Napredno programiranje i programski jezici

09 Java

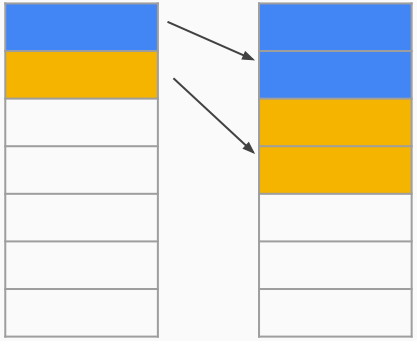
Fakultet tehničkih nauka, Novi Sad
23-24/Z
Dunja Vrbaški

```
ArrayList<Osoba>         osobeL = new ArrayList<Osoba>();

HashMap<String, Osoba>   osobeM = new HashMap<String, Osoba>();
```

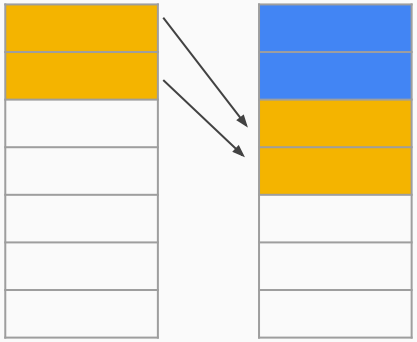
```
private static int pronadji(ArrayList<Osoba> osobe, String MB) {
      for (int i = 0; i < osobe.size(); i++) {
            if (osobe.get(i).getMB() == MB)
                  return i;
      }
      return -1;
}
```

Uspoređujemo dva stringa

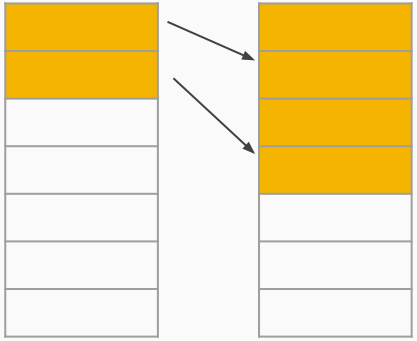
```
Pravougaonik p1 = new Pravougaonik(3, 5);
Pravougaonik p2 = new Pravougaonik(2, 4);

if (p1 == p2)
    System.out.println("isti");
else
    System.out.println("razliciti");
```



```
Pravougaonik p1 = new Pravougaonik(3, 5);
Pravougaonik p2 = new Pravougaonik(2, 4);

p1 = p2;

if (p1 == p2)
    System.out.println("isti");
else
    System.out.println("razliciti");
```



```
Pravougaonik p1 = new Pravougaonik(3, 5);
Pravougaonik p2 = new Pravougaonik(3, 5);

if (p1 == p2)
    System.out.println("isti");
else
    System.out.println("razliciti");
```



Stringovi su nizovi karaktera
Stringovi su objekti (class String)

Literali: “abc”, “”, “Petar”, “Dobar dan”,...
→ instance klase String

```
Pravougaonik p1 = new Pravougaonik(3, 5);
Pravougaonik p2 = new Pravougaonik(2, 4);

if (p1 == p2)
      System.out.println("isti");
else
      System.out.println("razliciti");
```

različiti

```
String str1 = new String("Dobar dan");
String str2 = new String("Laku noc");

if (str1 == str2)
      System.out.println("isti");
else
      System.out.println("razliciti");
```

različiti

```
Pravougaonik p1 = new Pravougaonik(3, 5);
Pravougaonik p2 = new Pravougaonik(2, 4);

p1 = p1;

if (p1 == p2)
      System.out.println("isti");
else
      System.out.println("razliciti");
```

isti

```
String str1 = new String("Dobar dan");
String str2 = new String("Laku noc");

str1 = str2;

if (str1 == str2)
      System.out.println("isti");
else
      System.out.println("razliciti");
```

isti

```
Pravougaonik p1 = new Pravougaonik(3, 5);
Pravougaonik p2 = new Pravougaonik(3, 5);

if (p1 == p2)
      System.out.println("isti");
else
      System.out.println("razliciti");
```

različiti

```
String str1 = new String("Dobar dan");
String str2 = new String("Dobar dan");

if (str1 == str2)
      System.out.println("isti");
else
      System.out.println("razliciti");
```

različiti

```
String str1 = new String("Dobar dan");
String str2 = new String("Laku noc");

if (str1 == str2)
      System.out.println("isti");
else
      System.out.println("razliciti");
```

različiti

```
String str1 = "Dobar dan";
String str2 = "Laku noc";

if (str1 == str2)
      System.out.println("isti");
else
      System.out.println("razliciti");
```

različiti

```
String str1 = new String("Dobar dan");
String str2 = new String("Laku noc");

str1 = str2;

if (str1 == str2)
      System.out.println("isti");
else
      System.out.println("razliciti");
```

isti

```
String str1 = "Dobar dan";
String str2 = "Laku noc";

str1 = str2;

if (str1 == str2)
      System.out.println("isti");
else
      System.out.println("razliciti");
```

isti

```
String str1 = new String("Dobar dan");
String str2 = new String("Dobar dan");

if (str1 == str2)
      System.out.println("isti");
else
      System.out.println("razliciti");
```

različiti

```
String str1 = "Dobar dan";
String str2 = "Dobar dan";

if (str1 == str2)
      System.out.println("isti");
else
      System.out.println("razliciti");
```

isti

string interning

```
String str1 = new String("Dobar dan");
String str2 = new String("Dobar dan");

if (str1 == str2)
      System.out.println("isti");
else
      System.out.println("razliciti");
```

različiti

```
String str1 = "Dobar dan";
String str2 = "Dobar dan";

if (str1 == str2)
      System.out.println("isti");
else
      System.out.println("razliciti");
```

isti

Literali: “abc”, “”, “Petar”, “Dobar dan”,...
→ instance klase String

Constant pool
optimizacija memorije

```
public class Pravougaonik {

   private int a;
   private int b;

   @Override
   public boolean equals(Object obj) {
      return super.equals(obj);
   }
}
```

```
public class Pravougaonik {

   private int a;
   private int b;

   @Override
   public boolean equals(Object obj) {
      return super.equals(obj);
   }
}
```

```
Pravougaonik p1 = new Pravougaonik(3, 5);
Pravougaonik p2 = new Pravougaonik(3, 5);

if (p1 == p2)
    System.out.println("==: isti");
else
    System.out.println("==: razliciti");

if (p1.equals(p2))
    System.out.println("equals: isti");
else
    System.out.println("equals: razliciti");
```

```
==: razliciti
equals: razliciti
```

```java
public class Pravougaonik {

   private int a;
   private int b;

   @Override
   public boolean equals(Object obj) {
      Pravougaonik temp = (Pravougaonik)obj;
      return (temp.a == a && temp.b == b);
   }
}
```

```java
Pravougaonik p1 = new Pravougaonik(3, 5);
Pravougaonik p2 = new Pravougaonik(3, 5);

if (p1 == p2)
    System.out.println("==: isti");
else
    System.out.println("==: razliciti");

if (p1.equals(p2))
    System.out.println("equals: isti");
else
    System.out.println("equals: razliciti");
```

```
==: razliciti
equals: isti
```

==
Uporeduje reference

equals
- Ako nije posebno implementirano - uporeduje reference
- Može se implemenitrati po želji → uporeivanje sadržaja instance

```
String str1 = new String("Dobar dan");
String str2 = new String("Dobar dan");

if (str1 == str2)
    System.out.println("==: isti");
else
    System.out.println("==: razliciti");

if (str1.equals(str2))
    System.out.println("equals: isti");
else
    System.out.println("equals: razliciti");
```

```
==: razliciti
equals: isti
```

```
String str1 = "Dobar dan";
String str2 = "Dobar dan";

if (str1 == str2)
    System.out.println("==: isti");
else
    System.out.println("==: razliciti");

if (str1.equals(str2))
    System.out.println("equals: isti");
else
    System.out.println("equals: razliciti");
```

```
==: isti
equals: isti
```

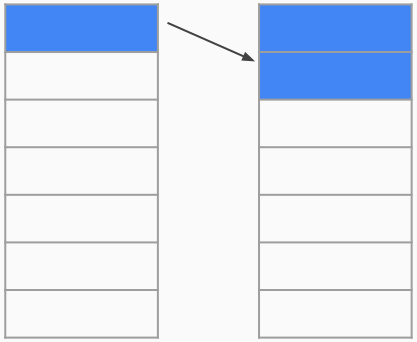
```
String str1 = "Dobar dan";
String str2 = new String("Dobar dan");

if (str1 == str2)
    System.out.println("==: isti");
else
    System.out.println("==: razliciti");

if (str1.equals(str2))
    System.out.println("equals: isti");
else
    System.out.println("equals: razliciti");
```

?

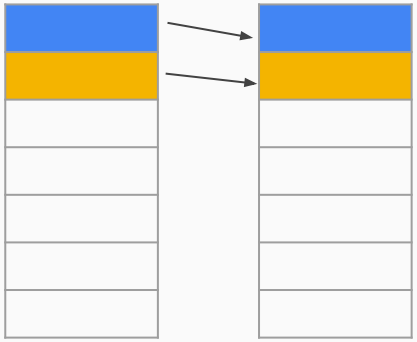
Objekti klase String su nepromenljivi
immutable

Objekti klase Pravougaonik su promenljivi
mutable

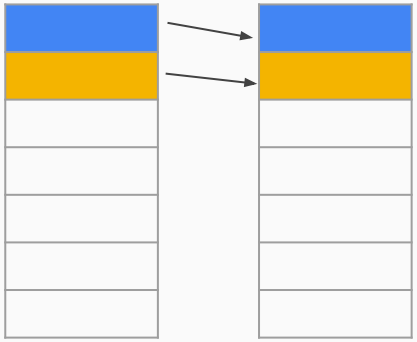
```
Pravougaonik p1 = new Pravougaonik(3, 5);
p1.setA(7);
```



```
String str1 = new String("Dobar dan");
String str2 = str1.replace('a', 'x');
```



```
String str1 = new String("Dobar dan");
String str2 = str1.replace('a', 'x');
```



```
String str1 = new String("Dobar dan");
str1.replace('a', 'x');
```

Paziti - neće biti prijavljena greška!

```
public class ModifiersTest {

	public static void main(String[] args) {
		int x = 3, y = 5;
		System.out.println(Saberi(x, y));
	}

	private static int Saberi(int x, int y) {
		return x + y;
	}
}
```

static članovi
pripadaju čitavoj klasi
nisu vezani za instance

```
public class Osoba {

   ...

   public static String KlasaInfo() {
       String str = "Klasa osoba ima ";
        str += "private polja: ime, prezime i MB, ";
       str += "public get/set metode, ";
       str += " toString (override) metod.";
       return str;
   }

   ...
}
```

```
public static void main(String[] args) {
   int x = 3, y = 5;
   System.out.println(Saberi(x, y));

   System.out.println(Osoba.KlasaInfo());
}

private static int Saberi(int x, int y) {
   return x + y;
}
```

```
public class Osoba {

   private String ime;
   ...

   public static String KlasaInfo() {
      ...
   }

   public String getIme() {
      ...
   }

   ...
}
```

metoda klase

metoda instance

```
public class Osoba {

   private String ime;
   ...

   public static String KlasaInfo() {
       String str = "Klasa osoba ima ";
        str += "private polja: ime, prezime i MB, ";
       str += "public get/set metode, ";
       str += " toString (override) metodu.";

       str += ime;

       return str;
   }

   ...
}
```

U static metodi ne možemo pristupati poljima instance

Ako se metod odnosi na celu klasu čije tačno ime želimo da uzmemo?

```
public static void main(String[] args) {
   int x = 3, y = 5;

   System.out.println(Saberi(x, y));

   System.out.println(Saberi2(x, y));

   System.out.println(Osoba.KlasaInfo());
}

private static int Saberi(int x, int y) {
   return x + y;
}

private int Saberi2(int x, int y) {
   return x + y;
}
```

U static metodama ne možemo pristupati metodama instance (iz static metode ne možemo pozivati metode koje nisu static)

```
public class Osoba {

   private String ime;
   ...

   public static String KlasaInfo() {
       String str = "Klasa osoba ima ";
        str += "private polja: ime, prezime i MB, ";
       str += "public get/set metode, ";
       str += " toString (override) metodu.";

       str += ime;

       return str;
   }

   ...
}
```

```
public static void main(String[] args) {
   int x = 3, y = 5;

   System.out.println(Saberi(x, y));

   System.out.println(Saberi2(x, y));

   System.out.println(Osoba.KlasaInfo());

   Osoba o1 = new Osoba();
   System.out.println(o1.KlasaInfo());
}

private static int Saberi(int x, int y) {
   return x + y;
}

private int Saberi2(int x, int y) {
   return x + y;
}
```

Možemo da pristupimo i preko bilo koje instance
(sve instance dele static članove)
Izbegavati; upozorenje

```
public class Osoba {

   private String ime;
   …
   private static int brojInstanci;

   public static String KlasaInfo() {
       String str = "Klasa osoba ima ";
        str += "private polja: ime, prezime i MB, ";
       str += "public get/set metode, ";
       str += " toString (override) metod. ";

       str += ime;
       str += "Broj instanci: ";
       str += brojInstanci;

       return str;
   }

   ...
}
```

```
public class Osoba {

   private String ime;
   …
   private static int brojInstanci;

   public static String KlasaInfo() {
       String str = "Klasa osoba ima ";
        str += "private polja: ime, prezime i MB, ";
       str += "public get/set metode, ";
       str += " toString (override) metod. ";

       str += "Broj instanci: ";
       str += brojInstanci;

       return str;
   }

   ...
```

```
   public Osoba() {
       brojInstanci++;
   }

   public Osoba(
              String ime,
              String prezime,
              String MB) {
       this.ime = ime;
       this.prezime = prezime;
       this.MB = MB;
       brojInstanci++;
   }


   ...
}
```

Obrnuto, u metodi instance možemo pristupati static članovima

Field Summary

Fields

Modifier and Type	Field	Description
static final double	E	The double value that is closer than any other to *e*, the base of t
static final double	PI	The double value that is closer than any other to *pi*, the ratio of

Method Summary

All Methods | Static Methods | Concrete Methods

Modifier and Type	Method	Description
static double	abs(double a)	Returns the absolute value of a double value.
static float	abs(float a)	Returns the absolute value of a float value.
static int	abs(int a)	Returns the absolute value of an int value.
static long	abs(long a)	Returns the absolute value of a long value.
static int	absExact(int a)	Returns the mathematical absolute value of an in positive int range.
static long	absExact(long a)	Returns the mathematical absolute value of an lo the positive long range.
static double	acos(double a)	Returns the arc cosine of a value; the returned an
static int	addExact(int x, int y)	Returns the sum of its arguments, throwing an ex
static long	addExact(long x, long y)	Returns the sum of its arguments, throwing an ex
static double	asin(double a)	Returns the arc sine of a value; the returned angl

Math class

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

	Math m = new Math();

	System.out.println(m.abs(-5));
}
```

?

- private ctor
- static članovi

→ recept?

OOP design patterns
šabloni, preporuke, uputstva za: modeliranje, rešavanje problema, izgradnju softvera

```
public final class Math
```

```
static final double E
static final double PI
```

Modifikator za klase, metode, promenljive.

```
public final class Osoba {

      private String ime;
      private String prezime;
      private String MB;

      ...

}
```

```
public class Student extends Osoba {
      ...

}
```

The type Student cannot subclass the final class Osoba

```
public class Osoba {

   private String ime;
   private String prezime;
   private String MB;

   ...

   public final void printInfo() {
      System.out.println("Informacije o osobi: ");
      System.out.println(this);
   }
}
```

```
public class Student extends Osoba {
      ...


      @Override
      public void printInfo() {
            ...
      }
}
```

Cannot override the final method from Osoba

```
public class Osoba {

   private String ime;
   private String prezime;
   private String final MB;

   ...
```

```
public Osoba() {
      ime = "";
      prezime = "";
      MB = "";
}

public Osoba(
            String ime,
            String prezime,
            String MB) {
      this.ime = ime;
      this.prezime = prezime;
      this.MB = MB;
}

public void setMB(String mB) {
      MB = mB;
}

The final field Osoba.MB cannot be assigned
```

```
public class Osoba {

   private String ime;
   private String prezime;
   private String final MB;

   ...

   private final int magicNumber = 42;

   private static final int magicNumber2 = 505;
```

Obratiti pažnju:

final polje
Zabrana nove dodele
Sama vrednost je potencijalno promenljiva

```
public class A {
	private int x;

	public int getX() {
		return x;
	}

	public void setX(int x) {
		this.x = x;
	}

	A(int x) {
		this.x = x;
	}

	@Override
	public String toString() {
		return "A: x = " + x;
	}
}
```

```
public class B {
	private int y;
	private A a;

	public B(int y, A a) {
		this.y = y;
		this.a = a;
	}

	@Override
	public String toString() {
		return "B: y = " + y + " a: " + a;
	}

}
```

```
public class A {
      private int x;
      ...
}

public class B {
      private int y;
      private A a;
       ...

}
```

```
A a1 = new A(3);
B b1 = new B(5, a1);

System.out.println(a1);
System.out.println(b1);

a1.setA(100);
System.out.println(a1);
System.out.println(b1);
```

```
A: x = 3
B: y = 5 a: A: x = 3
A: x = 100
B: y = 5 a: A: x = 100
```

```
public class A {
      private int x;
      ...
}

public class B {
      private int y;
      private final A a;
       ...

}
```

```
A a1 = new A(3);
B b1 = new B(5, a1);

System.out.println(a1);
System.out.println(b1);

a1.setA(100);
System.out.println(a1);
System.out.println(b1);
```

?

```
public class A {
      private int x;
      ...
}

public class B {
      private int y;
      private final A a;
       ...

}
```

```
A a1 = new A(3);
B b1 = new B(5, a1);

System.out.println(a1);
System.out.println(b1);

a1.setA(100);
System.out.println(a1);
System.out.println(b1);
```

```
A: x = 3
B: y = 5 a: A: x = 3
A: x = 100
B: y = 5 a: A: x = 100
```

```
public class A {
      private int x;
      ...
}

public class B {
      private int y;
      private final A a;
       …

      public void promeniA() {
            a.setX(42);
            a = new A(42);
      }



}
```

```
A a1 = new A(3);
B b1 = new B(5, a1);

System.out.println(a1);
System.out.println(b1);

a1.setA(100);
System.out.println(a1);
System.out.println(b1);
```

```
public class A {
      private int x;
      ...
}

public class B {
      private int y;
      private final A a;
       ...

}
```

```
A a1 = new A(3);
B b1 = new B(5, a1);

System.out.println(a1);
System.out.println(b1);

a1.setA(100);
System.out.println(a1);
System.out.println(b1);

a1 = new A(-10);
System.out.println(a1);
System.out.println(b1);
```

?

```
public class A {
      private int x;
      ...
}

public class B {
      private int y;
      private final A a;
       ...

}
```

```
A a1 = new A(3);
B b1 = new B(5, a1);

System.out.println(a1);
System.out.println(b1);

a1.setA(100);
System.out.println(a1);
System.out.println(b1);

a1 = new A(-10);
System.out.println(a1);
System.out.println(b1);
```

```
A: x = 3
B: y = 5 a: A: x = 3
A: x = 100
B: y = 5 a: A: x = 100
A: x = -10
B: y = 5 a: A: x = 100
```