, fromIndex, toIndex);  //Uredi seznam narascajoce (1,2,3,4, ...)

// Ko imamo urejen sezna in samo v tem primeru lahko poklicemo to funkcijo
int kajIscemo = 5;  // seznam in kajIscemo morata biti enakega tipa
Arrays.binarySearch(seznam, kajIscemo);  // Ki nam vrne mesto elementa

boolean enaka = Arrays.equals(seznam1, seznam2);  // primerja
// za primerjavo gnezdenih seznamov
boolean enaka = Arrays.deepEquals(new int[][], new int[][]);


// Dinamicni array
import java.util.ArrayList;

ArrayList<Integer> seznam = new ArrayList<Integer>();

seznam.add(4);
seznam.add(1,3);  // Da na index 1 integer 3
seznam.remove(4);  // Odstrani specificen element
seznam.remove(0);  // Odstrani na specificnem indeksu
int pos = seznam.indexOf(3);  // Dobi pozicijo
seznam.set(0, 1);  // Nastavi element na poziciji 0 na 1
int dobi = seznam.get(2);  // Dobi elemnt na 2 poziciji
int s = seznam.size();
boolean vsebuje = seznam.contains(3);
```

6.2 Queue

```
import java.util.Queue;
import java.util.ArrayDeque;
import java.util.PriorityQueue;
import java.util.Comparator;

ArrayDeque<Integer> q = new ArrayDeque<Integer>();

q.addLast(3);  // doda na konec (addFirst doda od spredaj)
int prvi = q.pollFirst();  // Dobi in odstrani prvi element (pollLast dobi zadnjega)
int prvi = q.peekFirst();  // Dobi vendar ne odstani
// Obe metodi vrneta null ce je queue prazen

boolean prazna = q.isEmpty();

// Vrsta pri kateri so elementi urejeni po prioritety
Comparator<Objekt> comparator = new PrimerjaObjekt();  // kako se bojo elementi primerjali
PriorityQueue<Objekt> q = new PriorityQueue<Objek>(velikost, comparator);

// Ta objekt je lahko tudi tipa int, long, string itd. ali pa celo kaksen svoj custom class
q.add(objekt);
Objekt o = q.poll();  // Metodi delujeta enako kot zgoraj le da vrneta prvega po prioriteti
Objekt o = q.peek();
boolean prazna = q.isEmpty();

private static class PrimerjajObjekt implements Comparator<Objekt> {
        @Override
        public int compare (Objekt a, Objekt b) {
                if (a.c > b.c) {
                        return -1;
                }
                if (a.c < b.c) {
```

```
                        return 1;
                }
                return 0;
        }

}
```

6.3 Stack

```
import java.util.Stack;

// Deluje na principu LIFO (last in first out)
Stack s = new Stack();
int a = 4;
s.push(a);  // Doda na vrh sklada
int b = s.pop();  // Vzame z vrha sklada
int b = s.peek();  // Pogleda na vrh sklada
boolean prazen = s.empty();
```

6.4 Set

```
import java.util.Set;
import java.util.HashSet;
import java.util.TreeSet;

// To je mnozica in v njej se vsi elementi pojavijo samo enkrat
Set<Integer> set = new HashSet<Integer>();

set.add(5);  // dodajanje
set.remove(5);  // Odstrani specificen element
int s = set.size();  // Velikost
boolean prazna = set.isEmpty();  // Prazen
boolean notri = set.contains(5);  // Ali ima dolocen element

// Posortirano
TreeSet<Integer> drevensaOblika = new TreeSet<Integer>(set);
int prvi = drevesnaOblika.first();  // poolFirst ga se odstrani
int zadnji = drevesnaOblika.last();  // poolLast ga se odstrani

int infinum = drevesnaOblika.floor(5);  // Vrne najvecji element, ki je manjsi od 5
int infinum = drevesnaOblika.ceiling(5);  // Vrne najmanjsi element, ki je vecji od 5

// + ima se vse zgornje funkcije od normalnega Set-a
```

7 Algoritmi

7.1 Fload Fill

Naloga DN09

```
// Struct(tip), ki drzi kordinati in ceno na kordinati
private static class Cord {
        public int x;
        public int y;
        public int c;

        public Cord (int _x, int _y) {
                this.x = _x;
                this.y = _y;
                this.c = 0;
        }

        public Cord (int _x, int _y, int _c) {
                this.x = _x;
                this.y = _y;
                this.c = _c;
        }
}

// Nastavimo vsa polja na default vrednosti
int[][] zemlja = new int[n][n];
```

```
for (int i = 0; i < n; ++i) {
        for (int j = 0; j < n; ++j) {
                zemlja[i][j] = -1;
        }
}
// Vse mozne smeri v katere lahko gremo
Cord[] s = new Cord[] {new Cord(0,1), new Cord(0,-1),
                       new Cord(1,0), new Cord(-1,0)};
int x = n/2;
int y = x;

// Naredimo vrsto v katero bomo dajali katera polja moramo obiskati in iz nje
// vzemali katero bo nase naslednje polje
ArrayDeque<Cord> q = new ArrayDeque<Cord>();
q.addLast(new Cord(x,y,c));  // Naslednje polje
while (!q.isEmpty()) {  // Dokler imamo se katero polje za obiskati
        Cord t = q.pollFirst();
        // Ce smo ze bili na polju ga preskocimo
        if (zemlja[t.x][t.y] != -1) {
                continue;
        }
        zemlja[t.x][t.y] = t.c;  // Nastavimo ceno polja
        // Gremo cez vse njegove sosede
        for (int i = 0; i < s.length; ++i) {
                if (t.x + s[i].x >= 0 && t.y + s[i].y >= 0 &&
                        t.x + s[i].x < n && t.y + s[i].y < n &&
                        (zemlja[t.x + s[i].x][t.y + s[i].y] == -1)) {  // Pogledamo ce je veljaven sosed
                        q.addLast(new Cord(t.x + s[i].x, t.y + s[i].y, Math.max(0,t.c - d)));
                }
        }
}
```

7.2 BFS

Naloga DN07

```
private static class Povezava {
    public int pretekliCas;
    public int indeks;

    Povezava (int _pretekliCas, int _indeks) {
        this.pretekliCas = _pretekliCas;
        this.indeks = _indeks;
    }
}

int n = in.nextInt();
int cas = in.nextInt();
int[][] sosedi = new int[n][n];
// Preberem vse sosede
for (int i = 0; i < n; ++i) {
    for (int j = 0; j < n; ++j) {
        sosedi[i][j] = in.nextInt();
    }
}
// Nastavim cas potovanja na najvec mogoce
int[] casPotovanja = new int[n];
for (int i = 0; i < n; ++i) {
    casPotovanja[i] = 2147483647;
}

// Naredim Queue z svojim classom Povezava v katerega bom potem shranil
// povezava ki jih morem se obdelati
ArrayDeque<Povezava> q = new ArrayDeque<Povezava>();
// Dam zacetnika sporocil v povezavo
q.addLast(new Povezava(0,0));
while (!q.isEmpty()) {
        // Vzamem eno povezavo
    Povezava tr = q.pollFirst();
        // Ce je cas trenutne poti vecji od prejsnje poti potem ne rabimo iti po tej poti
    if (tr.pretekliCas > casPotovanja[tr.indeks]) {
        continue;
    }else {
                // Drugace je to trenutna najkrajsa pot
        casPotovanja[tr.indeks] = tr.pretekliCas;
    }
        // Gremo cez vse sosede trenutne povezave
```

```
42      for (int i = 0; i < n; ++i) {
43          if (sosedi[tr.indeks][i] != 0) {
44              Povezava naslednja = new Povezava(
45              tr.pretekliCas + sosedi[tr.indeks][i], i);
46              q.addLast(naslednja);
47          }
48
49      }
50  }
51
52  int rez = 0;
53  for (int i = 0; i < n; ++i) {
54      if (casPotovanja[i] <= cas) {
55          rez++;
56      }
57  }
```

7.3 Fast Power

Naloga DN04

```
1   int mod = 13;  // Neko prastevilo, ki predstavlja sistem
2
3   private int FastPow (int a, long e) {
4           int rez = 1;
5           while (e > 0) {
6                   if (e % 2 == 1) {
7                           rez = (rez * a) % mod;
8                   }
9                   a = a * a % mod;
10                  e /= 2;
11          }
12
13          return rez;
14  }
```

8 TJ.exe

8.1 Uporaba

tj.exe <Program.java> <testi> <rezultati> -> normalno
tj.exe <razredi> <testi> <rezultati> -> razredi
tj.exe . . . -> slike
tj.exe -t 5s -> cas
tj.exe -p 5-10 -> primeri
Rocno:
javac program.java
java program < input.txt > output.txt
(java Program rezultat.png 700x500 za slike)
fc output.txt pravilno.txt (Linux/Mac diff)