INFORMATIČKI KLUB
FUTURA

RADIONICE OSNOVA PROGRAMIRANJA U JAVI

Java™

1. RADIONICA – UVOD U PROGRAMSKI JEZIK

Krunoslav Žubrinić, Informatički klub FUTURA

Dubrovnik, 26. veljače 2015.

Creative Commons

- **slobodno smijete:**
  - **dijeliti** — umnožavati, distribuirati i javnosti priopćavati djelo
  - **remiksirati** — prerađivati djelo
- **pod slijedećim uvjetima:**
  - **imenovanje**. Morate priznati i označiti autorstvo djela na način kako je specificirao autor ili davatelj licence (ali ne način koji bi sugerirao da Vi ili Vaše korištenje njegova djela imate njegovu izravnu podršku).
  - **nekomercijalno**. Ovo djelo ne smijete koristiti u komercijalne svrhe.
  - **dijeli pod istim uvjetima**. Ako ovo djelo izmijenite, preoblikujete ili stvarate koristeći ga, preradu možete distribuirati samo pod licencom koja je ista ili slična ovoj.

U slučaju daljnjeg korištenja ili distribuiranja morate drugima jasno dati do znanja licencne uvjete ovog djela. Najbolji način da to učinite je linkom na ovu internetsku stranicu.

Od svakog od gornjih uvjeta moguće je odstupiti, ako dobijete dopuštenje nositelja autorskog prava.

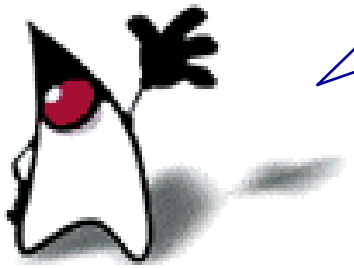
Ništa u ovoj licenci ne narušava ili ograničava autorova moralna prava.

Tekst licence preuzet je s http://creativecommons.org/.

Sadržaj

- Vrlo kratka povijest Jave
- Struktura Java programa
- Prvi Java program
- Ispis podataka na zaslon
- Varijable
- Razredi i objekti
- Unos podataka s tipkovnice
- Osnovni operatori
- Programske strukture

Ja sam **Duke**, maskota Java tehnologije!



Java

- programski jezik temeljen na ideji da se isti program bez ikakvim izmjena izvodi na računalima različitih platformi i operacijskih sustava
  - osobnim računalima (MS Windows PC, Linux, AIX, Mac,...), ručnim računalima, GSM uređajima, televizorima, Smart karticama,...
  - napisani program prevodi se u "*bytecode*"
    - interpreter (*Java Runtime Environment* - **JRE**) na odgovarajućoj platformi izvodi *bytecode*

Povijest

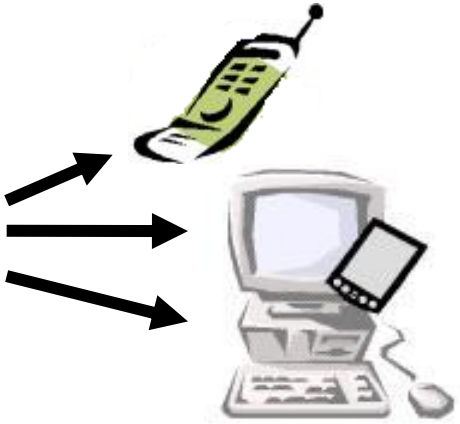
- Sun Microsystems - "*Green Project*" (1991/92.)
  - razvoj ručnih računala za kontrolu kućnih elektroničkih uređaja (TV, video, telefon,...)
  - stvorili su novi programski jezik **Oak**, a od C++ su preuzeli OO orijentaciju i osnove sintakse
- Razvoj Interneta - potreba za interaktivnošću stranica
  - 1993. WWW prelazi iz tekstualnog u grafičko okruženje
  - Java je idealna za korištenje u WEB okruženju jer ne ovisi niti o jednoj platformi
  - 1995. jezik službeno mijenja ime u **Java**
  - tehnologija se širi (danas na gotovo svim značajnijim platformama postoji mogućnost izvođenja programa u Javi

- Razlikujemo
  - Java platformu (**Java Runtime Environment**)
  - Programski jezik Java

Način na koji se radi Java program

```
public class DobarDan {

      public static void
      main(String[] args) {
      System.out.println
      ("Dobar dan!") ;
  }
      …
}
```

```
method main()
0 aload_ 0
invokespecial#1 <method
java.lang.Object()>
return
```



1. ULAZ

Izvorni kod Programa (**.java** datoteka)

2. PREVOĐENJE

Program prevoditelj vrši prevođenje izvornog koda u međukod (*bytecode*).

Prevoditelj (javac.exe) je dio *Java Development Kita* – **JDK**)

3. IZLAZ

Prevoditelj kreira međukod. Svaki uređaj koji može interpretirati ili prevesti **JAVA** kod može izvesti JAVA program. (**.class** datoteka)

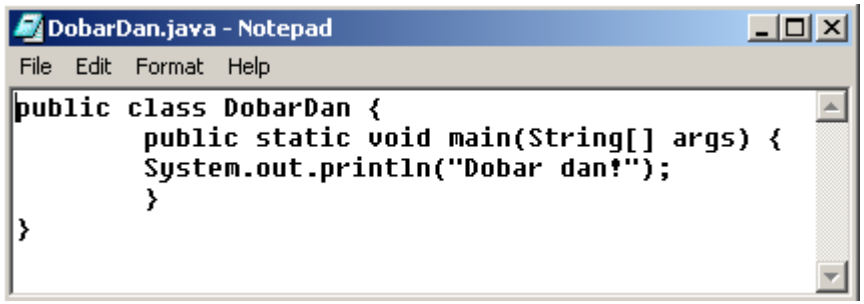
4. IZVOĐENJE

U svakom uređaju koji može izvesti **JAVA** program implementiran je **JAVA** virtualni Stroj (*Java Runtime Environment* – **JRE**)

Izrada Java programa

1. napišete Java program u tekst editoru.
   - spremite ga s ekstenzijom **.java**

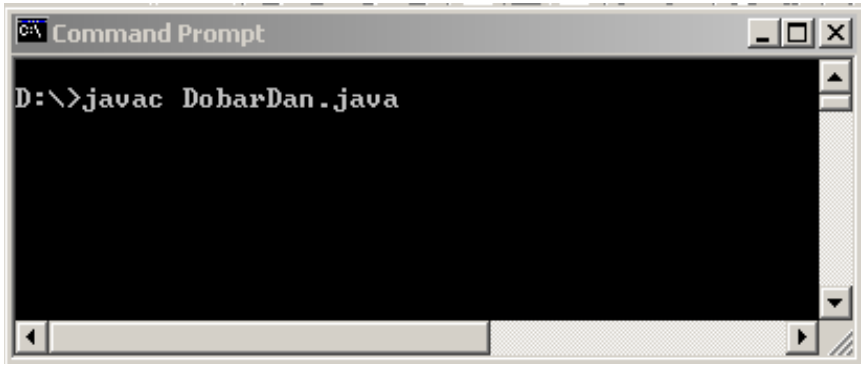
```
ImePrograma.java
```



2. prevedete program u međukod (*bytecode*)

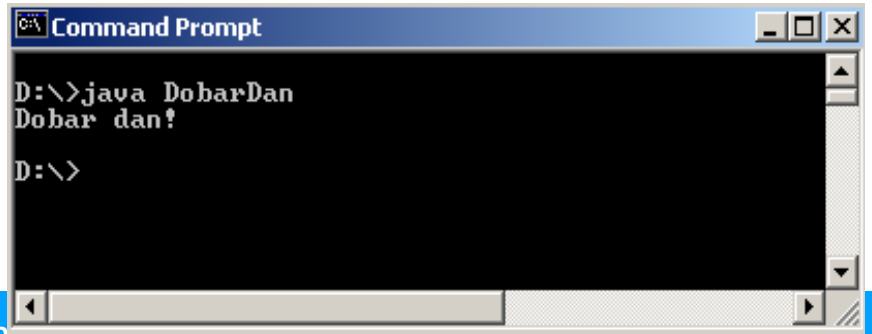
```
javac ImePrograma.java
```

- nastaje datoteka s ekstenzijom **.class**



3. izvedete program (bez navođenja ekstenzije **.class**)

```
java ImePrograma
```



Što vam treba za **izradu** Java programa

- Java Development Kit - JDK

http://www.oracle.com/technetwork/java/javase/downloads/index.html

Upute za instalaciju JDK-a

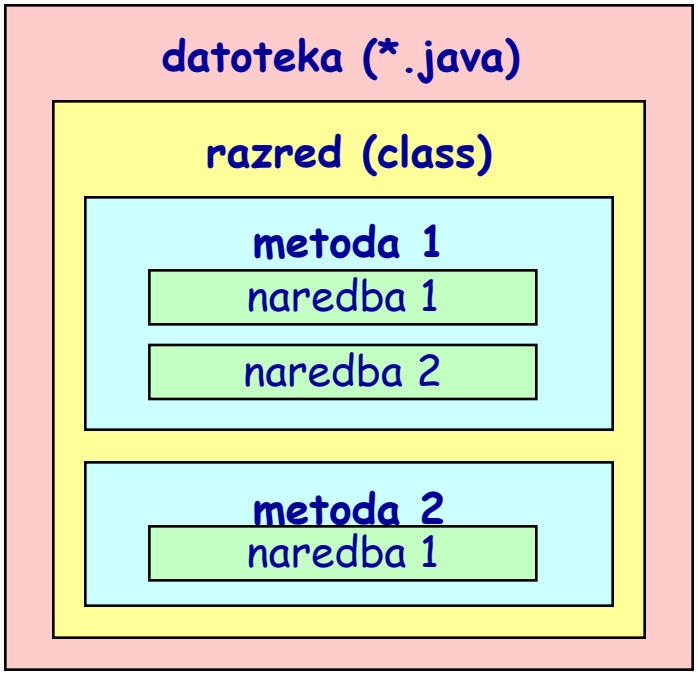
- Program možete pisati u bilo kojem editoru teksta (notepad, notepad++….)

- Zbog jednostavnosti, za pisanje programa na ovim radionicama ćemo koristiti **Eclipse** razvojno okruženje

https://eclipse.org/downloads/

Upute za instalaciju Eclipse

Struktura Java programa



- **razredi** se smještaju u **datoteku**
- **sve** je smješteno unutar **razreda**
- **metode** se ugnježđuju unutar razreda
- **naredbe** se smještaju unutar metoda

- razred

```
class Prijatelj
{
}
```

- metoda

```
class Prijatelj
{
    void pozdravi()
    {
    }
}
```

- naredbe

```
class Prijatelj
{
    void pozdravi()
    {
        naredba 1;
        naredba 2;
    }
}
```

Preporuke u pisanju koda

- nazivi razreda pišu se tzv. **Pascal kodom** (prvo slovo svake riječi je veliko)

PrviRazred

- nazivi metoda i varijabli pišu se **Camel kodom** (prvo slovo prve riječi je malo, a prva slova ostalih riječi su velika)

```
ovoJeMetoda()
```

Metode: prva riječ glagol

```
izračunProsjeka()
```

Varijable: prva riječ imenica

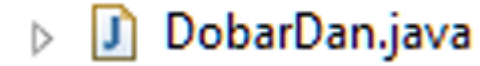
```
prosjekPlace
```

Prvi Java program

Datoteka: **DobarDan**.java

```
public class DobarDan
{
  public static void main(String[] args)
   {
        System.out.println ("Dobar dan!") ;
   }
}
```

```
Dobar dan!
```



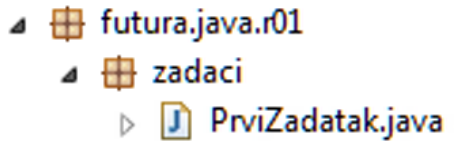
Izrada prvog Java programa

Demonstracija…

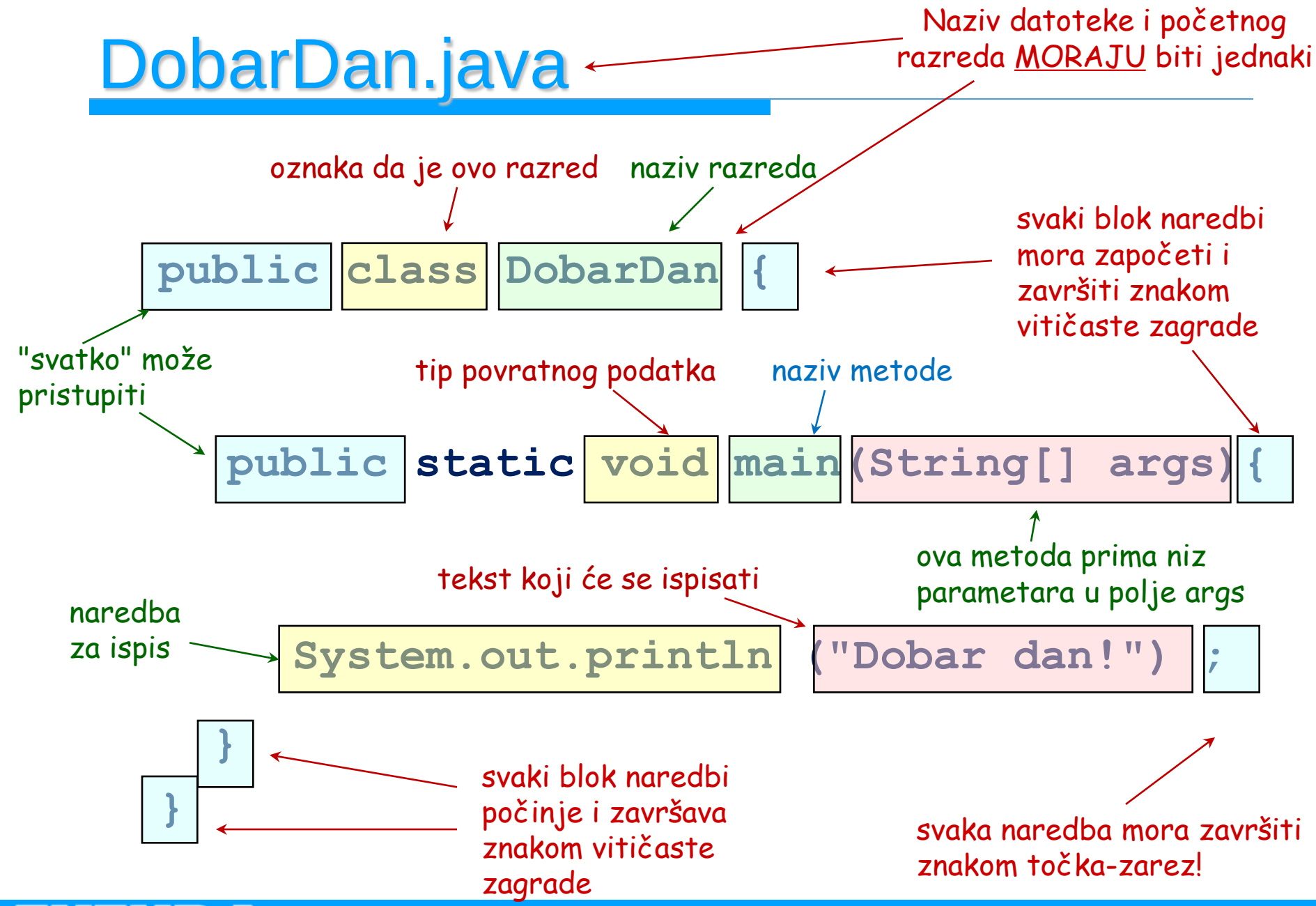
http://www.futura.com.hr/materijali/java-2015/java-01.html#_izrada_prvog_java_programa

1. zadatak

- Pokrenite Eclipse i kreirajte razred PrviZadatak
- Prepišite tekst programa DobarDan.java
  - Promijenite naziv klase u **PrviZadatak**
  - Pazite na ekstenziju datoteke (mora biti **.java**)
  - U program dodajte još jedan redak u kojem na zaslon ispišite tekst "**Danas je četvrtak.**"
- Izvedite nastali program (pozovite **main** metodu)
  - Tijekom izvođenja **Eclipse** automatski prevodi napisani program



DobarDan.java



Osobine jezika

- svaki program je razred (*class*)
  - kada se izvodi program, izvodi se određeni razred (.class datoteka)
  - pri izvođenju, na temelju razreda se stvara konkretan objekt
- metoda main() pokreće program i svaki program koji se želi izravno izvesti mora imati jednu main() metodu
- jezik je "*case sensitive*" (IME<> ime)
- na kraju retka se obavezno navodi točka-zarez
- svaki blok naredbi mora biti omeđen znakovima vitičastih zagrada { }
- varijable prije korištenja treba deklarirati (odrediti naziv i tip varijable)

Ispis podataka na zaslon

- za ispis na zaslon koriste se metode:

```
System.out.println(izraz);
System.out.print(izraz);
System.out.printf(izraz);
```

Sintaksa **printf** metode identična je sintaksi printf funkcije u C-u

Ispis podataka na zaslon

- svaki ispis ide u novi red

```
System.out.println("Danas je četvrtak");
System.out.println("26.veljače 2015.");
```

```
Danas je četvrtak
26. veljače 2015.
```

- ispis u isti red

```
System.out.print("Danas je četvrtak ");
System.out.print("26. veljače 2015.");
```

```
Danas je četvrtak 26. veljače 2015.
```

PrimjeriIspisa.java

Ispis podataka na zaslon

- formatirani ispis podataka

```
System.out.println("PI je: "+ Math.PI);
System.out.printf("PI na 3 decimale: %.3f", Math.PI);
```

- način formatiranja i oznake su jako slični sintaksi kod printf C funkcije
  - **%*n*d** – ispis cijelog broja (širine ***n*** znamenki)
  - **%.*n*f** – ispis broja na ***n*** decimala

```
PI je 3.141592653589793
PI na 3 decimale je 3.142
```

PrimjeriIspisa.java

Escape sekvence

- posebni znakovi koji formatiraju ispis
  - `\n` – prijelaz na novu liniju
  - `\t` – prijelaz na sljedeću poziciju tabulatora
  - `\\` - prikaz znaka `\`
  - `\"` – prikaz znaka navodnika `"`

```
System.out.print("Danas je tako \"kišno\" vrijeme!\n");
System.out.println("Stvarno je \t \\kišno\\!");
```

```
Danas je tako "kišno" vrijeme!
Stvarno je       \kišno\!
```

PrimjeriIspisa.java

Varijable

- služe za pohranu podataka
- imaju **ime**, **tip** i **područje djelovanja** (*scope*)
- **osnovni** tipovi podataka:

Cjelobrojni

byte (2^8)
(od -128 do +127)

short (2^{16})
(od -32.768 do +32.767)

int (2^{32})
(od -2.147.483.648 do +2.147.483.647)

long (2^{64})
(od -9.223.372.036.854.775.808 do +9.223.372.036.854.775.807)

Realni

float (32 bitni) npr. **3.7f**

double (64 bitni) npr. **3.7**

Znakovni

char (1 znak) npr. **'a'**

Logički

boolean (**true**/**false**)

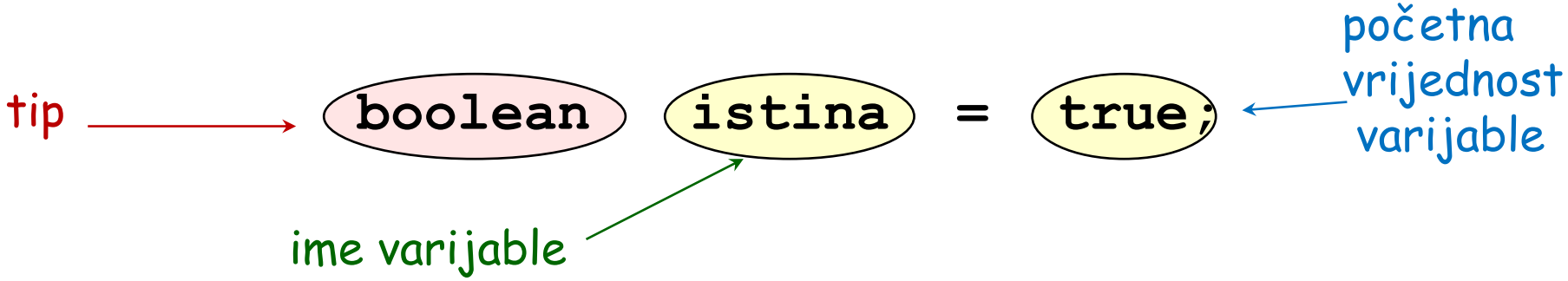
String

Za pohranu **niza znakova**. Nije osnovni tip (**veliko slovo S**!), ali se često koristi pa je način njegovog korištenja isti kao kod osnovnih tipova podataka.

Varijable

- deklariraju se tipom i imenom

- mogu se inicijalizirati odmah prilikom deklariranja



Imena varijabli u Javi

- moraju početi:
  - slovom,
  - podcrtom (_),
  - ili znakom dolara ($)
- znakovi koji slijede mogu biti bilo koji
- ime varijable ne smije biti rezervirana riječ u Javi

```
int -visina;
float 3polumjer;
String #osoba;
double while;
```

```
int visina35;
float _polumjer;
String $osoba;
```

Operatori

aritmetički operatori

+	a+b	zbrajanje
-	a-b	oduzimanje
*	a*b	množenje
/	a/b	dijeljenje
%	a%b	modulo
++	a++	inkrementiranje
--	a--	dekrementiranje

relacijski operatori

<	a<b	manje od
>	a>b	veće od
<=	a<=b	manje ili jednako
>=	a>=b	veće ili jednako
!=	a!=b	različito
==	a==b	jednako

operatori uvjeta

&&	a&&b	logičko I
||	a||b	logičko ILI
!	!a	logičko NE

operatori pridruživanja

+=	a+=b	a=a+b
-=	a-=b	a=a-b
*=	a*=b	a=a*b
/=	a/=b	a=a/b
%=	a%=b	a=a%b

znakovni operator

+	a+b	spajanje Stringova

Pridruživanje vrijednosti varijabli

- treba obratiti pažnju na veličinu varijabli
  - vrijednost varijable manjeg tipa može se smjestiti u veće, ali većeg tipa ne može u manju

```
int x = 24;
byte y = x;
```

```
byte x = 5;
int y = x;
```

Korištenje varijabli

- varijablu **TREBA** incijalizirati prije prvog korištenja

```
int suma;

suma = suma + 2;
```

```
int suma = 0;

suma = suma + 2;
```

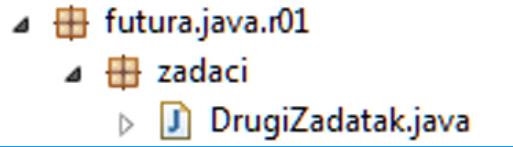
2. zadatak

- napišite Java program koji će pomoću tri naredbe za ispis (print, println ili printf) ispisati podatke u obliku prikazanom na slici.
  - Za ispis broja koristite konstantu **Math.PI** i ispišite njezinu vrijednost na dvije decimale
  - Za formatirani prikaz broja na dvije decimale koristite metodu

```
printf("… %.2\n", Math.PI)
```

- program prevedite i izvedite

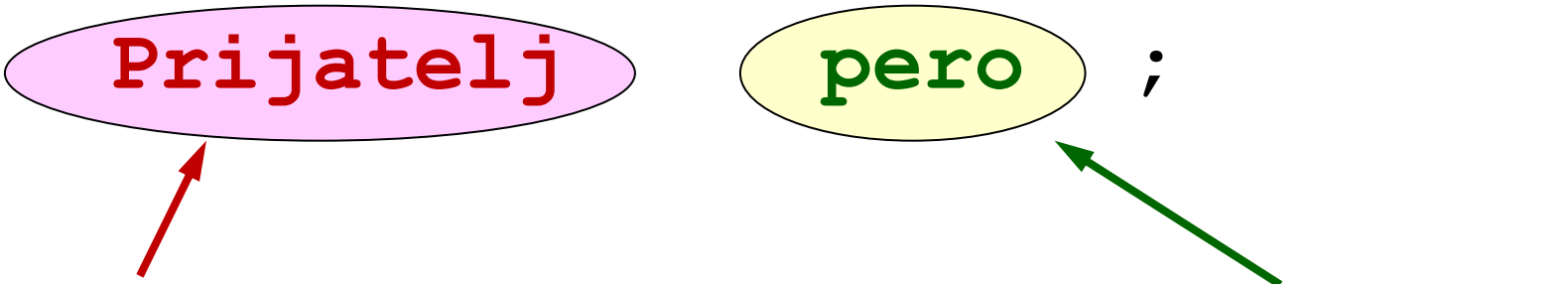
```
x
xx
xxx
xxxx
PI na 2 decimale je: 3.14
On je rekao "Dobar dan"!
```



Varijable

osnovni tipovi varijabli sadrže konkretne podatke (int, char, double,…)

- tip podatka ne mora biti osnovni



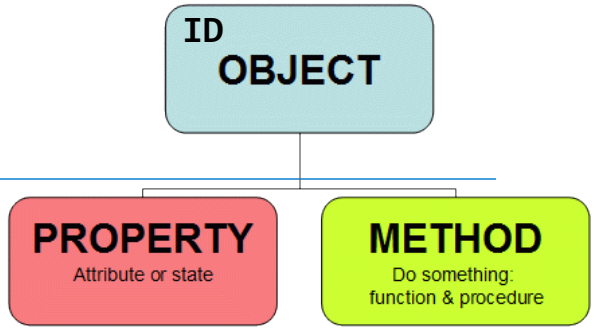
tip varijable u ovom slučaju je **razred** Prijatelj

ime varijable

Varijabla je u ovom slučaju **referenca** odnosno **pokazivač na objekt tipa Prijatelj**

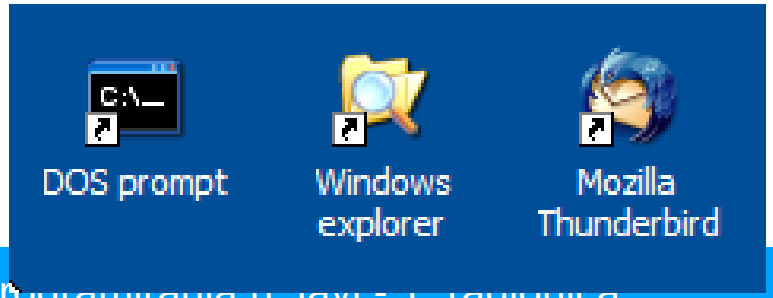
reference sadrže pokazivač na mjesto u memoriji na kojem se nalazi **objekt** odgovarajućeg tipa (***String***, ***Array***, ***Prijatelj***, …)

Objekt



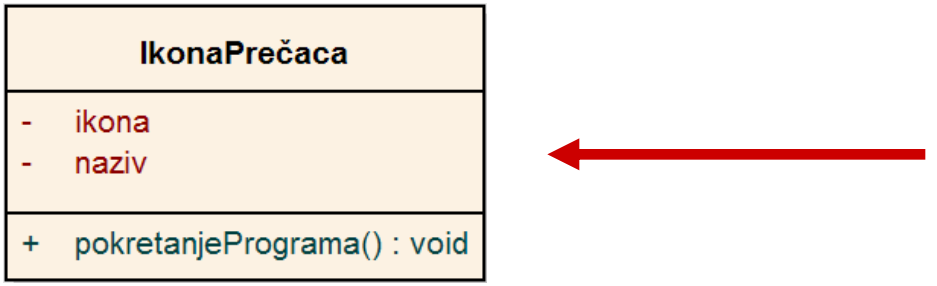
- sadrži **podatke**
  - **varijable** u programu predstavljaju stanje objekta
- ima **skup operacija za rad s podacima** (poruka koje mogu razmjenjivati s drugim objektima)
  - **metode** – funkcije koje pripadaju objektu i mijenjaju stanje
- ima svoj **identitet** (jedan objekt se jasno razlikuje od drugog)

različite ikone prečaca



Razred (*class*)

- shema ili obrazac za kreiranje objekata
- određuje tip objekata
- definira:
  - koje podatke će objekti (tog tipa) imati (**varijable**)
  - koje operacije će objekti (tog tipa) izvršavati (**metode**)
- programiranje u Javi je pisanje razreda (klasa)



u praksi ne postoji nešto što se zove "ikona prečaca", međutim svaka konkretna ikona ima neke elemente.

Razred (class)

ključna riječ class označava razred

naziv razreda

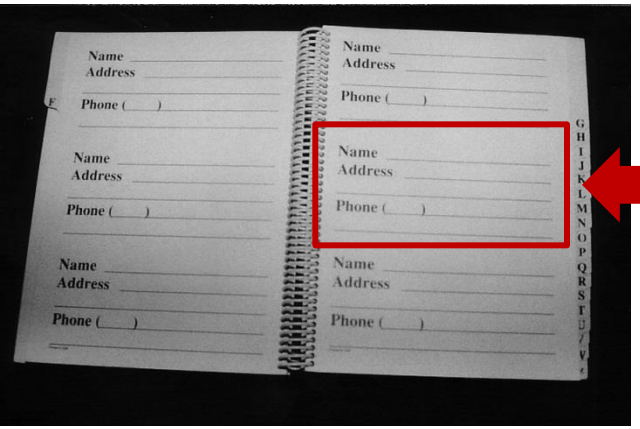
```
public class Prijatelj
{
    String ime;
    int starost = 0;
    void rodjendan(){
       starost = starost + 1;
    }
}
```

sve unutar vitičastih zagrada pripada razredu

Prijatelj.java

- unutar jedne java datoteke može biti više razreda (class), ali samo jedna može biti dostupna "izvana" (public)

Objekti ⇨ Klasa



kontakt podaci = **klasa**

Ime i prezime:
Adresa:
Telefon:

operacije s kontaktima = **metode**

Ime i prezime: Ana Anić
Adresa: I. Vojnovića 2, Dubrovnik
Telefon: 020/277-110

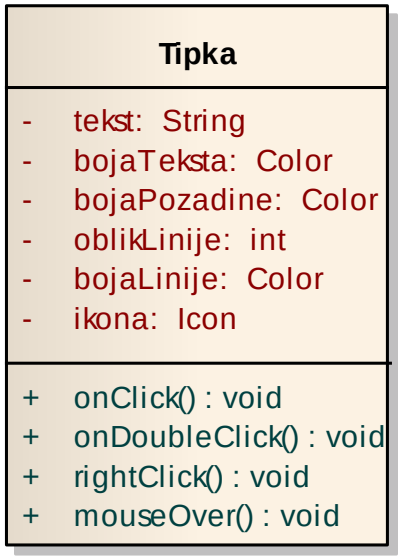
Ime i prezime: Pero Perić
Adresa: Vukovarska 67 Dubrovnik
Telefon: 020/221-112

instance klase = **objekti**

Objekti ⇨ Klasa

Elementi grafičkog sučelja

Objekti



Klasa

Elementi tipke i akcije koje su dozvoljene s tipkom

Stvaranje objekta (u 3 koraka)

1. Deklarira se varijabla referenca

```
Prijatelj susjed = new Prijatelj();
```

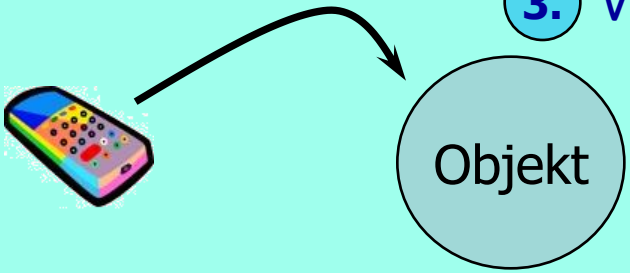
osigurava se prostor za varijablu referencu koja omogućuje pristup elementima objekta tipa **Prijatelj**

2. Stvara se objekt u memoriji

```
Prijatelj susjed = new Prijatelj();
```

osigurava se prostor za novi (***new***) objekt

3. Varijabla referenca se povezuje sa objektom




```
Prijatelj susjed = new Prijatelj();
```

omogućuje se pristup objektu pomoću varijable reference (povezivanjem varijable s adresom stvorenog objekta)

Stvaranje i korištenje objekata

```
public class Upoznavanje {
   public static void main(String[] args) {

      Prijatelj susjed;

      susjed = new Prijatelj();
      // možemo i kraće u 1 retku
      // Prijatelj susjed = new Prijatelj();

      susjed.ime = "Ivo";
      susjed.rodjendan();

      System.out.println("Dobar dan, ja sam " +
susjed.ime);
      }
}
```

deklariramo varijablu susjed koja je referenca na razred Prijatelj

povezujemo objekt i referencu

kreiramo objekt susjed pomoću ključne riječi new i konstruktora Prijatelj()

točka je separator pristupa članovima objekta

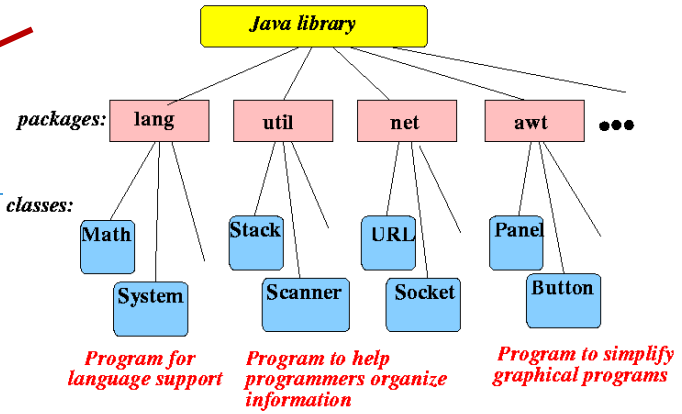
Upoznavanje.java

Stvaranje i korištenje objekata

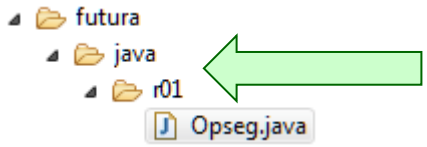
- možemo stvoriti onoliko objekata koliko nam treba

```
Prijatelj desniSusjed = new Prijatelj();
desniSusjed.ime = "Maro";

Prijatelj lijeviSusjed = new Prijatelj();
lijeviSusjed.ime = "Baro";

Prijatelj kolegicaIzSkole = new Prijatelj();
kolegicaIzSkole.ime = "Mare";
kolegicaIzSkole.rodjendan();


Prijatelj prijateljicaIzVrtica = new Prijatelj();
prijateljicaIzVrtica.ime = "Kate";
...
```

Paketi



- **paketi** (*packages*) grupiraju razrede po funkcionalnosti
  - služe za rješavanje problema "sukoba imena"
  - imenovani su hijerarhijski, primjerice **java.util**
  - I vaše klase možete uključiti u paket




```
package futura.java.r01;
import java.io.File;
import java.util.*;
public class Opseg {

    ...

}
```

Ako vaš razred uključujete u paket, to činite navođenjem ključne riječi **package**. To mora biti prva naredba u programu.

Paketi se uključuju pomoći ključne riječi **import** prije prvog razreda.

Ako uključujemo sve Razrede unutar izabranog paketa stavljamo *, inače navodimo naziv razreda koji uključujemo.

Unos podataka s tipkovnice

- Koriste se metode objekata tipa **Scanner**
- Programski kod razreda **Scanner** se nalazi u **java.util** paketu

1. Moramo **uključiti razrede** paketa **java.util**

```
import java.util.*;
```

2. Moramo **stvoriti novi objekt** tipa **Scanner**

```
Scanner unos = new Scanner(System.in);
```

3. Moramo pozivati metode **na stvorenom objektu**

```
String ime = unos.next();
String imePrezime = unos.nextLine();
int starost = unos.nextInt();
float polumjer = unos.nextFloat();
```

4. Kada nam više ne treba, prekidamo unos

```
unos.close();
```

http://docs.oracle.com/javase/7/docs/api/java/util/package-summary.html

Primjer unosa

Razred spremamo u paket futura.java.r01

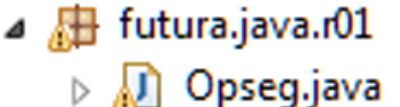
paket **java.util** koji sadrži razred **Scanner** (i mnoge druge)

```
package futura.java.r01;
import java.util.*;
public class Opseg {
   public static void main(String[] args) {
      System.out.println("Unesite duljinu:");
      Scanner sc = new Scanner(System.in);
      int i = sc.nextInt();

      System.out.println("Opseg kvadrata je ");
      System.out.print(i*4);
      sc.close();
   }
}
```

Stvara se objekt za unos podataka s komandne linije (*System.in*)

s komandne linije se unosi podatak tipa *int*

Završetak unosa



Pretvaranje tipova podataka

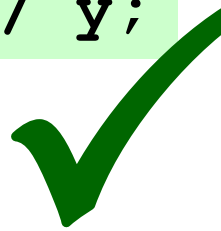
- Osnovne tipove podataka možete međusobno pretvarati, kada je potrebno
  - ***casting*** osnovnih tipova

```
int x=24, y=5;
float z;
z = x / y;
```

```
4.0
```

```
int x=24, y=5;
float z;
z = (float) x / y;
```

```
4.8
```



Casting: unutar okruglih zagrada navodi se tip u koji želimo pretvoriti podatak

```
int r = 7;
polumjer = (float) r;
```

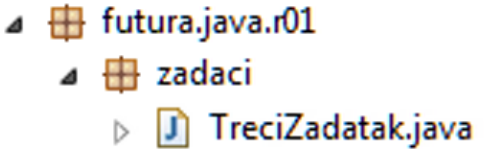
- ako je vrijednost veća od varijable u nju će se smjestiti neispravna vrijednost, ali program neće javiti grešku!

3. zadatak

- Deklarirajte u programu 2 **cjelobrojne** varijable: jednu za osnovicu, a drugu za stopu poreza
- Pridružite vrijednosti varijablama pomoću metoda na objektu tipa **Scanner**
- Deklarirajte u programu varijablu za pohranu iznosa poreza
  - Pazite na **tip varijable** (jesu li za prikaz kvocjenta dovoljne **cjelobrojne** varijable? Treba li kod izračuna pretvarati cjelobrojne varijable u drugi tip?)
  - Kod ispisa iznose prikažite na dvije decimale

$$porez = \frac{osnovica * stopa}{100}$$

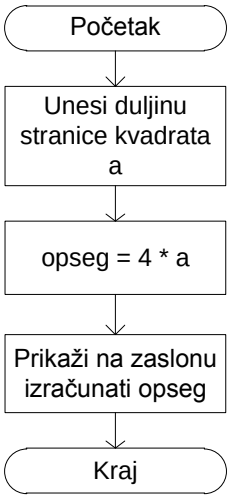
```
Unesite osnovicu poreza:
120
Unesite stopu poreza:
23
Osnovica je 120.00, stopa je 23.00, a porez je 27.60
```

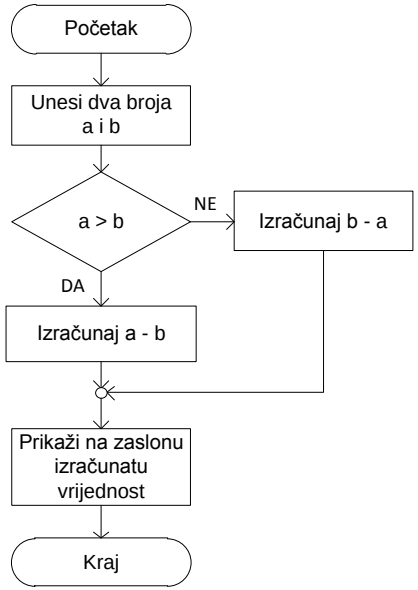


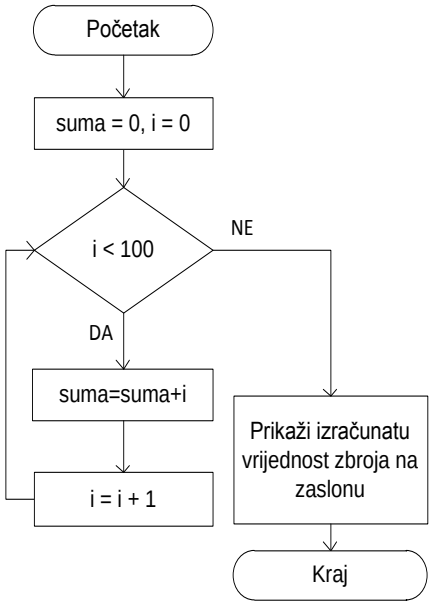
Programske strukture

- Opisuju način i redosljed izvršavanja manjih zadataka.
- Tri osnovne vrste programskih struktura:

1. Slijed instrukcija **2. Grananje** `if … else` **3. Petlja** `for …` `while …`







Grananje (if...else)

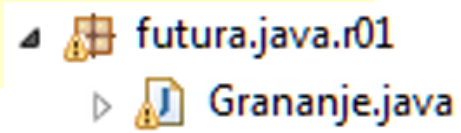
```
if (uvjet1){
     ...naredbe...
}
else if (uvjet2) {
     ...naredbe...
}
else {
     ...naredbe...
}
```

Podatak logičkog tipa **boolean**

- omogućuje uvjetno izvođenje **if** dijela programa ako je zadovoljen odgovarajući **uvjet**
- **else** grana se izvršava ako niti jedan od prethodnih logičkih uvjeta nije zadovoljen

Primjer: Grananje (if...else)

```
int starost;
…
if (starost < 13)
   System.out.println("Dijete");
else if ((starost > 12) && (starost < 20))
   System.out.println("Tinejdžer");
else if ((starost > 19) && (starost < 41))
   System.out.println("Mlada osoba");
else if ((starost > 40) && (starost < 61))
   System.out.println("Srednje stara osoba");
else
   System.out.println("Stara osoba");

…
```



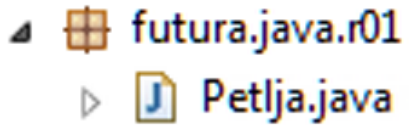
Petlja (for...)

```
for (inicijalizacija;
   uvjet_izvođenja; ažuriranje){
   ...niz naredbi...
}
```

- **inicijalizacija** se izvodi jednom, na početku petlje - inicijalizira varijablu brojača (npr. int i=0;)
- na početku izvođenja svake iteracije provjerava se **uvjet izvođenja** (logički izraz). Ako je uvjet istinit, izvođenje se nastavlja, inače se prekida (npr. i<10;)
- **na kraju svake iteracije** izvodi se ažuriranje vrijednosti brojača (npr. i++; ili i=i+3;)
- **for** petlja se koristi kada znamo koliko puta se naredbe trebaju izvesti

Primjer: for petlja

```
…
for (int i = 0; i < n; i++){
    if (i % 2 == 0)
       System.out.println(i + " je paran broj.");
    else
       System.out.println(i + " je neparan broj.");
}
…
```

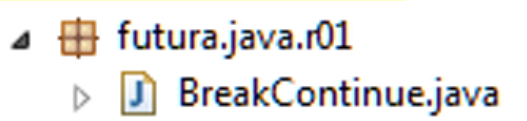


break i continue

- **break** prekida izvođenje petlje
  - izvođenje se nastavlja na prvoj naredbi nakon petlje
- **continue** vraća izvođenje na početak petlje
  - brojač petlje povećava vrijednost
- koriste se kada je potrebno izvođenje programa preusmjeriti u odnosu na "normalno"
- koriste se kod bilo koje vrste petlje

Primjer: break i continue

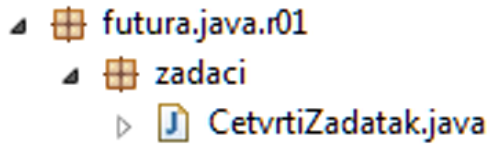
```
…
Scanner s = new Scanner(System.in);
int broj = 0;

for (;;;){ // beskonačna petlja
    broj = s.nextInt();
// ako je unesena nula ili negativan broj, odmah prekini
// izvođenje petlje
    if (broj <= 0)
        break;
// ako je unesen troznamenkasti ili veći broj
// nemoj ga ispisivati
    else if (broj > 99)
        continue;
    else
// inače, ispiši broj (svaki jedno i dvoznamenkasti broj)
        System.out.println(broj);
}
```



4. zadatak

- Pomoću metoda stvorenog objekta tipa Scanner unesite s tipkovnice cijeli broj u intervalu 10-1.000
  - Ako uneseni broj nije u zadanom intervalu, ispišite poruku i prekinite izvođenje programa
  - Ako je broj u zadanom intervalu, napišite program koji će izračunati i na ekranu prikazati zbroj svih parnih brojeva od 1 do unesenog broja (uključujući taj broj).

```
Unesite cijeli broj u intervalu 10-1000:
100
Zbroj parnih brojeva od 1 do 100 je 2550
```



5. zadatak

- Pomoću metoda stvorenog objekta tipa Scanner unesite cijeli broj u intervalu 5-15
- Ako uneseni broj nije u zadanom intervalu, ispišite poruku i prekinite izvođenje programa
- Ako je uneseni broj u redu napišite program koji će u tabličnom obliku prikazati tablicu množenja svih **neparnih** brojeva od 1 do unesenog broja

```
Unesite cijeli broj u intervalu 5-15:
11
  1   3   5   7   9  11
  3   9  15  21  27  33
  5  15  25  35  45  55
  7  21  35  49  63  77
  9  27  45  63  81  99
 11  33  55  77  99 121
```