Java tečaj

1. dio

Uvod

© 2012.

Najvažnije adrese

- Java
  http://www.oracle.com/technetwork/java/javase/downloads/index.html
- Dokumentacija API-ja:
  http://docs.oracle.com/javase
- Eclipse (trenutno 3.7, tj. Indigo)
  http://www.eclipse.org/
  (Eclipse IDE for Java EE Developers)

“Hello World” program

```
package hr.fer.zemris.java.tecaj_1;

/**
 * Demonstracijski program.
 * @author Marko Cupic
 * @version 1.0
 */
public class HelloWorld {

    /**
     * Metoda koja se poziva prilikom pokretanja
     * programa. Argumenti su objasnjeni u nastavku.
     * @param args Argumenti iz komandne linije.
     */
    public static void main(String[] args) {
        System.out.println("Hello World!");
    }
}
```

Naziv paketa

Razred

Metoda main

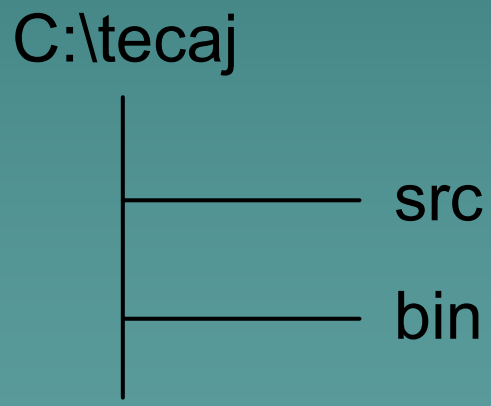
“Hello World” program

- Otvorite *Command prompt*!
- Nađite mjesto gdje možete zapisivati datoteke (npr. `C:\tecaj`)
- Udite u taj direktorij:

```
C:
cd C:\tecaj
```

“Hello World” program

- Napravite dva direktorija:
  - src – tu će doći izvorni programi
  - bin – tu će doći izvršivi kod



“Hello World” program

- Izgradite strukturu direktorija koja odgovara paketu hr.fer.zemris.java.tecaj_1:

```
mkdir src\hr
mkdir src\hr\fer
mkdir src\hr\fer\zemris
mkdir src\hr\fer\zemris\java
mkdir src\hr\fer\zemris\java\tecaj_1
```

“Hello World” program

- U *notepadu* prepišite program, i snimite ga u direktorij:
  `src\hr\fer\zemris\java\tecaj_1`
  pod nazivom:
  `HelloWorld.java`

- Pazite! Notepad zna dodati još ekstenziju txt pa dobijete HelloWorld.java.txt → to ne valja!

“Hello World” program

- Uočite kako **ime datoteke mora odgovarati imenu razreda** (ono iza class), i pri tome paziti na velika i mala slova!

  class HelloWorld ←→ HelloWorld.java

“Hello World” program

- Prevođenje programa
  - morate biti pozicionirani u direktoriju koji sadrži direktorije `src` i `bin`
  - Zadajte:
    ```
    javac -cp bin -sourcepath src -d bin
    src\hr\fer\zemris\java\tecaj_1\HelloWorld.java
    ```
    (sve u jednom retku!)
- Rezultat je `HelloWorld.class` datoteka u strukturi direktorija unutar direktorija `bin`

“Hello World” program

- Pokretanje programa
  `java -cp bin hr.fer.zemris.java.tecaj_1.HelloWorld`
  (sve u jednom retku!)
- Ne piše se `.class` ekstenzija
- Navodi se puno ime razreda
- “`-cp bin`” kaže gdje da traži `.class`
  (to je kao varijabla okruženja PATH za virtualni stroj koji izvodi program)

“Hello World” program

- Pokretanje programa – opći format
  `java -cp staze puno.ime.razreda argumenti`
  (sve u jednom retku!)

- “`-cp staze`” može biti nepotreban, ovisno o postojanju varijable okruženja `CLASSPATH`

“Hello World” program

- “Hello World” je primjer jednog (javnog) razreda (engl. **class**)
- Ime datoteke == ime tog razreda
- Razredi se organiziraju hijerarhijski u pakete – slično kao datoteke i kazala
  - **package** ključna riječ
- Metode koje pripadaju samom razredu: **static**

“Hello World” program

- Pokretanje programa – **main** metoda

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

- Argumenti iz komandne linije
  - String[] args → polje stringova

Komentiranje koda

- Obični komentari
  - /* komentar */ i // komentar
- Komentari iz kojih se generira dokumentacija (javadoc komentari)
  - /** *komentar* */
- Javadoc komentari za:
  - Same razrede
  - Pojedine metode

Komentiranje koda

- Javadoc komentari sadrže oznake oblika @naziv vrijednost, npr.
  - @author ime_autora, npr.
    @author Marko Cupic
  - @version verzija_razreda, npr.
    @version 1.0
  - @param ime_argumenta opis
    @param x broj čiji sinus treba izračunati
  - @return opis
    @return vraća sinus zadanog broja

Komentiranje koda

```
package hr.fer.zemris.java.tecaj_1;

/**
 * @author Marko Cupic
 * @version 1.0
 */
public class HelloWorld {

    /**
     * Metoda koja se poziva prilikom pokretanja
     * programa. Argumenti su objasnjeni u nastavku.
     * @param args Argumenti iz komandne linije.
     */
    public static void main(String[] args) {
            System.out.println("Hello World!");
    }
}
```

Ispis na ekran (.out), ili na izlaz za pogreške (.err)
Postoji i printf funkcija!

Komentiranje koda

```
/**
 * Metoda računa y-tu potenciju od broja x.
 * @param x argument x
 * @param y argument y; mora biti nenegativan
 * @return vraća iznos izraza x^y
 */
public static double pow(int x, int y) {
    …
}
}
```

Tipovi varijabli

Primitivni	Objektni omotači (wrappers)	Zauzeće
byte	Byte	1 oktet / ?, signed
short	Short	2 okteta / ?, signed
int	Integer	4 okteta / ?, signed
long	Long	8 okteta / ?, signed
char	Character	2 okteta / ?, UTF-16
-	String	?
boolean	Boolean	1 bit / ?
float	Float	4 okteta / ?
double	Double	8 okteta / ?

Pravila

- boolean: true, false (različito od 0, 1)
- if(boolean_izraz) {…}
- Pogrešno:

```
int v = 7;
if(v) {…}
```

- Pogrešno:

```
int x = 7;
while(x) { x--; }
```

Pravila

- Nikada:

```
double x = nestoIzracunaj(…);
if(x==0.7) {…}
```

Ne koristiti == za usporedbe decimalnih tipova!

```
if(Math.abs(x-0.7)<1E-6) {…}
```

Primitivni tipovi ⇔ text

- Koristiti **statičke** funkcije wrappera:

```
String sBroj = "375.83";
double dBroj =
    Double.parseDouble(sBroj);
String sBroj2 =
    Double.toString(dBroj);
```

- Pogledati javadoc za dokumentaciju

Nekoliko jednostavnih primjera

- Napisati program koji će na zaslon ispisati argumente koje dobiva prilikom pokretanja programa

```
package hr.fer.zemris.java.tecaj_1;

/**
 * @author Marko Cupic
 * @version 1.0
 */
public class IspisArgumenata {

    /**
     * Metoda koja se poziva prilikom pokretanja
     * programa. Argumenti su objasnjeni u nastavku.
     * @param args Argumenti iz komandne linije.
     */
    public static void main(String[] args) {
        int brojArgumenata = args.length;

        for(int i = 0; i < brojArgumenata; i++) {
            System.out.println(
                "Argument " + (i+1) + ": " + args[i]
            );
        }
    }
}
```

Svako polje ima svojstvo ".length"

Nekoliko jednostavnih primjera

- Napisati program koji će prilikom pokretanja primiti jedan argument (x), te izračunati koliko iznosi e^x razvojem u Taylorov red
- Razvoj riješiti u zasebnoj funkciji
- Program na zaslon mora ispisati rezultat

```
package hr.fer.zemris.java.tecaj_1;

/**
 * @author Marko Cupic
 * @version 1.0
 */
public class SumaReda {

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

    private static double racunajSumu(double broj) {
    ...
    }

}
```

```
/**
 * Metoda koja se poziva prilikom pokretanja
 * programa. Argumenti su objasnjeni u nastavku.
 * @param args Argumenti iz komandne linije.
 */
public static void main(String[] args) {

	if(args.length != 1) {
		System.err.println(
			"Program mora imati jedan argument!"
		);
		System.exit(1);
	}

	double broj = Double.parseDouble(args[0]);

	System.out.println("Racunam sumu...");

	double suma = racunajSumu(broj);

	System.out.println("f(" + broj + ")=" + suma + ",");
}
```

Svako polje ima svojstvo ".length"

```
/**
 * Racuna e^x razvojem u Taylorov red, prema formuli:
 * e^x=1+x+(x^2/(2!))+(x^3/(3!))+(x^4/(4!))+...
 * @param broj argument funkcije e^x
 * @return iznos funkcije u tocki x=broj dobiven kao
 *         suma prvih 10 clanova Taylorovog reda.
 */
private static double racunajSumu(double broj) {
	double suma = 0.0;
	double potencija = 1.0;
	double faktorijela = 1.0;

	suma += 1.0;

	for(int i = 1; i < 10; i++) {
		potencija = potencija * broj;
		faktorijela = faktorijela * i;
		suma += potencija/faktorijela;
	}

	return suma;
}
```

Nekoliko jednostavnih primjera

- Napisati program koji sadrži funkciju koja prima polje double-ova, koje ispisuje na zaslon po zadanom formatu
- Napisati glavni program koji će brojeve ispisati
  - Najmanje tri mjesta za cijelobrojni dio, dva mjesta za decimalni
  - Dva + dva mjesta s obaveznim ispisom predznaka

```
package hr.fer.zemris.java.tecaj_1;

import java.text.DecimalFormat;

/**
 * @author Marko Cupic
 * @version 1.0
 */
public class FormatiraniIspisDecBrojeva {

    public static void ispis(double[] polje, String format) {
      ...
    }

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

```
/**
 * Metoda na standardni izlaz ispisuje polje decimalnih
 * brojeva prema zadanom formatu.
 * @param polje polje decimalnih brojeva koje treba ispisati.
 * @param format format koji govori kako polje treba ispisati.
 * @see DecimalFormat
 */
public static void ispis(double[] polje, String format) {
        DecimalFormat formatter = new DecimalFormat(
                        format
        );
        for(int i = 0; i < polje.length; i++) {
                        System.out.println(
                                        "(" + i + "): [" +
                                        formatter.format(polje[i]) +
                                        "]"
                        );
        }
}
```

```
/**
 * Metoda koja se poziva prilikom pokretanja
 * programa. Argumenti su objasnjeni u nastavku.
 * @param args Argumenti iz komandne linije.
 */
public static void main(String[] args) {
	double[] brojevi = new double[] {
			3.712, 55.813, 55.816, -4.18
	};
	ispis(brojevi, "000.00");
	ispis(brojevi, "+00.00;-00.00");
}
```

Nekoliko jednostavnih primjera

- Napisati program koji će s tipkovnice čitati decimalni broj po broj i računati njihovu sumu, sve dok se upisuju nenegativni brojevi

```
package hr.fer.zemris.java.tecaj_1;

import java.io.BufferedReader;
import java.io.IOException;
import java.io.InputStreamReader;

/**
 * @author marcupic
 * @version 1.0
 */
public class CitanjeSTipkovnice {

    public static void main(String[] args) throws
    IOException {
        ..
    }
}
```

```
public static void main(String[] args) throws IOException {
        System.out.println("Program za računanje sume pozitivnih brojeva.");
        System.out.println("Unosite brojeve, jedan po retku.");
        System.out.println(
                "Kada unesete negativan broj, ispisat ce se suma.");

        BufferedReader reader = new BufferedReader(
                new InputStreamReader(System.in)
        );

        double suma = 0.0;
        while(true) {
                String redak = reader.readLine();
                if(redak==null) break;
                double broj = Double.parseDouble(redak);
                if(broj<0) break;
                suma += broj;
        }

        System.out.print("Suma je: ");
        System.out.println(suma);

        reader.close();
}
```

- Za napredniju obradu ulaza postoji razred java.util.Scanner

```
Scanner sc = new Scanner(System.in);
int i = sc.nextInt();
```

```
Scanner sc = new Scanner(new File("myNumbers"));
while (sc.hasNextLong()) {
    long aLong = sc.nextLong();
}
```

```
String input = "1 fish 2 fish red fish blue fish";
Scanner s = new
Scanner(input).useDelimiter("\\s*fish\\s*");
System.out.println(s.nextInt());
System.out.println(s.nextInt());
System.out.println(s.next());
System.out.println(s.next());
s.close();
```

- Za naprednije generiranje izlaza postoji podrška formatiranju

```
System.out.printf("%s\n", "bla");
System.out.format("%s\n", "bla");
```

```
String novi = String.format("%4$s %3$s %2$s %1$s
%4$s %3$s %2$s %1$s",
                "a", "b", "c", "d");
```

➔ "d c b a d c b a"

- Za detalje oko formatnog stringa pogledati dokumentaciju: http://docs.oracle.com/javase/7/docs/api/java/util/Formatter.html#syntax

Rad sa stringovima

- U Javi String nije polje znakova terminirano s ‘\0’
- Kako se točno String pohranjuje – nije nas briga
- Zahvaljujući tome, Java omogućava lakše baratanje Stringovima
- Važno: stringovi su nepromijenjivi (immutable); metode koje nešto mijenjaju vraćaju nove stringove!

```
package hr.fer.zemris.java.tecaj_1;

/**
 * @author Marko Cupic
 * @version 1.0
 */
public class RadSaStringovima {

    /**
     * Metoda koja se poziva prilikom pokretanja
     * programa. Argumenti su objasnjeni u nastavku.
     * @param args Argumenti iz komandne linije.
     */
    public static void main(String[] args) {
          ispis1();
          ispis2();
          ispis3();
          ispis4();
    }

}
```

```
/**
 * Demonstracija zbrajanja stringova.<br>
 * Zbrajanje uporabom operatora + kroz vise naredbi.
 * Vrlo neefikasno!
 */
private static void ispis1() {
	String tekst = null;

	tekst = "The quick " + "brown ";
	tekst += "fox jumps over ";
	tekst += 3;
	tekst += " lazy dogs.";

	System.out.println(tekst);
}
```

```
/**
 * Demonstracija zbrajanja stringova.<br>
 * Zbrajanje operatorom + u jednoj naredbi. Efikasnije.
 */
private static void ispis2() {
	String tekst = null;

	int broj = 3;

	tekst = "The quick brown fox jumps over " +
		broj + " lazy dogs.";

	System.out.println(tekst);
}
```

```
/**
 * Demonstracija zbrajanja stringova.<br>
 * Zbrajanje uporabom StringBuffer objekta. Jednako efikasno
 * kao i primjer 2? Inicijalno se stvara spremnik
 * velicine 16 koji se tri puta realocira kako bi se prosirio.
 * Napomena: prije Java 5.0 koristio se StringBuffer koji je bitno
 * sporiji (ali je višedretveno siguran).
 */
private static void ispis3() {
	String tekst = null;

	StringBuilder sb = new StringBuilder();

	sb.append("The quick ").append("brown ");
	sb.append("fox jumps over ").append(3);
	sb.append(" lazy dogs.");

	tekst = sb.toString();

	System.out.println(tekst);
}
```

```
/**
 * Demonstracija zbrajanja stringova.<br>
 * Zbrajanje uporabom StringBuffer objekta. Najefikasnije
 * ako unaprijed znamo potrebnu velicinu spremnika. U primjeru
 * se alocira spremnik velicine 50 znakova.
 * Napomena: prije Java 5.0 koristio se StringBuffer koji je bitno
 * sporiji (ali je višedretveno siguran).
 */
private static void ispis4() {
	String tekst = null;
	StringBuilder sb = new StringBuilder(50);

	sb.append("The quick ").append("brown ");
	sb.append("fox jumps over ").append(3);
	sb.append(" lazy dogs.");

	tekst = sb.toString();

	System.out.println(tekst);
}
```