

# Dokumentacija za Izpit

Rok Kos

verzija: 27. januar 2017

---

# Kazalo

<b>1</b>	<b>Input/Output</b>	<b>3</b>
1.1	Input . . . . .	3
1.2	Output . . . . .	3
<b>2</b>	<b>Delo s števili</b>	<b>3</b>
2.1	Parsing . . . . .	3
2.2	Math knjiznica . . . . .	3
2.3	Int/Long class . . . . .	4
<b>3</b>	<b>Delo z besedili</b>	<b>4</b>
3.1	String class . . . . .	4
<b>4</b>	<b>Razredi</b>	<b>4</b>
4.1	Primer . . . . .	4
<b>5</b>	<b>Podatkovne strukture</b>	<b>6</b>
5.1	Array . . . . .	6
5.2	Queue . . . . .	6
5.3	Stack . . . . .	7
<b>6</b>	<b>Algoritmi</b>	<b>7</b>
6.1	Flood Fill . . . . .	7
6.2	BFS . . . . .	8
6.3	Fast Power . . . . .	8
<b>7</b>	<b>TJ.exe</b>	<b>9</b>
7.1	Uporaba . . . . .	9

---

# 1 Input/Output

## 1.1 Input

```
1 import java.util.Scanner;
2 Scanner in = new Scanner(System.in);
3 byte b = in.nextByte();
4 int i = in.nextInt();
5 long l = in.nextLong();
6 double d = in.nextDouble();
7 String s = in.next(); // Vrne naslednji string
8 String line = in.nextLine(); // Prebere celotno vrstico in skoci v novo
9
10 // Branje do konca inputa
11 while(in.hasNextInt()){
12     int a = in.nextInt();
13 }
14 // Namesto hasNextInt bi lahko bilo tudi:
15 // -hasNext()
16 // -hasNextDouble()
17 // -hasNextLong
18 // -hasNextLine
```

## 1.2 Output

```
1 System.out.println(dnevi + ". dan: " + prej + " -> " + d + " (prehodil " + p + ")");
2 System.out.print("Ne gre v naslednjo vrsico" + 5);
3
4 System.out.printf("%d. dan: %d -> %d (prehodil %d)%n", dnevi, prej, d, p);
5 System.out.format("%1$+020.10f", Math.PI); // Enako kot print pri println in print(ne gre v novo vrstico)
6 // "1 dolar kateri argument"
7 // + pomeni predznacen, 0 pomeni da naj bodo spredaj vodilne 0
8 // 20.10 pomeni 20 mest spredaj in na 10 decimalk
9 // FORMATER
10 // %d - int, long, byte, %f - double, float, %s -string, %n - newline
11 // + -> predznak, - -> levo poravnan, 010.5 -> vodilne nicle 10 mest z 5 decimalkami
```

# 2 Delo s števili

## 2.1 Parsing

```
1 String s = "123456789";
2 int a = Integer.parseInt(s);
3 long l = Long.parseLong(s);
4 double d = Double.parseDouble(s);
5
6 String besedilo = "Neka dolga poved v kateri mores dobit vsako besedo";
7 String[] besede = besedilo.split(" "); // Lahko tudi po , . !, ali celo po regex
8 for (int i = 0; i < besede.length; ++i) {
9     System.out.println(besede[i]);
10 }
```

## 2.2 Math knjižnica

```
1 double pi = Math.PI; // Math.E
2 double a = abs(a); // Tudi za int, long in float
3 double naj = Math.max(int, int) // lahko tudi double, float in long
4 double naj = Math.min(int, int) // lahko tudi double, float in long
5
6 // Trigonometrične funkcije
7 double kot = asin(val) / Math.PI * 180; // acos, atan vrne vrednost v PI radianih
8 double val = sin(kot * Math.PI / 180); // cos, tan
9 // Možno pretvorba tudi z toDegrees ali toRadians
10
11 // Hiperbolične funkcije
12 double h = sinh(val); // cosh, tanh
13
14 // Zaokroževanje
15 double navzgor = Math.ceil(decimalalka);
16 double navzdol = Math.floor(decimalalka);
17 long navzdol = Math.round(decimalalka); // vrne celo stevilo(lahko tudi int)
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.toHexString();
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 -1 ce je str1 pred str2 in 1 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); // zamenja vse pojavitve, ki ustrezajo regex (lahko tudi normalen string)
16 String rep = str1.replaceFirst(regex, sCim); // zamenja 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.endsWith(str2);

```

## 4 Razredi

### 4.1 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     private int Metoda () {
18         return 0;
19     }
20
21 }
22
23 public class PodPrimer extends Primer { // Lahko bi ezendali si en class takole : Primer, SeEnPrimer
24     private int samoOdTega;
25
26     public PodPrimer(int _skrito, int _polaPola, int _samoOdTega) {
27         super(_skrito, _polaPola); // Klic konstruktorja od Primer
28         this.samoOdTega = _samoOdTega;
29     }
30     @Override
31     public int Metoda() {
32         super.Metoda(); // Klic metode Primer
33         return 1; // Mogoce je narobe
34     }
35
36 }
37
38 public abstract class AbstraktenPrimer {
39     // Enak kot primer samo da so v njem definirane metode in spremenljivke, ki jih kasneje
40     // drugi razredi podeduje, kot nek modelcek po katerem se dela ostale clase
41     // PAZI: ce podedujes tak class moras napisati definicije za vse njegove abstraktne metode
42
43     // Ce imamo abstrakno metodo hocemo, da imajo vsi, ki se dedujejo iz tega to metodo
44     // ampak jo usak po svoje implementira
45     public abstract int Metoda();
46 }
47
48 public interface interfacePrimer {
49     // V interfacu samo specificiramo katere metode imamo(vse so abstrakne) in
50     // tudi class sam je abstrakten, ce ga podedujemo modramo definirati vse njegove
51     // metode
52
53     public void Metoda();
54 }
55
56 public class interfacePodPrimer implements interfacePrimer {
57     public void Metoda() {
58         return 0;
59     }
60
61 }
62
63 public static void main(String[] args) {
64     Primer[] t = Primer[3];
65     t[0] = new Primer();
66     t[1] = new Primer(1,2);
67     t[2] = new PodPrimer(1,2,3);
68 }
69
70
71 // Sortiranje objektov
72 public class Primerjava implements Comparable<Primerjava> {
73     private int a;
74     @Override
75     public int compareTo (Primerjava other) {
76         if (this.a < other.a) {
77             return -1;
78         } else if (this.a > other.a) {
79             return 1;
80         }
81         return 0;
82     }
83 }
84
85 // Drug Primer
86 import java.util.Comparator;
87
88 private class Obj {
89     public int c;
90 }
91
92 private static class PrimerjajObj implements Comparator<Obj> {

```

---

```

93         @Override
94         public int compare (Obj a, Obj b) {
95             if (a.c < b.c) {
96                 return -1;
97             } else if (a.c > b.c) {
98                 return 1;
99             }
100             return 0;
101         }
102     }

```

## 5 Podatkovne strukture

### 5.1 Array

```

1  import java.util.Arrays;
2
3  final int SIZE = 10;
4  // definicija (lahko je kateregakoli tipa, tudi class)
5  int[] seznam = new int[SIZE]; // to ga ze nastavi na default vrednost (0)
6
7  // napolni z 0 odKje(inkluzivno) do doKje(eksluzivno), ce spustimo ta dva argumenta
8  // bo napolnilo celoten array
9  Arrays.fill(seznam, 5, odKje, doKje);
10
11 // zgornja metoda je ekvivalentna temu
12 for (int i = 0; i < SIZE; ++i) {
13     seznam[i] = 5;
14 }
15
16 Arrays.sort(seznam, fromIndex, toIndex); //Uredi seznam narascajoco (1,2,3,4, ...)
17
18 // Ko imamo urejen sezna in samo v tem primeru lahko poklicemo to funkcijo
19 int kajIscemo = 5; // seznam in kajIscemo morata biti enakega tipa
20 Arrays.binarySearch(seznam, kajIscemo); // Ki nam vrne mesto elementa
21
22 boolean enaka = Arrays.equals(seznam1, seznam2); // primerja
23 // za primerjavo gnezdenih seznamov
24 boolean enaka = Arrays.deepEquals(new int[] {}, new int[] {});

```

### 5.2 Queue

```

1  import java.util.Queue;
2  import java.util.ArrayDeque;
3  import java.util.PriorityQueue;
4  import java.util.Comparator;
5
6  ArrayDeque<Integer> q = new ArrayDeque<Integer>();
7
8  q.addLast(3); // doda na konec (addFirst doda od spredaj)
9  int prvi = q.pollFirst(); // Dobi in odstrani prvi element (pollLast dobi zadnjega)
10 int prvi = q.peekFirst(); // Dobi vendar ne odstani
11 // Obe metodi vrmeta null ce je queue prazen
12
13 boolean prazna = q.isEmpty();
14
15 // Vrsta pri kateri so elementi urejeni po prioritety
16 Comparator<Objekt> comparator = new PrimerjaObjekt(); // kako se bojo elementi primerjali
17 PriorityQueue<Objekt> q = new PriorityQueue<Objekt>(velikost, comparator);
18
19 // Ta objekt je lahko tudi tipa int, long, string itd. ali pa celo kaksen svoj custom class
20 q.add(objekt);
21 Objekt o = q.poll(); // Metodi delujeta enako kot zgoraj le da vrmeta prvega po prioriteti
22 Objekt o = q.peek();
23 boolean prazna = q.isEmpty();
24
25 private static class PrimerjaObjekt implements Comparator<Objekt> {
26     @Override
27     public int compare (Objekt a, Objekt b) {
28         if (a.c > b.c) {
29             return -1;
30         }
31         if (a.c < b.c) {
32             return 1;
33         }

```

---

```

34         return 0;
35     }
36
37 }

```

## 5.3 Stack

```

1  import java.util.Stack;
2
3  // Deluje na principu LIFO (last in first out)
4  Stack s = new Stack();
5  int a = 4;
6  s.push(a); // Doda na vrh sklada
7  int b = s.pop(); // Vzame z vrha sklada
8  int b = s.peek(); // Pogleda na vrh sklada
9  boolean prazen = s.empty();

```

# 6 Algoritmi

## 6.1 Flood Fill

Naloga DN09

```

1  // Struct(tip), ki drži kordinati in ceno na kordinati
2  private static class Cord {
3      public int x;
4      public int y;
5      public int c;
6
7      public Cord(int _x, int _y) {
8          this.x = _x;
9          this.y = _y;
10         this.c = 0;
11     }
12
13     public Cord(int _x, int _y, int _c) {
14         this.x = _x;
15         this.y = _y;
16         this.c = _c;
17     }
18 }
19
20 // Nastavimo vsa polja na default vrednosti
21 int[][] zemlja = new int[n][n];
22 for (int i = 0; i < n; ++i) {
23     for (int j = 0; j < n; ++j) {
24         zemlja[i][j] = -1;
25     }
26 }
27 // Vse mozne smeri v katere lahko gremo
28 Cord[] s = new Cord[] {new Cord(0,1), new Cord(0,-1),
29                         new Cord(1,0), new Cord(-1,0)};
30 int x = n/2;
31 int y = x;
32
33 // Naredimo vrsto v katero bomo dajali katera polja moramo obiskati in iz nje
34 // vzemali katero bo nase naslednje polje
35 ArrayDeque<Cord> q = new ArrayDeque<Cord>();
36 q.addLast(new Cord(x,y,c)); // Naslednje polje
37 while (!q.isEmpty()) { // Dokler imamo se katero polje za obiskati
38     Cord t = q.pollFirst();
39     // Ce smo ze bili na polju ga preskocimo
40     if (zemlja[t.x][t.y] != -1) {
41         continue;
42     }
43     zemlja[t.x][t.y] = t.c; // Nastavimo ceno polja
44     // Gremo cez vse njegove sosede
45     for (int i = 0; i < s.length; ++i) {
46         if (t.x + s[i].x >= 0 && t.y + s[i].y >= 0 &&
47             t.x + s[i].x < n && t.y + s[i].y < n &&
48             (zemlja[t.x + s[i].x][t.y + s[i].y] == -1)) { // Pogledamo ce je veljaven sosed
49                 q.addLast(new Cord(t.x + s[i].x, t.y + s[i].y, Math.max(0,t.c - d)));
50             }
51         }
52     }
53 }

```

---

```

51     }
52 }

```

## 6.2 BFS

Naloga DN07

```

1  private static class Povezava {
2      public int pretekliCas;
3      public int indeks;
4
5      Povezava (int _pretekliCas, int _indeks) {
6          this.pretekliCas = _pretekliCas;
7          this.indeks = _indeks;
8      }
9  }
10
11  int n = in.nextInt();
12  int cas = in.nextInt();
13  int[] [] sosedi = new int[n][n];
14  // Preberem vse sosed
15  for (int i = 0; i < n; ++i) {
16      for (int j = 0; j < n; ++j) {
17          sosedi[i][j] = in.nextInt();
18      }
19  }
20  // Nastavim cas potovanja na največ mogoče
21  int[] casPotovanja = new int[n];
22  for (int i = 0; i < n; ++i) {
23      casPotovanja[i] = 2147483647;
24  }
25
26  // Naredim Queue z svojim classom Povezava v katerega bom potem shranil
27  // povezava ki jih morem se obdelati
28  ArrayDeque<Povezava> q = new ArrayDeque<Povezava>();
29  // Dam zacetnika sporocil v povezavo
30  q.addLast(new Povezava(0,0));
31  while (!q.isEmpty()) {
32      // Vzamem eno povezavo
33      Povezava tr = q.pollFirst();
34      // Če je cas trenutne poti vecji od prejsnje poti potem ne rabimo iti po tej poti
35      if (tr.pretekliCas > casPotovanja[tr.indeks]) {
36          continue;
37      } else {
38          // Drugace je to trenutna najkrajša pot
39          casPotovanja[tr.indeks] = tr.pretekliCas;
40      }
41      // Gremo cez vse sosedne 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 }

```

## 6.3 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    }
```

## 7 TJ.exe

### 7.1 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)