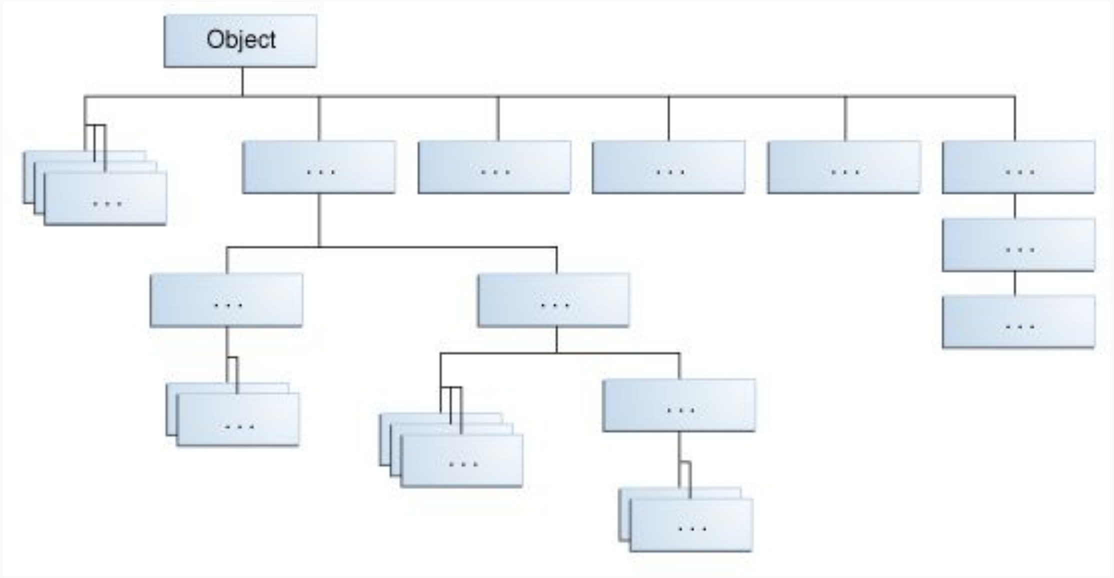
Napredno programiranje i programski jezici

07 Java

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


Object
. . .

```
public class Pravougaonik {
	private double a;
	private double b;

	public Pravougaonik(double a, double b) {
		this.a = a;
		this.b = b;
	}
	public double getA() ...
	public void setA(double a) ...
	public double getB() ...
	public void setB(double b) ...

	public double getO() ...
	public double getP() ...

	@Override
	public String toString() {
		return super.toString();
	}

}
```

```
public static void main(String[] args) {
	Pravougaonik p = new Pravougaonik(3, 5);
	System.out.println("P = " + p.getP());
	System.out.println("O = " + p.getO());
}
```

class Object → toString

tekst reprezentacija objekta
očekuje se na mnogim mestima

```
public class Pravougaonik {
	private double a;
	private double b;

	public Pravougaonik(double a, double b) {
		this.a = a;
		this.b = b;
	}
	public double getA() ...
	public void setA(double a) ...
	public double getB() ...
	public void setB(double b) ...

	public double getO() ...
	public double getP() ...

	@Override
	public String toString() {
		return super.toString();
	}

}
```

@Override

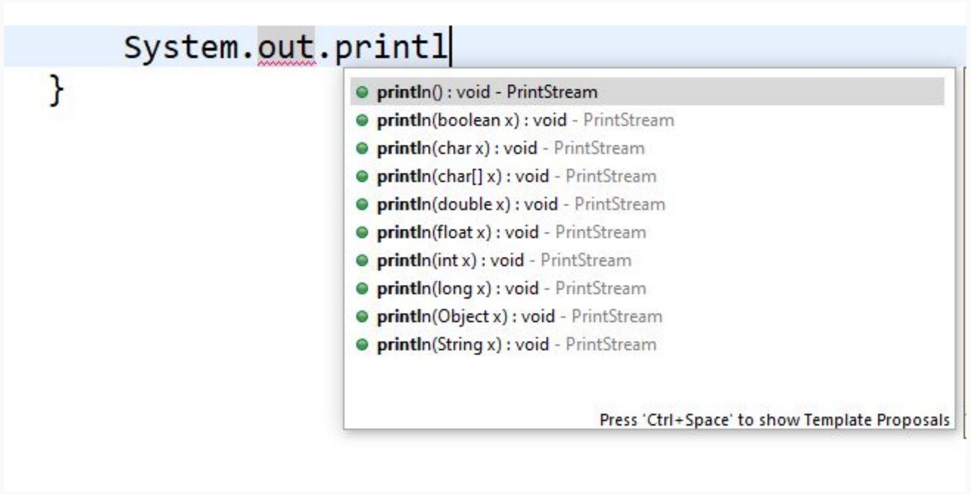
- kompajler proverava
- čitljivost
- greške su češće nego što se možda čini

```
	public String tostring() {
		return "dummy";
	}
```

```java
public class Pravougaonik {
	private double a;
	private double b;

	public Pravougaonik(double a, double b) {
		this.a = a;
		this.b = b;
	}
	public double getA() ...
	public void setA(double a) ...
	public double getB() ...
	public void setB(double b) ...

	public double getO() ...
	public double getP() ...

	@Override
	public String toString() {
		return super.toString();
	}

}
```

```java
public static void main(String[] args) {
	Pravougaonik p = new Pravougaonik(3, 5);
	System.out.println("P = " + p.getP());
	System.out.println("O = " + p.getO());

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

```
P = 15.0
O = 16.0
nppj.Pravougaonik@436e852b
```



```
        System.out.printl
    }
```

```
println() : void - PrintStream
println(boolean x) : void - PrintStream
println(char x) : void - PrintStream
println(char[] x) : void - PrintStream
println(double x) : void - PrintStream
println(float x) : void - PrintStream
println(int x) : void - PrintStream
println(long x) : void - PrintStream
println(Object x) : void - PrintStream
println(String x) : void - PrintStream
```

Press 'Ctrl+Space' to show Template Proposals

```java
public class Pravougaonik {
	private double a;
	private double b;

	public Pravougaonik(double a, double b)...
	public double getA() ...
	public void setA(double a) ...
	public double getB() ...
	public void setB(double b) ...

	public double getO() ...
	public double getP() ...

	@Override
	public String toString() {
		String str   = "";
		str += "a = " + a;
		str += " b = " + b;
		str += " P = " + getP();
		str += " O = " + getO();
		return str;
	}
}
```

```java
public static void main(String[] args) {
	Pravougaonik p = new Pravougaonik(3, 5);
	System.out.println("P = " + p.getP());
	System.out.println("O = " + p.getO());

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

```
P = 15.0
O = 16.0
a = 3.0 b = 5.0 P = 15.0 O = 16.0
```

ZADATAK: Realizovati klase Osoba i Student.

```
public class Osoba {
        private String ime;
        private String prezime;
        private String MB;

        public Osoba() {}

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

        public String getIme() {
                return ime;
        }

        public void setIme(String ime) {
                this.ime = ime;
        }
```

```
        public String getPrezime() {
                return prezime;
        }

        public void setPrezime(String prezime) {
                this.prezime = prezime;
        }

        public String getMB() {
                return MB;
        }

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

        @Override
        public String toString() {
                String str = "";
                str += ime + " ";
                str += prezime + " ";
                str += MB + " ";
                return str;
        }
}
```

```java
public class Osoba {
      private String ime;
      private String prezime;
      private String MB;

      public Osoba()...
      public Osoba(...)...

      public String getIme()...
      public void setIme(String ime)...
      public String getPrezime() ...
      public void setPrezime(String prezime)...
      public String getMB()...
      public void setMB(String mB)...

      @Override
      public String toString() {
            String str = "";
            str += ime + " ";
            str += prezime + " ";
            str += MB + " ";
            return str;
      }
}
```

```java
public class StudentApp {

      public static void main(String[] args) {
            Osoba o1 = new Osoba();
            System.out.println(o1);

            Osoba o2 = new Osoba(
                        "Petar", "Petrovic",
                        "123456");
            System.out.println(o2);
      }
}
```

```
null null null
Petar Petrovic 123456
```

```java
public class Osoba {
	private String ime;
	private String prezime;
	private String MB;

	public Osoba()...
	public Osoba(...)...

	public String getIme()...
	public void setIme(String ime)...
	public String getPrezime() ...
	public void setPrezime(String prezime)...
	public String getMB()...
	public void setMB(String mB)...

	@Override
	public String toString() {
		String str = "";
		str += ime + " ";
		str += prezime + " ";
		str += MB + " ";
		return str;
	}
}
```

```java
public class Student extends Osoba {

}
```

```
public class Osoba {
	private String ime;
	private String prezime;
	private String MB;

	public Osoba() {}

	public Osoba(
			String ime,
			String prezime,
			String MB) {
		this.ime = ime;
		this.prezime = prezime;
		this.MB = MB;
	}
	// set, get
	@Override
	public String toString() {
		String str = "";
		str += ime + " ";
		str += prezime + " ";
		str += MB + " ";
		return str;
	}
}
```

```
public class Student extends Osoba {
	private int brIndeksa;



	public Student() {  }

	public Student(
			String ime,
			String prezime,
			String MB,
			int brIndeksa) {
		super(ime, prezime, MB);
		this.brIndeksa = brIndeksa;
	}


}
```

```
public class Osoba {
        // private polja: ime, prezime, MB

        public Osoba()...
        public Osoba(...)...

        // set, get

        @Override
        public String toString() {
                String str = "";
                str += ime + " ";
                str += prezime + " ";
                str += MB + " ";
                return str;
        }
}
```

```
public class Student extends Osoba {
        // private polja: brIndeksa

        public Student() ...
        public Student(...)...

        // set, get
}
```

```
public static void main(String[] args) {
        Student s1 = new Student();
        System.out.println(s1);

        Student s2 = new Student(
                        "Petar", "Petrovic",
                        "123456",
                        42);
        System.out.println(s2);
}
```

?

```
public class Osoba {
	// private polja: ime, prezime, MB

	public Osoba()...
	public Osoba(...)...

	// set, get

	@Override
	public String toString() {
		String str = "";
		str += ime + " ";
		str += prezime + " ";
		str += MB + " ";
		return str;
	}
}
```

```
null null null
Petar Petrovic 123456
```

```
public class Student extends Osoba {
	// private polja: brIndeksa

	public Student() ...
	public Student(...)...

	// set, get
}
```

```
public static void main(String[] args) {
	Student s1 = new Student();
	System.out.println(s1);

	Student s2 = new Student(
			"Petar", "Petrovic",
			"123456",
			42);
	System.out.println(s2);
}
```

```
public class Osoba {
        // private polja: ime, prezime, MB

        public Osoba()...
        public Osoba(...)...

        // set, get

        @Override
        public String toString() {
                String str = "";
                str += ime + " ";
                str += prezime + " ";
                str += MB + " ";
                return str;
        }
}
```

```
public class Student extends Osoba {
        // private polja: brIndeksa

        public Student() ...
        public Student(...)...

        // set, get

        @Override
        public String toString() {
                String str = super.toString();
                str += " " + brIndeksa;
                return str;
        }

}
```

```
null null null 0
Petar Petrovic 123456 42
```

```
public abstract class Osoba {

        // private polja: ime, prezime, MB

        public Osoba()...
        public Osoba(...)...

        // set, get

        @Override
        public String toString()...
}
```

```java
public abstract class Osoba {

	// private polja: ime, prezime, MB

	public Osoba()...
	public Osoba(...)...

	// set, get

	@Override
	public String toString()...
}
```

```java
public static void main(String[] args) {
	Osoba o1 = new Osoba();
	System.out.println(o1);

	Osoba o2 = new Osoba(
			"Petar", "Petrovic",
			"123456");
	System.out.println(o2);
}
```

```
Cannot instantiate the type Osoba
```

C++

```
void virtual mojMetod(...)  = 0;
```

- bar jedan pure virtual metod
- metod i klasu smo zvali apstraktnim

Java

```
public abstract class MojaKlasa {...}
```

- eksplicitno se navodi KW
- može, ali ne mora imati apstraktne metode

```
public abstract class Osoba {

        // private polja: ime, prezime, MB

        public Osoba()...
        public Osoba(...)...

        // set, get

        @Override
        public String toString()...

        public abstract void log();
}
```

```java
public abstract class Osoba {

        // private polja: ime, prezime, MB

        public Osoba()...
        public Osoba(...)...

        // set, get

        @Override
        public String toString()...

        public abstract void log();
}
```

```java
public class Student extends Osoba {

        // private polja: brIndeksa

        public Student() ...
        public Student(...)...

        // set, get

        @Override
        public String toString() ...

}
```

```
The type Student must implement the inherited
abstract method Osoba.log()
```

```java
public abstract class Osoba {

      // private polja: ime, prezime, MB

      public Osoba()...
      public Osoba(...)...

      // set, get

      @Override
      public String toString()...

      public abstract void log();
}
```

```java
public class Student extends Osoba {

      // private polja: brIndeksa

      public Student() ...
      public Student(...)...

      // set, get

      @Override
      public String toString() …

      @Override
      public void log() {

      }
}
```

```
public abstract class Osoba {

      // private polja: ime, prezime, MB

      public Osoba()...
      public Osoba(...)...

      // set, get

      @Override
      public String toString()...

      public abstract void log();
}
```

```
public class Student extends Osoba {

      // private polja: brIndeksa

      public Student() ...
      public Student(...)...

      // set, get

      @Override
      public String toString() …

      @Override
      public void log() {
            System.out.println(this);
      }
}
```

```
public class Osoba {

      // private polja: ime, prezime, MB

      public Osoba()...
      public Osoba(...)...

      // set, get

      @Override
      public String toString()...

}
```

```
public class Student extends Osoba {

      // private polja: brIndeksa

      public Student() ...
      public Student(...)...

      // set, get

      @Override
      public String toString() …

}
```

Vratimo se na početak, klasa osoba nije apstraktna

```java
public interface Logger {
	void log();
}
```

```java
public interface Logger {
     void log();
}
```

```java
public class Osoba implements Logger {

      // private polja: ime, prezime, MB

      public Osoba()...
      public Osoba(...)...

      // set, get

      @Override
      public String toString()...

}
```

metoda je public abstract.

```
public interface Logger {
      void log();
}
```

```
public class Osoba implements Logger {

      // private polja: ime, prezime, MB

      public Osoba()...
      public Osoba(...)...

      // set, get

      @Override
      public String toString()...

}
```

metode je public abstract.

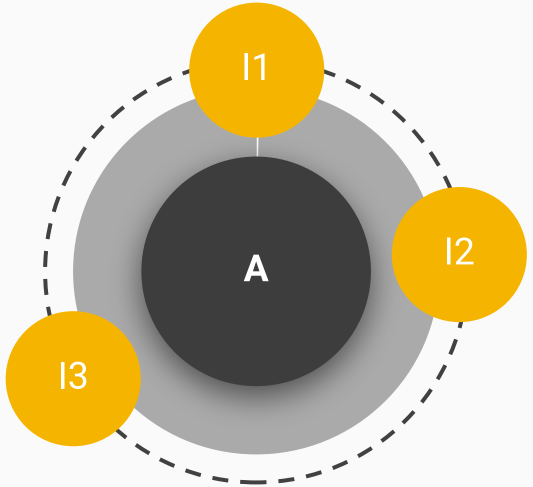
```
The type Osoba must implement the inherited abstract method Logger.log()
```

```
public interface Interfjes1 {
      void metod11(...);
      int metod12(...);
     ...
      void metod1N(...);
}
```

```
public interface Interfjes2 {
      void metod21(...);
      void metod22(...);
     ...
      double metod2M(...);
}
```

```
public interface Interfjes3 {
      void metod31(...);
      Object metod32(...);
     ...
      String metod3K(...);
}
```

```
public class A implements Interfejs1, Interfejs2, Interfejs3 {
      public void metod11() {...}
      public int metod12() {...}

      ...

      String void metod3K() {...}
}
```



```java
public class Osoba implements Logger {

        // private polja: ime, prezime, MB

        public Osoba()...
        public Osoba(...)...

        // set, get

        @Override
        public String toString()...

}
```

```java
public class Student extends Osoba {

        // private polja: brIndeksa

        public Student() ...
        public Student(...)...

        // set, get

        @Override
        public String toString() …

}
```

```
The type Osoba must implement the inherited abstract method Logger.log()
```

- Osoba implementira log(), Student redefiniše
- Osoba ostane apstraktna, Student definiše log()

```java
public abstract class Osoba implements Logger {

       // private polja: ime, prezime, MB

       public Osoba()...
       public Osoba(...)...

       // set, get

       @Override
       public String toString()...

}
```

```java
public class Student extends Osoba {

       // private polja: brIndeksa

       public Student() ...
       public Student(...)...

       // set, get

       @Override
       public String toString() …

}
```

~~The type **Osoba** must implement the inherited abstract method Logger.log()~~

The type **Student** must implement the inherited abstract method Logger.log()

```
public abstract class Osoba implements Logger {

	// private polja: ime, prezime, MB

	public Osoba()...
	public Osoba(...)...

	// set, get

	@Override
	public String toString()...

}
```

```
public class Student extends Osoba {

	// private polja: brIndeksa

	public Student() ...
	public Student(...)...

	// set, get

	@Override
	public String toString() …

	@Override
	public void log() {
		System.out.println(this);
	}
}
```

~~The type **Osoba** must implement the inherited abstract method Logger.log()~~

~~The type **Student** must implement the inherited abstract method Logger.log()~~

```
public interface Logger {
     void log();
     void logShort();
     void logExtended();
     void logCrypto();
}
```

ZADATAK

(neobavezno)

Napraviti dve iste klase u C++ i Javi (Pravougaonik, Student, …)
Posmatrati dodele u obe jezika.

Napisati promeni()

- C++: void promeni(Pravougaonik p), slobodna funkcija
- Java: static void promeni(Pravougaonik p), metod u klasi gde je main metod

Tu promeniti objekat. Posmatrati.

Sve realizovano u C++ (do lista) možete probati da realizujete u Javi.
Umesto operatora << koristiti toString.