



Programiranje I – RIN Računalništvo I – MA

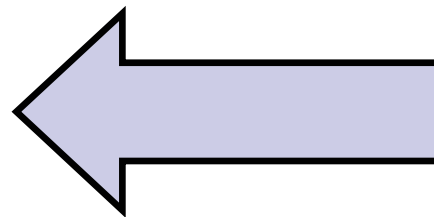
OSNOVNI GRADNIKI PROGRAMSKIH JEZIKOV 2. DEL

3. predavanje

Vsebina – razredi in predmeti

■ Razredi

- Deklaracija – zgradba
- Spremenljivke – lastnosti
- Funkcije – metode
- Konstruktorji



**Podajanje
vhodnih vrednosti**

■ Predmeti

- Življenjski cikel predmeta:
 - Kako predmet ustvarimo, uporabljamo, uničimo?
 - Garbage collector (JAVA)

■ Primeri

Predmeti oziroma stvari

- Predmeti se razlikujejo po svojih lastnostih
- Lastnosti tudi določajo, kaj lahko s predmeti **počnemo** - imamo **metode** za *spreminjanje* lastnosti in za *poizvedovanje* o lastnostih
- Predmeti (*objects*):
 - hranijo **podatke**: vrednosti lastnosti; npr. lastnost *barva* ima lahko vrednost *zelena*, *modra*, *rdeča*, ...
 - nudijo **metode** za delo s temi podatki; npr. *pobarvaj*

Razredi

- Predmete z enakimi lastnostmi združujemo v razrede: **razred je** (približno povedano) **množica predmetov z enakimi lastnostmi**
 - vsi predmeti v razredu imajo enake lastnosti; npr. pri vseh lahko govorimo o barvi
 - nimajo pa vsi predmeti v razredu nujno enako vrednost te lastnosti; npr. ni nujno, da so vsi zeleni

Predmeti

- Pri programiranju najprej definiramo (opišemo) lastnosti razreda predmetov – ***definiramo razred***

```
public class Trikotnik { ... }
```

- Nato ***tvorimo*** (*instantiate*) enega predstavnika razreda - naredimo en predmet

- v Javi to naredimo z ukazom **new**:

```
predmet = new Razred (parametri);
```

```
abc = new Trikotnik (3, 4, 5);
```

Predmeti

- Predmet ima svoj življenjski cikel:
 - se rodi (ga tvorimo)
 - o njem lahko proizvedujemo ali ga spreminjamo (ga uporabljamo)
 - na koncu premine (ga uničimo)

Razredi

- Pri definiciji razreda metode združujemo po skupinah:
 - tvorjenje / uničevanje (*create / destroy*)
 - poizvedovanje (*query*)
 - spreminjanje (*update*)
 - (zasebne) lastnosti in metode (*private*)

Posebna metoda `main`

- V definiciji razreda lahko obstaja metoda `main`
- Metoda `main` se požene kot prva, predno se tvori katerikoli predmet iz tega razreda
 - Primer uporabe: kot glavna metoda našega programa
- Obstajajo tudi druge posebne metode (`finalize`, itd.)

Primer razreda – **Trikotnik**

- Definirali bomo razred **Trikotnik**, ki bo imel lastnosti:
 - ob tvorjenju bomo trikotniku določili dolžine stranic, ki jih bomo lahko kasneje spreminjali
 - poznan bo njegov obseg
 - poznano bo, koliko trikotnikov je bilo narejenih (obstaja)

Primer razreda – trikotnik

```
public class Trikotnik {  
    protected int a, b, c; // stranice tega (this) trikotnika  
    static protected  
    int stTrikotnikov= 0; // stev. narejenih trikotnikov
```

- tvorjenje / uničevanje
- poizvedovanje
- spreminjanje
- (zasebne) lastnosti in metode

Primer razreda – trikotnik

```

/* ----- */
/* -----[ tvorjenje / unicevanje ]--- */
public Trikotnik(int an, int bn, int cn) {
    a= an; b= bn; c= cn; stTrikotnikov++;
}; // (tvoritelj) trikotnik
public Trikotnik() {
    a= b= c= 0; stTrikotnikov++;
}; // trikotnik
    
```

- ob tvorjenju bomo trikotniku določili dolžine stranic, ki jih bomo lahko kasneje spreminjali
- poznan bo njegov obseg
- poznano bo, koliko trikotnikov je bilo narejenih (obstaja)

–**tvorjenje / uničevanje**

–poizvedovanje

–spreminjanje

–(zasebne) lastnosti in metode

Primer razreda – trikotnik

```

/* ----- */
/* -----[ poizvedovanje ]--- */
public int Obseg() { return a+b+c; };
public int narejenihTrikotnikov() {
    return stTrikotnikov;
}; // narejenihTrikotnikov
    
```

- ob tvorjenju bomo trikotniku določili dolžine stranic, ki jih bomo lahko kasneje spreminjali
- **poznan bo njegov obseg**
- **poznano bo, koliko trikotnikov je bilo narejenih (obstaja)**

- tvorjenje / uničevanje
- **poizvedovanje**
- spreminjanje
- **(zasebne) lastnosti in metode**

Primer razreda – trikotnik

```

/* ----- */
/* -----[ spreminjanje ]--- */
public void aNajBo(int noviA) { a= noviA; };
public void bNajBo(int noviB) { b= noviB; };
public void cNajBo(int noviC) { c= noviC; };

} // trikotnik

```

- ob tvorjenju bomo trikotniku določili dolžine stranic, **ki jih bomo lahko kasneje spreminjali**
- poznan bo njegov obseg
- poznano bo, koliko trikotnikov je bilo narejenih (obstaja)

- tvorjenje / uničevanje
- poizvedovanje
- spreminjanje
- (zasebne) lastnosti in metode

Primer – študent

- Primer, ki si ga bomo pogledali je *študent*
- Katere lastnosti ima študent?
 - ime in priimek, ki sta mu dodeljena ob tvorjenju
 - letnik
 - vpisna številka, ki mu je dodeljena ob vpisu
 - da zna izpisati svoje podatke

Razred student

```
public class Student {
  /* -----[ data ]--- */
  private String mIme;
  private String mPriimek;
  protected int mLetnik= 0;
  private int mVpisnaStevilka= 0;
  /* -----[ create / destruct ]--- */
  public Student(String ime, String priimek);
  /* -----[ update ]--- */
  public void vpis(int vpisnaStevilka);
  /* -----[ query ]--- */
  public String ime();
  public String priimek();
  public int letnik();
  public int vpisnaStevilka();
  public void izpisi();
} // student
```

Razred student

```
/* -----[ create / destruct ]--- */
public Student(String ime, String priimek) {
    mIme= ime; mPriimek= priimek;
    mLetnik= 0; mVpisnaStevilka= 0;
};
/* -----[ update ]--- */
public void vpis(int vpisnaStevilka) {
    mVpisnaStevilka= vpisnaStevilka;
    mLetnik= 1;
};
```


Razred student

```
/* -----[ query ]---- */  
public String ime() {  
    return mIme;  
};  
public String priimek() {  
    return mPriimek;  
};  
public int    letnik() {  
    return mLetnik;  
};  
public int    vpisnaStevilka() {  
    return mVpisnaStevilka;  
};
```

Razred student

```
public void izpisi() {  
    System.out.print(mIme + " " + mPriimek);  
    System.out.print(" (" + mVpisnaStevilka + ")");  
    System.out.print(" - " + mLetnik + ":");  
};
```

Razred student – uporaba

```
public class Uporaba {
    public static lepoIzpisi(student kdo) {
        kdo.izpisi(); System.out.println();
    }; // lepoIzpisi

    public static void main(String args[]) {
        student martin= new student("Martin", "Krpan");
        student peter=  new student("Peter",  "Klepec");
        student pehta=  new student("Botra",   "Pehta");

        lepoIzpisi(martin); lepoIzpisi(peter);
        lepoIzpisi(pehta);
        martin.vpis(123); peter.vpis(124); pehta.vpis(125);

        lepoIzpisi(martin); lepoIzpisi(peter);
        lepoIzpisi(pehta);
    }; // main
}; // uporaba
```

Študent ima predmet

- Po vpisu ima študent v prvem letniku različne izpite (predmete)
- Predmeti so odvisni od smeri: nekateri študenti imajo *Sisteme I* in drugi *Diskretno matematiko*
- Ko študent opravi izpit, se takoj vpiše v drugi letnik

Študent ima predmet

- Naš razred `student` nima podpore ne za en predmet, ne za drugega.
- Želeli bi imeti funkcijo, ki bi jo klicali, ko bo študent naredil predmet in bi preko parametra vpisali oceno.
- Kaj sedaj?
 - definiramo povsem nov razred, ki bo vključeval oba predmeta in potem z neko notranjo logiko uporabljal le enega od njiju

Študent z S1 *ali* DM

```

public class StudentMAaliRIN {
    /* -----[ data ]--- */
        ...

    private    boolean mZS1;
    /* -----[ create / destruct ]--- */
    public Student(String ime, String priimek, boolean zS1);
    /* -----[ update ]--- */
        ...

    public void s1(int ocena);
    public void dm(int ocena);
    /* -----[ query ]--- */
        ...

    public int ocenaS1();
    public int ocenaDM();
} // studentMAaliRIN

    martin= new studentMAaliRIN("Martin", "Krpan", false);
    peter=  new studentMAaliRIN("Peter",  "Klepec", true);

```

Študent ima predmet

- Definiramo dva ločena razreda
 - prvega študentov matematike (MA) in drugega študentov računalništva (RIN)
 - Razreda se razlikujeta po tem, da ima prvi funkcijo d_m in drugi s_1

Razred studentMA

```
public class StudentMA {
    /* -----[ data ]--- */
    private String mIme;          private String mPriimek;
    protected int  mLetnik= 0;    private int    mVpisnaStevilka= 0;
    private int    mOcenaDM= 0;
    /* -----[ create / destruct ]--- */
    public StudentMA(String ime, String priimek);
    /* -----[ update ]--- */
    public void vpis(int vpisnaStevilka);
    public void dm(int ocena);
    /* -----[ query ]--- */
    public String ime();
    public String priimek();
    public int    letnik();
    public int    vpisnaStevilka();
    public int    ocenaDM();
    public void   izpisi();
} // studentMA
```


Razred studentMA

```
/* -----[ create / destruct ]--- */
public StudentMA(String ime, String priimek) {
    mIme= ime; mPriimek= priimek;
    mLetnik= 0; mVpisnaStevilka= 0;
    mOcenaDM= 0;
}; // studentMR
/* -----[ update ]--- */
public void vpis(int vpisnaStevilka) {
    mVpisnaStevilka= vpisnaStevilka;
    letnik= 1;
}; // vpis
public void dm(int ocena) {
    mOcenaDM= ocena;
    letnik= 2;
}; // dm
```

Razred studentMA

```
/* -----[ query ]--- */  
public String ime()      { return mIme;      };  
public String priimek() { return mPriimek; };  
public int    letnik()   { return mLetnik;   };  
public int    vpisnaStevilka() {  
    return mVpisnaStevilka;  
}; // vpisnaStevilka  
public int    ocenaDM() { return mOcenaDM; };
```

Razred studentMA

```
public void izpisi() {  
    System.out.print(mIme + " " + mPriimek);  
    System.out.print("(" + mVpisnaStevilka + ")");  
    System.out.print(" - " + mLetnik + ":");  
    System.out.print("dm: ");  
    if (mOcenaDM == 0) System.out.print("-");  
    else                System.out.print(mOcenaDM);  
}; // izpis
```

Razred studentMA - uporaba

```
public class UporabaMA {  
    public static lepolzpsi(student kdo) {  
        kdo.izpisi(); System.out.println();  
    }; // lepolzpsi  
  
    public static void main(String args[]) {  
        studentMA martin= new studentMA("Martin", "Krpan");  
        student peter= new student ("Peter", "Klepec");  
        student pehta= new student ("Botra", "Pehta");  
  
        lepolzpsi(martin); lepolzpsi(peter); lepolzpsi(pehta);  
        martin.vpis(123); peter.vpis(124); pehta.vpis(125);  
        martin.dm(9);  
  
        lepolzpsi(martin); lepolzpsi(peter); lepolzpsi(pehta);  
    }; // main  
} // uporabaMA
```

Razred StudentRIN

```
public class StudentRIN {
    /* -----[ data ]--- */
    private String mIme;          private String mPriimek;
    protected int  mLetnik= 0;    private int    mVpisnaStevilka= 0;
    private int    mOcenaS1= 0;
    /* -----[ create / destruct ]--- */
    public StudentRIN(String ime, String priimek);
    /* -----[ update ]--- */
    public void vpis(int vpisnaStevilka);
    public void s1(int ocena);
    /* -----[ query ]--- */
    public String ime();
    public String priimek();
    public int    letnik();
    public int    vpisnaStevilka();
    public int    ocenaS1();
    public void   izpisi();
} // StudentRIN
```

Razred studentRIN

```
/* -----[ create / destruct ]--- */
public StudentRIN(String ime, String priimek) {
    mIme= ime; mPriimek= priimek;
    mLetnik= 0; mVpisnaStevilka= 0;
    mOcenaS1= 0;
}; // studentRIN
/* -----[ update ]--- */
public void vpis(int vpisnaStevilka) {
    mVpisnaStevilka= vpisnaStevilka;
    mLetnik= 1;
}; // vpis
public void s1(int ocena) {
    mOcenaS1= ocena;
    letnik= 2;
}; // s1
```

Razred studentRIN

```
/* -----[ query ]--- */
public String ime()      { return mIme;      };
public String priimek() { return mPriimek; };
public int    letnik()   { return mLetnik;   };
public int    vpisnaStevilka() {
    return mVpisnaStevilka;
}; // vpisnaStevilka
public int    ocenaS1() { return mOcenaS1; };
```

Razred studentRIN

```
public void izpisi() {  
    System.out.print(mIme + " " + mPriimek);  
    System.out.print("(" + mVpisnaStevilka + ")");  
    System.out.print(" - " + letnik + ":");  
    System.out.print("s1: ");  
    if (mOcenaS1 == 0) System.out.print("-");  
    else                System.out.print(mOcenaS1);  
}; // izpis
```


Razredi študentov – uporaba

```
public class UporabaMAinRIN {
    public static lepoIzpis(student kdo) {
        kdo.izpisi(); System.out.println();
    }; // lepoIzpis

    public static void main(String args[]) {
        studentMA martin= new studentMA("Martin", "Krpan");
        studentRIN peter= new studentRIN("Peter", "Klepec");
        student pehta= new student ("Botra", "Pehta");

        lepoIzpis(martin); lepoIzpis(peter);
        lepoIzpis(pehta);
        martin.vpisi(123); peter.vpisi(124); pehta.vpisi(125);
        martin.dm(9); peter.sl(8);

        lepoIzpis(martin); lepoIzpis(peter); lepoIzpis(pehta);
    }; // main
} // uporabaMAinRIN
```

Kaj vsebuje predmet

`dostopnost tip ime(parametri);`

- Predmet vsebuje *podatke* in *metode* za rokovanje s temi podatki (podatki o lastnostih)
- Pri definiciji obeh moramo označiti, kdo jih lahko uporablja (***dostopnost***)
- Uporabljajo jih lahko:
 - Vsi: ***public***
 - Nihče: ***private***
 - Prijatelji (v modulu - *package*): ***friend***
 - Prijatelji in družina: ***protected***
- Podobno o uporabi velja tudi za razrede

Dostopnost

■ *public in private:*

```
public class A {  
    public void javno() { zasebno(); };  
    private void zasebno() { ... };  
} // A  
  
public class B {  
    public void javno(A x) {  
        x.javno(); x.zasebno();  
    }  
} // B
```

Kaj vsebuje predmet

- Poleg dostopnosti lahko pri podatkih in metodah tudi označimo ali so **skupne vsem** predmetom v razredu ali samo **posameznemu** predmetu (*this*)
 - Če so skupne vsem živim predmetom iz razreda, uporabimo oznako: *static*
- Označimo lahko ali je možno **spreminjati** vrednost ali je ta **dokončna**: *final*

Skupnost - primer za `static`

- Metode ali podatki so lahko skupni ali ne:

```
public class C {
    public static int s= 0; // skupno
    public int p= 1; // vsak posebej
} // C
C c1= new C(); C c2= new C();
```

(s, p): c1 (0, 1); c2 (0, 1);

```
c1.s= 2;
```

c1 (2, 1); c2 (2, 1);

```
c2.p= 3;
```

c1 (2, 1); c2 (2, 3);

Dokončnost - primer za `final`

- Metode ali podatki so lahko skupni ali ne:

```
public class D {  
    public static int s= 0; // spremenljivo  
    public final int p= 1; // dokončno  
} // D
```

```
D c1= new D(); D c2= new D();
```

(s, p): c1 (0, 1); c2 (0, 1);

```
c1.s= 2;
```

c1 (2, 1); c2 (2, 1);

c2.p= 3; ni dovoljeno

Funkcije, vmesniki in pogodbe...

- Na funkcijo lahko gledamo kot na črno škatlo, ki za nas nekaj naredi
 - *uporabniški pogled*
 - *uporabniški vmesnik* – **API**
 - Primer: $X(u, v) \rightarrow z$

*Ne zanima nas **kako** se nekaj naredi, ampak samo **kaj** se naredi!*

...vmesniki in pogodbe...

- Takšen pogled nam omogoča neodvisno delo v skupini
- Da je opis funkcije uporaben, potrebujemo dva podatka o funkciji:
 - opis, kako funkcijo klicati (***podpis***) in
 - opis, kaj v resnici funkcija počne (***pogodba***)

Podpis

- V našem primeru je opis kako funkcijo klicati naslednji:

```
int sestejDo0(int x)
```

- tip rezultata funkcije
- ime funkcije
- **parametri**: tip parametra in njegovo ime

Pogodbe

- Pogodba veže tistega, ki funkcijo *uporablja* in tistega, ki funkcijo *napiše* (implementira)
 - prvi vê, kaj lahko od funkcije pričakuje in zahteva
 - drugi vê, kaj mora narediti
- Če je pogodba zelo formalna, jo lahko uporabimo za formalno dokazovanje obnašanja programja
- Mi bomo uporabljali (pol)formalni zapis

Členi pogodbe

■ Opis (*description*)

Desc:

Funkcija izracuna vsoto stevil
od 0 do x.

■ Parametri (*parameters*)

Params:

x - operand [in]

Členi pogodbe

- Predpogoj (*precondition*)

Pre: $x > 0$

- Popogoj (*postcondition*)

Post: `RESULT > 0`

Členi pogodbe

- Rezultat (*result*)

Result: $x + (x-1) + (x-2) + \dots + 1$


- Okolje (*environment*)

Env: -

Podpis in pogodba

```
/*  
Desc:    Funkcija izracuna vsoto stevil  
         od 0 do x.  
Params:  x - operand [in]  
Pre:     x > 0  
Post:    RESULT > 0  
Result:  x + (x-1) + (x-2) + ... + 1  
Env:     -  
*/  
  
public int sestejDo0(x) { ... };
```

Podajanje argumentov funkcijam

- Kako se prenašajo parametri funkcij / metod?
 - Po vrednosti
 - Po referenci
 - Kaj pa je to?
- 

Parametri metod

- Prenáša se VREDNOST parametra
- Formalni / dejanski parameter
- `metodaA(12 + 5, a)`
- Enako, kot če bi pisalo `metodaA(12 + 5, 0 + a)`
- Kar se dogaja s parametri v metodi, se na spremenljivkah, uporabljenih ob klicu, NE pozna!
- ```
public static int povecajZa2 (int x){
 return x + 2;
}
```

...

```
int x = 5;
y = povecajZa2(x);
povecajZa2(z);
povecajZa2(x + y);
```



# Tabele – parametri metod

```
■ public static void sprTab(int[] x) {
 ...
}
```

■ Klic: `sprTab(nekaTabela)` ;

■ Prenese se vrednost parametra

□ naslov tabele

■ Zato – če spreminjamo elemente – spreminjajo se dejanski podatki!

# Tabele – parametri metod

- Kar se dogaja s posameznimi elementi tabele `x` v metodi `sprTab` se bo poznalo na elementih tabele `nekaTabela`
  - Ob klicu namreč `x` začne kazati na isto tabelo kot `nekaTabela` (prenese se tisto, kar je v `nekaTabela` – to pa je naslov)
  - Razen, če v metodi `sprTab` spremenljivka `x` ne pokaže na neko drugo tabelo!
    - `x = ...`
  
- "Običajne" spremenljivke
  - ```
public static void sprSprem(int x) {
    ...
}
```
 - Klic: `sprSprem(y) ;`
 - Kar se dogaja s spremenljivko `x` v metodi `sprSprem` se **NE** bo poznalo na `y` (prenese se vrednost v `y` – vrednost shranjena v `y`)

Tabele

- ```
int tab1 = new int[100];
int tab2 = new int[20];
double tab3 = new double[17];
...
tab1 = tab2; // tab1 in tab2 sta ISTI tabeli
 // do "stare" tab1 NE MOREMO VEČ
tab3 = tab1; // NAPAKA, ker niso tabele iste VRSTE
```
- S tabelo kot celoto načeloma ne počnemo nič!
- Kopija tabele
  - Prepis vseh elementov
- Izpis tabele
  - Izpis vseh posameznih elementov

# Prenos parametrov - zgled 1

```
public class Parametri1 {
 public static void main(String[] args) {
 int x = 10;
 int y = 20;
 metodaA(x, y + 2);
 }

 public static void metodaA(int a, int b) {
 int c;
 c = a;
 a = a * b;
 b = c;
 }
}
```

x <-- 10

y <-- 20

# Prenos parametrov - zgled 2

```
public class Parametri2 {
 public static void main(String[] args) {
 int x = 10;
 int y = metodaB(20);
 metodaA(x, y + 2);
 }

 public static void metodaA(int a, int b) {
 int c;
 c = a;
 a = a * b;
 b = metodaB(a);
 }

 public static int metodaB(int a) {
 int c;
 c = a;
 a = a + c;
 return a;
 }
}
```

# Povzetek

# Viri

- <http://java.sun.com/docs/books/tutorial/java/javaOO/>
- [http://publib.boulder.ibm.com/infocenter/macxhelp/v6v81/index.jsp?topic=/com.ibm.vacpp6m.doc/language/ref/clrc07examples\\_calling\\_functions.htm](http://publib.boulder.ibm.com/infocenter/macxhelp/v6v81/index.jsp?topic=/com.ibm.vacpp6m.doc/language/ref/clrc07examples_calling_functions.htm)
- <http://www.cs.toronto.edu/~dianeh/tutorials/params/>

# Naloge