

Java-projekt @ ZEMRIS

Osnove programskog jezika Java

Uvod

© 2020.

A stylized, dark teal silhouette of a mountain range is positioned in the bottom right corner of the slide, partially overlapping the copyright notice.

Korisna adresa

- ◆ Knjiga u nastajanju uz ovu vještinu

<http://java.zemris.fer.hr/nastava/opjj/>

- NE printati sve – poglavlja se još mijenjaju i nastajat će kako ide vještina
- U nazivu knjige gledajte najnoviji datum

Korisna adresa

- ♦ Na predavanjima ćemo koristiti
 - Javu 13
 - Eclipse
 - Apache Maven
 - JUnit 5
- ♦ Nećemo koristiti sve novitete svježijih inačica Jave (ali neke hoćemo)

Podešavanje varijabli okruženja

- ◆ Ako je Java dobro instalirana, kada u naredbenom retku zadate naredbe:

`java -version`

odnosno:

`javac -version`

dobit će se ispisane verzije programa

“Hello World” program

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

Naziv paketa

```
/**
```

```
 * Demonstracijski program.
```

```
 * @author Marko Cupic
```

```
 * @version 1.0
```

```
 */
```

```
public class HelloWorld {
```

Razred

```
/**
```

```
 * 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!");
```

```
    Util.sayHi();
```

```
}
```

```
}
```

Metoda main

“Hello World” program

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

Naziv paketa

```
/**
```

```
 * Demonstracijski program – pomoćni razred.
```

```
 * @author Marko Cupic
```

```
 * @version 1.0
```

```
 */
```

```
public class Util {
```

Razred

```
    /**
```

```
     * Pomoćna metoda koja se ispisuje pozdrav.
```

```
     */
```

```
    public static void sayHi() {
```

```
        System.out.println("Hi from Util!");
```

```
    }
```

```
}
```

Metoda
sayHi

“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: Eclipse

- ◆ Idemo ovo napraviti u Eclipse-u
- ◆ Potom pogledati nastalu strukturu direktorija
 - Vršni direktorij projekta
 - ◆ `src`
 - Poddirektoriji paketa i `*.java`
 - ◆ `bin`
 - Poddirektoriji paketa i `*.class`
 - ◆ `.classpath`
 - ◆ `.project`

“Hello World” program: Eclipse

- ◆ Paket `hr.fer.zemris.java.tecaj_1`
struktura direktorija na disku
`src/hr/fer/zemris/java/tecaj_1`
- ◆ Razred `HelloWorld`
datoteka na disku
`HelloWorld.java`
smještena u gornji direktorij
- ◆ Prevedeni kod je u direktoriju `bin` u istoj strukturi direktorija

“Hello World” program: ručno

- ◆ Idemo sada ručno iskompajlirati i pokrenuti ovaj program
 - Napravite negdje na disku novi vršni direktorij
 - U njemu napravite src i bin
 - U src napravite strukturu direktorija prema paketu i naše dvije *.java datoteke

“Hello World” program: ručno

```
playground$ mkdir helloworld
playground$ cd helloworld/
playground/helloworld$ mkdir -p src/hr/fer/zemris/java/tecaj_1
playground/helloworld$ gedit src/hr/fer/zemris/java/tecaj_1/HelloWorld.java
playground/helloworld$ gedit src/hr/fer/zemris/java/tecaj_1/Util.java
playground/helloworld$ mkdir bin
```

```
helloworld
├── bin
├── src
│   ├── hr
│   │   ├── fer
│   │   │   ├── zemris
│   │   │   │   ├── java
│   │   │   │   │   ├── tecaj_1
│   │   │   │   │   │   ├── HelloWorld.java
│   │   │   │   │   │   └── Util.java
```

“Hello World” program: ručno

- ◆ Budite u vršnom direktoriju projekta
- ◆ Prevođenje se pokreće naredbom
javac
- ◆ `javac -d bin
src/hr/fer/zemris/java/tecaj_1/HelloWorld.java
src/hr/fer/zemris/java/tecaj_1/Util.java`

Opcija -d navodi u koji se direktorij generiraju datoteke s izvršnim kodom (*.class) odnosno prikladna struktura direktorija. Programu javac daju se fizičke putanje do svih datoteka s izvornim kodom koje treba prevesti.

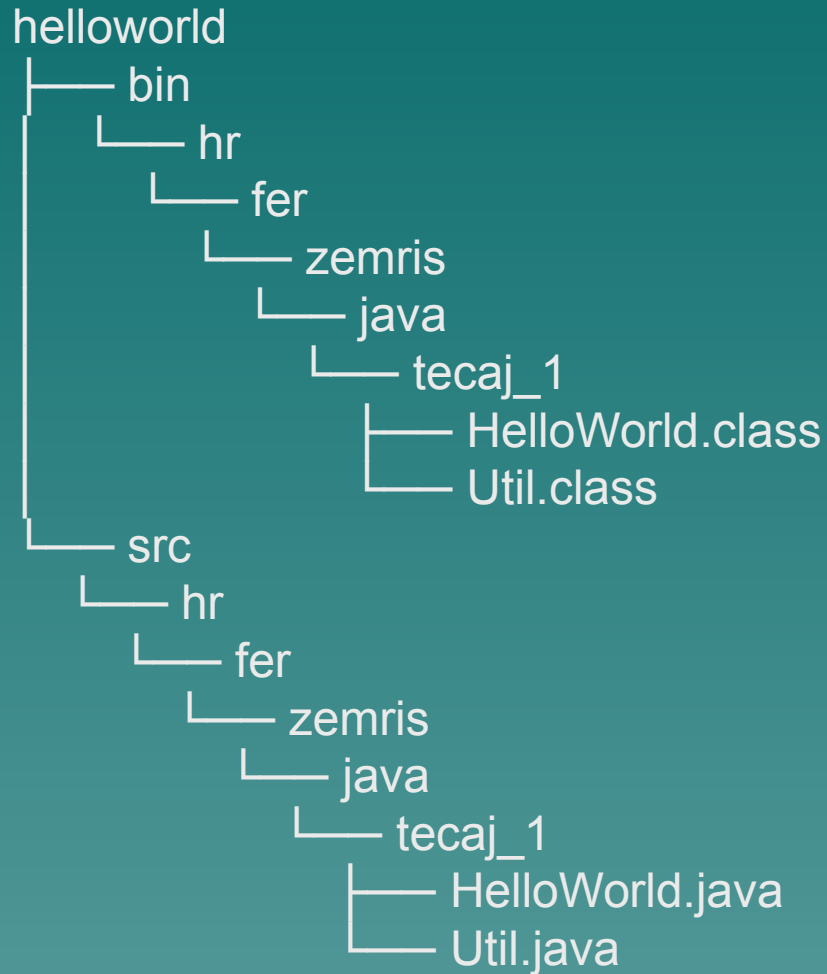
“Hello World” program: ručno

◆ Može i ovako:

- ◆ `javac -d bin -sourcepath src
src/hr/fer/zemris/java/tecaj_1/HelloWorld.java`

Sada smo naveli samo putanju do prvog razreda. Prevođenjem kompajler ustanovljava da mu treba i definicija od `hr.fer.zemris.java.tecaj_1.Util` odnosno datoteka `hr/fer/zemris/java/tecaj_1/Util.java`; argument `-sourcepath` mu kaže gdje da traži te datoteke.

“Hello World” program: ručno



“Hello World” program: ručno

- ◆ Budite u vršnom direktoriju projekta
- ◆ Izvođenje programa pokreće se naredbom `java`
- ◆

```
java -cp bin  
hr.fer.zemris.java.tecaj_1.HelloWorld
```

Opcija `-cp` (ili `-classpath`) navodi u kojim se direktorijima ili jar arhivama traže datoteke s izvršnim kodom (`*.class`). Programu `java` zadaje se logičko ime razreda koji sadrži metodu `main` nakon čega se mogu navesti argumenti koji se programu predaju preko naredbenog retka.

“Hello World” program: ručno

- ◆ Bez navođenja `-cp`, naredba `java`

- ◆ `java -cp bin
hr.fer.zemris.java.tecaj_1.HelloWorld`

bi pukla s greškom jer bi u trenutnom direktoriju tražila datoteku `hr/fer/zemris/java/tecaj_1/HelloWorld.class` koja ne postoji.

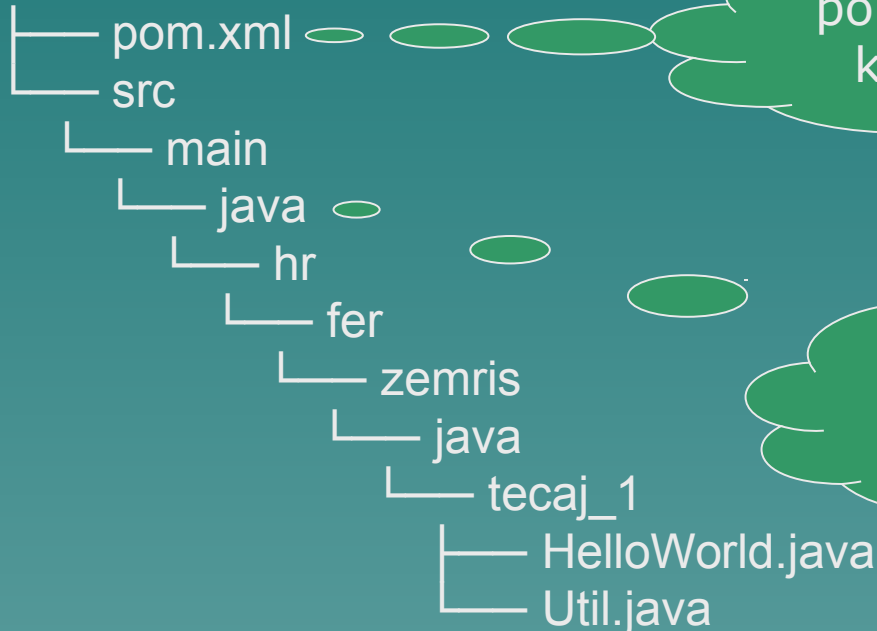
“Hello World” program: maven

- ◆ Maven je alat za automatizirano upravljanje cjelokupnim projektom
 - Za prevoditi, pokretati testove, pakirati kod u jar-arhive, ...
 - Omogućava jednostavno deklariranje ovisnosti o drugim bibliotekama i njihovo automatsko dohvaćanje

“Hello World” program: maven

- ◆ Maven definira standardnu strukturu direktorija za projekt

helloworld-maven/



pom.xml mora biti u
korijenu projekta

Izvorni kod ide u
src/main/java

“Hello World” program: maven

- ◆ Dohvatite primjer pom.xml datoteke <http://java.zemris.fer.hr/nastava/opjj/> (uzmite pom-osnovni.xml)
- ◆ Napravite potrebnu strukturu direktorija i prekopirajte unutra izvorne kodove

“Hello World” program: maven

- ◆ mvn compile
 - Prevodi program (u target/classes).
- ◆ mvn clean
 - Briše target (tj. sve što se generira).
- ◆ mvn package
 - Generira jar arhivu za projekt (u target).
- ◆ Isprobajte
- ◆ Izravno pokrenite prevedeni program

“Hello World” program

- ◆ Pokretanje programa – **main** metoda
public static void main(String[] args) {
...
}
- ◆ Argumenti iz naredbenog retka
– 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 \Leftrightarrow 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 preko naredbenog retka primiti dva argumenta (decimalna broja) te će ispisati njihovu sumu

Pomoć:

```
double x = 0;  
try {  
    x = Double.parseDouble("3.14");  
} catch (NumberFormatException ex) {  
    System.out.println("Niste unijeli broj!");  
    return;  
}
```


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) {
```

Svako polje
ima svojstvo
".length"

```
    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 + ",");
```

```
}
```

```
/**
 * Racuna e^x razvojem u Taylorov red, prema formuli:
 *  $e^x = 1 + x + \frac{x^2}{2!} + \frac{x^3}{3!} + \frac{x^4}{4!} + \dots$ 
 * @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 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 alokira 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);  
}
```

Uporaba "struktura" podataka

- ◆ U ovom primjeru tretirat ćemo Javu kao C-oliki jezik; baš kao što C ima `struct` za strukture, u Javi možemo koristiti `class` za simulaciju takvog ponašanja

Uporaba "struktura" podataka

- ◆ Tada umjesto C-ovskog alociranja:

```
struct x *var = (struct x*)malloc(  
    sizeof(struct x)  
);
```

u Javi pišemo:

```
x var = new x();
```

Uporaba "struktura" podataka

- ◆ Razmotrit ćemo jednostavan primjer izgradnje jednostavnog stoga
- ◆ Treba podržati mogućnost dodavanja na stog, ispitivanja je li stog prazan te mogućnost skidanja elementa sa stoga
- ◆ Prikazano rješenje nije u duhu OOP-a; Javu ovdje koristimo kao da je C

```
package hr.fer.zemris.java.tecaj_1;
public class DemoStoga {

    static class Zapis {
        Zapis stari;
        String vrijednost;
    }

    public static void main(String[] args) {
        Zapis stog = null;

        stog = dodaj(stog, "Ana");
        stog = dodaj(stog, "Ivana");
        stog = dodaj(stog, "Jasna");

        while(nijePrazan(stog)) {
            Zapis vrh = stog;
            stog = ukloni(stog);
            System.out.println("Uklonio sam ime: "+vrh.vrijednost);
        }
    }

    // još implementacije metoda: sljedeći slide...
}
```

```
public class DemoStoga {  
  
    // nastavak s prethodnog slidea:  
  
    static Zapis dodaj(Zapis stog, String ime) {  
        Zapis glava = new Zapis();  
        glava.vrijednost = ime;  
        glava.stari = stog;  
        return glava;  
    }  
  
    static boolean nijePrazan(Zapis stog) {  
        return stog != null;  
    }  
  
    static Zapis ukloni(Zapis stog) {  
        return stog.stari;  
    }  
}
```