Академија струковних
студија Шумадија
одсек у Крагујевцу

# Увод у програмирање

## Презентација 8

---

Академија струковних студија Шумадија

Одсек у Крагујевцу

Студијски програм Информатика

Крагујевац, 2020. година

# Објектно оријентисано програмирање

- Појам објектно оријентисаног програмирања датира још од 1960-их година прошлог века.
- Објектно оријентисано програмирање је програмска парадигма заснована **на скупу објеката који имају међусобну интеракцију**.
- Шта је у ствари објекат?
- На изглед једноставно питање, а одговор, по дефиницији, би био: **објекат је целина која садржи податке и понашање**.
- Објекти свакако **нису** прости типови података, попут целобројних типова, карактера, ...
- Подаци унутар објекта представљају **атрибуте** (особине) објекта, а у литератури их можете наћи и под именом **поља**.
- **Понашање** објекта је дефинисано његовим методама.

# Објектно оријентисано програмирање

- Објектно орјентисано програмирање је нови приступ реализацији софтвера као модела из реалног света.
- Објекти су за објектно оријентисани програм исто што и цигле за неку зграду или кућу.
- У објектним програмским системима све је представљено као објекат (процеси, текст, улазно/излазне операције, итд.).
- Објекат има своје **унутрашње стање** чија је реализација недоступна другим објектима, као и **операције** (методе) које се над њим споља могу извршавати.
- Можемо рећи да је објекат је све што има јединствен идентитет: студент, сто, столица, екран, круг, кредит, испит, …
- Сваки реални појам из природе се путем **апстракције** може представити једним објектом у програмском језику.

# Објектно оријентисано програмирање

## Апстракција података

- Програмски језик мора програмеру понудити могућност систематског дефинисања нових типова података, заједно са операцијама које су над њима могуће.
- Апстракција је принцип игнорисања оних особина неког објекта које нису релевантне у датој ситуацији, тј. **усредсређивање на битне ствари**.
- Као пример можемо посматрати особу.
- Свака особа има: име, презиме, датум рођења, висину, тежину, ЈМБГ, боју косе, боју очију, пребивалиште, ..., број индекса (ако је студент), ...
- Нпр. ако правимо апстракцију појма особа за неку медицинску ординацију, а у исто време имамо задатак да креирамо информациони систем студентске службе (нпр. нашег Одсека), јасно је да нећемо узети обзир исте карактеристике особе за медицинску ординацију и студентску службу.

# Објектно оријентисано програмирање

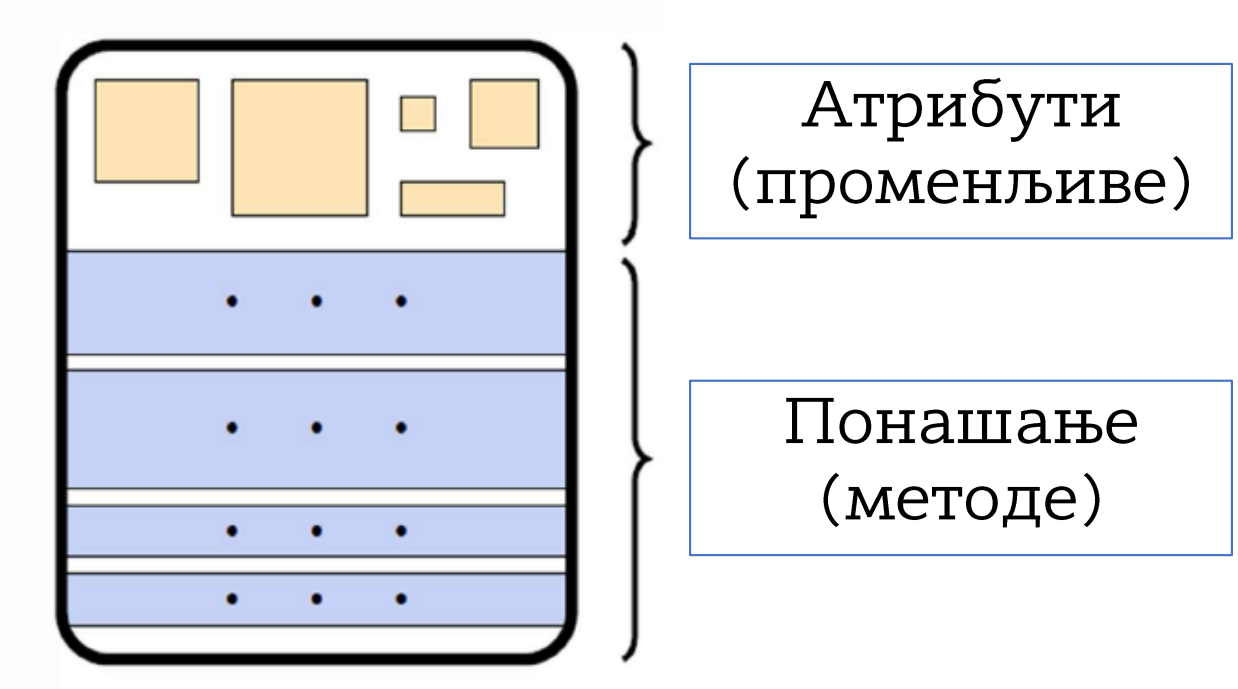
**Апстракција података**

- Објекат из реалног света – аутомобил.

# Објектно оријентисано програмирање

## Апстракција података

- Пример апстракције појма аутомобил.
- Атрибути:
  a) марка = "Ford"
  b) модел = "Mustang GT"
  c) боја = "црвена"
  d) број врата = 2
  e) брзина = 280
- Понашање:
  a) покрени мотор
  b) заустави мотор
  c) додај гас
  d) промени брзину
  e) кочи
  f) скрени



# Објектно оријентисано програмирање

**Пример**: реализација апстракције у класи Аутомобил

```
public class Automobil {

    // atributi klase Automobil
    public String marka = "Ford";
    public String model = "Mustang GT";
    public String boja = "crvena";
    public int brojVrata = 2;
    public int brzina = 280;

    // metode klase Automobil
    public void pokreniMotor(){ // naredbe }
    public void zaustaviMotor(){ // naredbe }
    public void dodajGas(){ // naredbe }

    // ostale metode
}
```

# Објектно оријентисано програмирање

- **Класа** је општи представник неког **скупа објеката** (предмета или појава) који имају исту структуру и понашање.
- Класа је упрошћена слика ових реалних предмета и појава и обухвата њихове:

a) карактеристике (атрибуте),

b) понашања (методе),

c) односе са другим класама (релације).

- Класа одређује **шаблон** (мустру) како изгледају њени појединачни објекти.
- Објекти се у програму не описују појединачно, већ дефинисањем класе тих објеката.
- Претходни пример представља дефинисање класе "*Automobil*".
- Објекти се након дефинисања класе конструишу, по потреби, на основу те класе и та конструкција представља **инстанцу** (примерак) те класе.

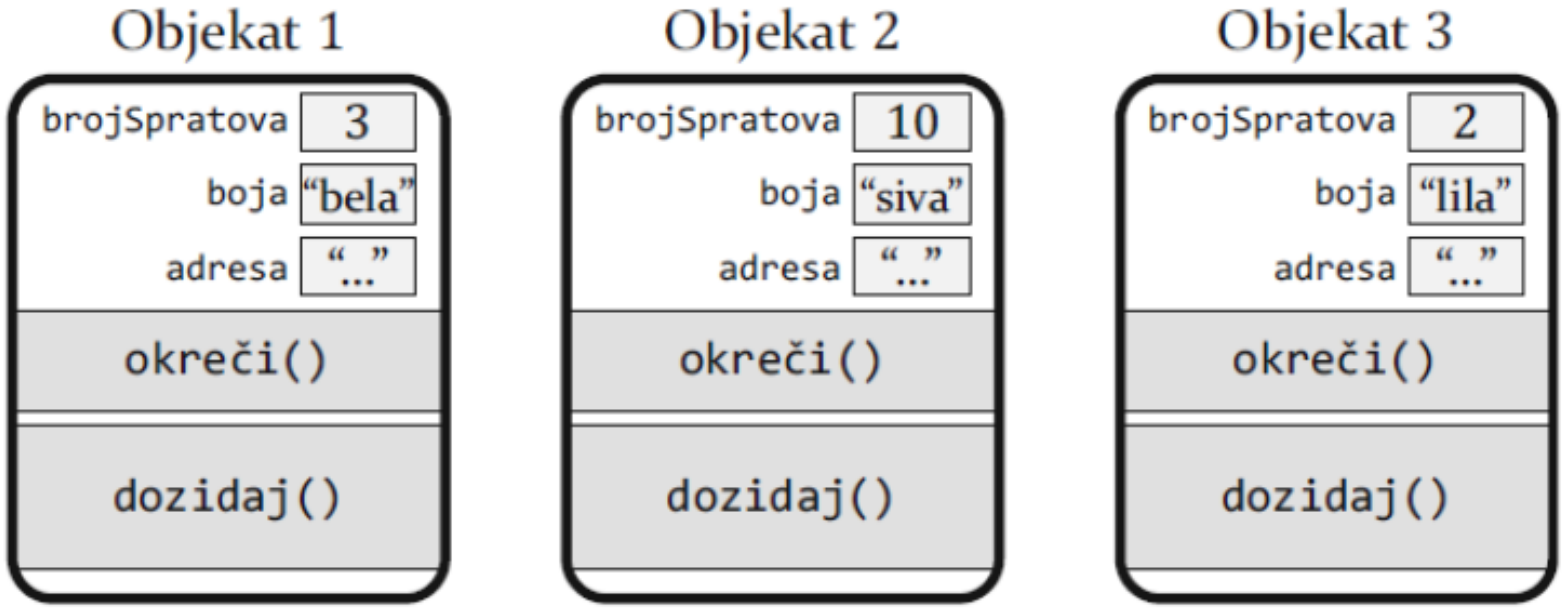
# Објектно оријентисано програмирање

**Аналогија**

- Аутомобил је представљен класом.
- Сваки конкретан примерак те класе представља објекат.
- На основу класе Аутомобил може се инстанцирати (креирати) више објеката, који при том могу имати различите атрибуте и понашања.
- Конкретно, из поменуте класе могуће је креирати аутомобиле различитих произвођача – довољно је само променити вредности атрибута и начин реализације метода.
- Тако нпр. могли смо да добијемо различите моделе аутомобила марке Форд: Фокус, Куга, Мондео, ..., али и да креирамо објекте који ће представљати аутомобиле марке Мерцедес, са својим атрибутима и понашањима.
- Тренутна реализација класе Аутомобил није погодна за промену марке, модела и осталих атрибута и понашања али даљом апстракцијом лако можемо доћи до такве класе.

# Објектно оријентисано програмирање

- Класа се у Јави може схватити и као модел за дефинисање новог типа података.
- Дефиниција класе се користи да се креирају објекти **<u>тог класног типа</u>**, тј. да се креирају објекти који садрже све компоненте које су наведене у класи.
- На основу дефиниције класе може се конструисати више објеката (већ смо помињали али посматрајмо класу "Зграда"):



# Објектно оријентисано програмирање

- Атрибути класе могу да буду примитивног типа или могу да дају референцу на објекте произвољног класног типа, укључујући и онај који дефинишемо (о томе на неким од наредних предавања).
- Дефиниција атрибута се мора налазити изван свих метода, али унутар класе.
- Разликујемо две врсте атрибута (као и код метода):
  - Нестатички атрибут (објектни) – један по објекту.
  - Статички атрибут (класни) – један по класи.
- Сваки објекат класе поседује своју сопствену копију нестатичког атрибута.
  - Овакви атрибути дају објектима индивидуалност, тј. разликују их међусобно.
- Статички атрибут је заједнички за све објекте класе.
  - Постоји само једна копија тог атрибута без обзира на то колико објеката те класе је креирано и та копија постоји чак и када није креиран ниједан објекат те класе.
  - Као и код метода, декларише се кључном речју ***`static`***.

# Објектно оријентисано програмирање

- Статички атрибути обично описују неко својство које је заједничко за све објекте те класе.
- Још једна примена статичких атрибута је чување вредности које су заједничке свим објектима те класе.
- Пример:

```
public class PorodičnoStablo {
    public static String prezime;
    public String ime;
    public int uzrast;

    public void informacija(){
        if(uzrast < 18){
            System.out.println("Maloletnik!");
        }else {
            System.out.println("Odrasla osoba!");
        }
    }
}
```

# Објектно оријентисано програмирање

- Аналогно атрибутима, постоје две врсте метода:
  - статичке или класне методе и
  - нестатичке или објектне методе – извршавају се само у односу на објекат.
- Статичке методе припадају класи и постоје за све време извршавања програма.
- Због ове њихове особине, статичке методе се могу позивати у програму независно од конструисаних објеката њихове класе.
  - Метод **`main()`** је, као што смо видели раније, увек декларисан као статички.
  - Дакле, пре него што нека апликација почне да се извршава, у методи **`main()`** не мора да постоји ни један објекат.
- Статичке методе могу обављати општи задатак за све објекте класе.
- Статичке методе могу директно приступати само статичким пољима и другим статичким методама класе.
- Посматраћемо следећи пример, нашу класу Аутомобил:

# Објектно оријентисано програмирање

```java
public class Automobil {

    // atributi klase Automobil
    public static String marka;
    public static String model;
    public String boja = "crvena";
    public int brojVrata = 2;
    public int brzina = 280;

    // statička metoda klase Automobil
    public static String info(){
        String auto = "Vozimo automobil "
                + marka + " " + model;
        return  auto;
    }

    // ostale metode klase Automobil
}
```

```java
public class GlavnaKlasa {

    public static void main(String[] args) {

        Automobil.marka = "Ford";
        Automobil.model = "Mistang GT";
        System.out.println(Automobil.info());
    }
}
```

```
Run:  GlavnaKlasa
"C:\Program Files\Java\jdk1.8.0_121\bin\java.exe" ...
Vozimo automobil Ford Mistang GT

Process finished with exit code 0
```

```java
// statička metoda klase Automobil
public static String info(){
    String auto = "Vozimo automobil "
            + marka + " " + model + boja + brzina;

    return  auto;
}
```

Пошто **boja** и **brzina** нису статичка поља (атрибути) не можемо их користити у статичком методу **info()**.

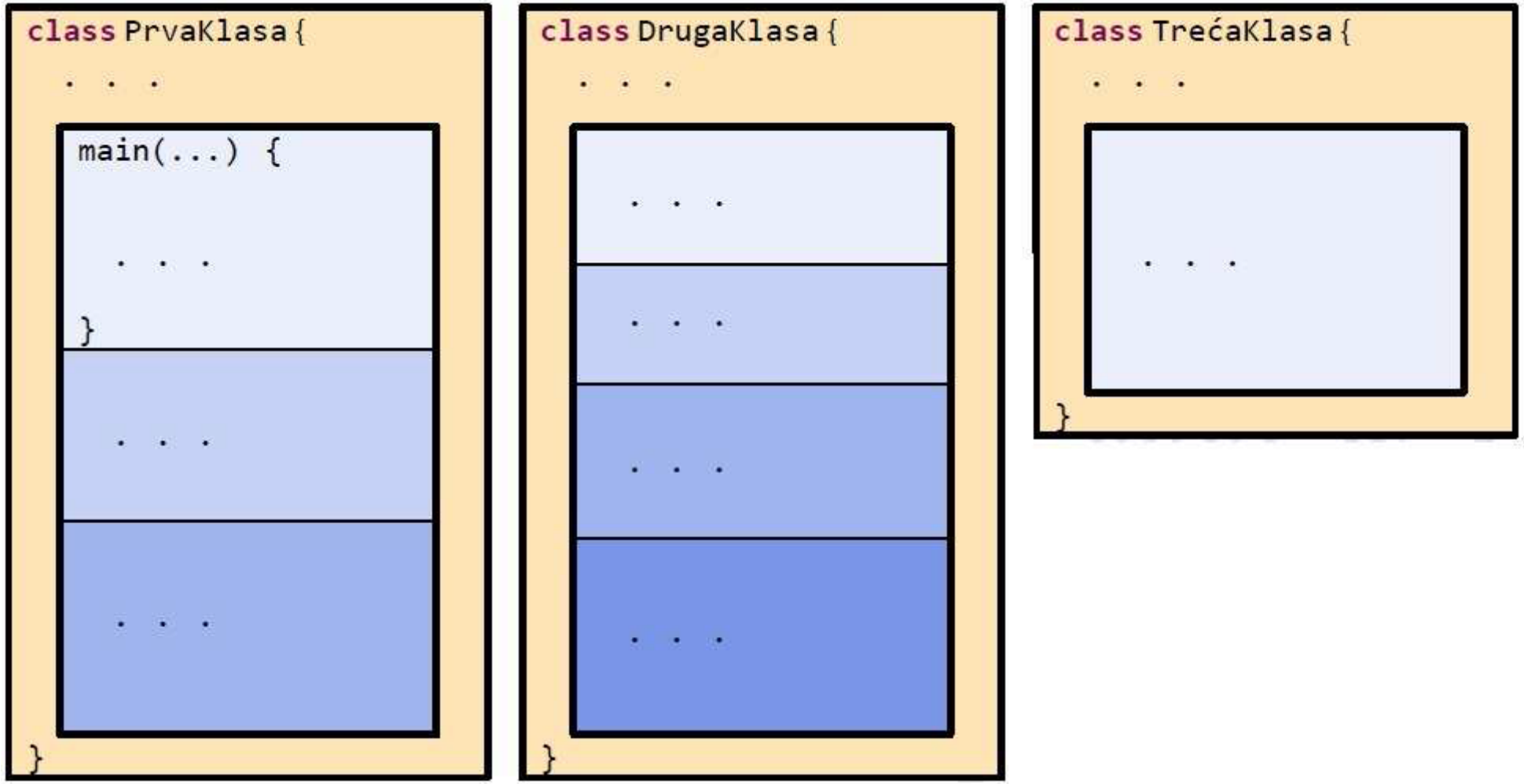
# Објектно оријентисано програмирање

- Нестатичке (објектне) методе логички припадају инстанцираним објектима, а не класи и могу се примењивати само инстанциран објекат.
- Иако се нестатичке методе везују за објекат, у меморији постоји само једна копија сваке нестатичке методе коју деле сви објекти одговарајуће класе.
- Било би јако "скупо" да се за сваки нови објекат креира и нова копија нестатичке методе.
- Посебан механизам омогућава да сваки пут када се позове метод, његов кôд се извршава на начин специфичан за конкретан објекат.
- Да би смо могли да приступимо и користимо нестатике чланове класе, морамо да направимо (инстанцирамо) објекат те класе.
- То смо већ увелико радили креирањем објекта класе ***Scanner***.

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

# Објектно оријентисано програмирање

- Структура једног програма у јави дефинисана објектно орјентисаном методологијом:



# Објектно оријентисано програмирање

- У ООП се класе **дефинишу** (пишу), а објекти се **праве** (инстанцирају) на основу дефиниција класа. Поново користимо пример класе Аутомобил:

```
public class Automobil {

    public static String marka;
    public static String model;
    public String boja = "crvena";
    public int brojVrata = 2;
    public int brzina = 280;

    public static String info(){
        String auto = "Vozimo automobil " + marka + " " + model;
        return  auto;
    }

    public void pokreniMotor(){
        System.out.println("Okreni ključ - pozicija START!");
    }
    public void zaustaviMotor(){
        System.out.println("Okreni ključ - pozicija STOP!");
    }
    public void dodajGas(){
        System.out.println("Pritisni papučicu gasa!");
    }
}
```

# Објектно оријентисано програмирање

```
public class GlavnaKlasa {

    public static void main(String[] args) {
        // statički članovi klase Automobil
        Automobil.marka = "Ford";
        Automobil.model = "Mistang GT";
        System.out.println(Automobil.info());

        // instanciranje objekta iz klase Automobil
        Automobil mojAuto = new Automobil();
        System.out.println("Vozimo auto boje: " +
                           mojAuto.boja);
        System.out.println("Ovaj auto ima " +
                           mojAuto.brojVrata + "-ja vrata.");
        System.out.println("Maksimalna brzina auta je: " +
                           mojAuto.brzina);
    }
}
```

За креирање објекта неке класе користимо оператор **new**, након којег можемо навести позив **конструктора** одговарајуће класе.

```
Run:  GlavnaKlasa
"C:\Program Files\Java\jdk1.8.0_121\bin\java.exe" ...
Vozimo automobil Ford Mistang GT
Vozimo auto boje: crvena
Ovaj auto ima 2-ja vrata.
Maksimalna brzina auta je: 280

Process finished with exit code 0
```

# Објектно оријентисано програмирање

- Чланови класе су приступачни у целом коду класе у којој се налазе.
- Права приступа члану из других класа се одређују модификатором приступа.
- Без модификатора приступа (подразумевано, "пакетско право"), члан је приступачан само у коду датог пакета.
- Декларација атрибута (поља) класе се састоји од три целине:
  - модификатора поља класе (опционо – може да се користи подразумеван приступ),
  - тип поља класе (прости типови, класни тип, константа),
  - име поља класе (у нашем првом примеру: **marka**, **model**, **boja**, ...).
- Вредност промењљиве, чији је тип нека класа, може бити само референца на објекат те класе или константа **null**.
- Ако промењљива има вредност **null**, тада она не показује ни на један објекат.
- Вредност **null** се може доделити промењљивој било ког референцијалног типа – нпр.

```
public String boja = null;
```

# Објектно оријентисано програмирање

## Права приступа

- **`public`**
  - Члан је приступачан са произвољног места у било којој класи дефинисаног пројекта.
- **`protected`**
  - Члан је приступачан у изведеним поткласама и у коду целог пакета.
- **`private`**
  - Члан је приступачан само у класи где је дефинисан.

```
public String ime;

protected String prezime;

private int uzrast;
```

**Пример**: Креираћемо класу Особа и дефинисати различите модификаторе приступа атрибутима класе. Проширићемо пример додавањем још једне класе, класе Студент, и разматрати права приступа атрибутима из обе класе.

# Објектно оријентисано програмирање

- Као што је то приказано у примерима, за дефинисање класе користимо кључну реч **`class`**.
- По конвенцији, имена класа у Јави почињу великим словима.
- За атрибуте, како статичке тако и нестатичке, се у дефиницији класе могу поставити иницијалне вредности (пример класе **`Automobil`**) али то није правило (пример класе **`PorodičnoStablo`**).
- Уколико се не наведе иницијална вредност, приликом креирања објекта биће придружена подразумевана вредност и то:
  - 0 - за нумеричке типове,
  - '\u0000' - за тип **`char`**,
  - **`null`** за референце на објекте и референце на низове.
- Методи су заправо функције и тако се и дефинишу.
- Тип вредности коју метод враћа може бити **`void`**, произвољан тип или класа.

# Објектно оријентисано програмирање

- Већ смо причали о овоме, методама се аргументи преносе по **вредности**.
- Дакле, за сваки аргумент који се преноси, прави се његова копија и копија се преноси методу преко **имена параметра**, а не оригинална вредност.
- Тако на пример ако је аргумент променљива примитивног типа, метода **не може променити** њену вредност.
- Иако се механизам преношења аргумената по вредности примењује на све типове аргумената, ефекат **за објекте се разликује** од оног за променљиве примитивних типова.
- Објекат се може променити јер променљива класног типа садржи референцу на објекат, а не сам објекат.
- Када се таква променљива пренесе као аргумент метода, преноси се **копија референце на објекат**, а не копија самог објекта.
- Сваки параметар метода може се навести и као `final` чиме се спречава да метод промени његову вредност, о чему ће компајлер водити рачуна.

# Објектно оријентисано програмирање

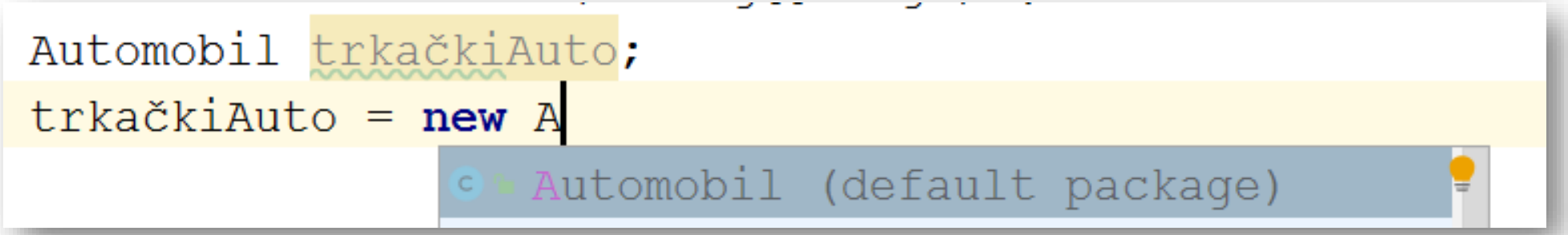
- Објекти настају динамички, у току рада програма, применом **`new`** оператора.
- Применом **`new`** оператора се алоцира меморијски простор који објекат заузима и врши се позив **конструктора** одговарајуће класе.
- У Јави се ослобађање меморијског простора заузетог од стране објеката врши аутоматски (*JVM runs the Garbage Collector program*).
- **Конструктор је специјални метод** у класи који служи за иницијализацију објекта одмах након његовог конструисања.
- Особине конструктора:
  - Име конструктора мора бити исто као име класе.
  - Конструктор нема тип резултата, чак ни **`void`**.
  - Позива се само оператором **`new`**.
- Конструктор може да не садржи ниједан параметар и онда се он назива **подразумевани конструктор** (уколико га сами не дефинишемо, компајлер ће обезбедити његово подразумевано коришћење).

# Објектно оријентисано програмирање

- Примарна сврха конструктора је да за објекат који се креира изврши иницијализацију нестатичких атрибута.
- Иницијализациони блокови који су евентуално дефинисани у класи увек се извршавају пре тела конструктора.
- Када декларишемо променљиву типа неке класе, не позива се конструктор, јер се и не креира објекат:

```
Automobil trkačkiAuto;
```

- На овом месту креира се променљива **`trkačkiAuto`** која може да чува **референцу** на објекат типа **`Automobil`**.
- Креирање објекта **`trkačkiAuto`** се врши позивом кључне речи **`new`**, а након тога позива подразумевани конструктор класе **`Automobil`**:



# Објектно оријентисано програмирање

- Како користити конструисане објекте?
- Објекти се могу користити само индиректно преко њихових референци.
- Референца објекта се мора сачувати у променљивој класног типа.

```
Student S3=new Student();
```

Позив подразумеваног конструктора

```
Student S4=new Student("Pera", 101);
```

Позив конструктора са два параметра

**Пример**: креирамо класу **`Радник`** и објекте помоћу оба типа конструктора.

- Кључна реч **`this`**.
  - Указује на имплицитни аргумент нестатичког (објектног) метода
  - Служи за позивање једног конструктора унутар другог конструктора.

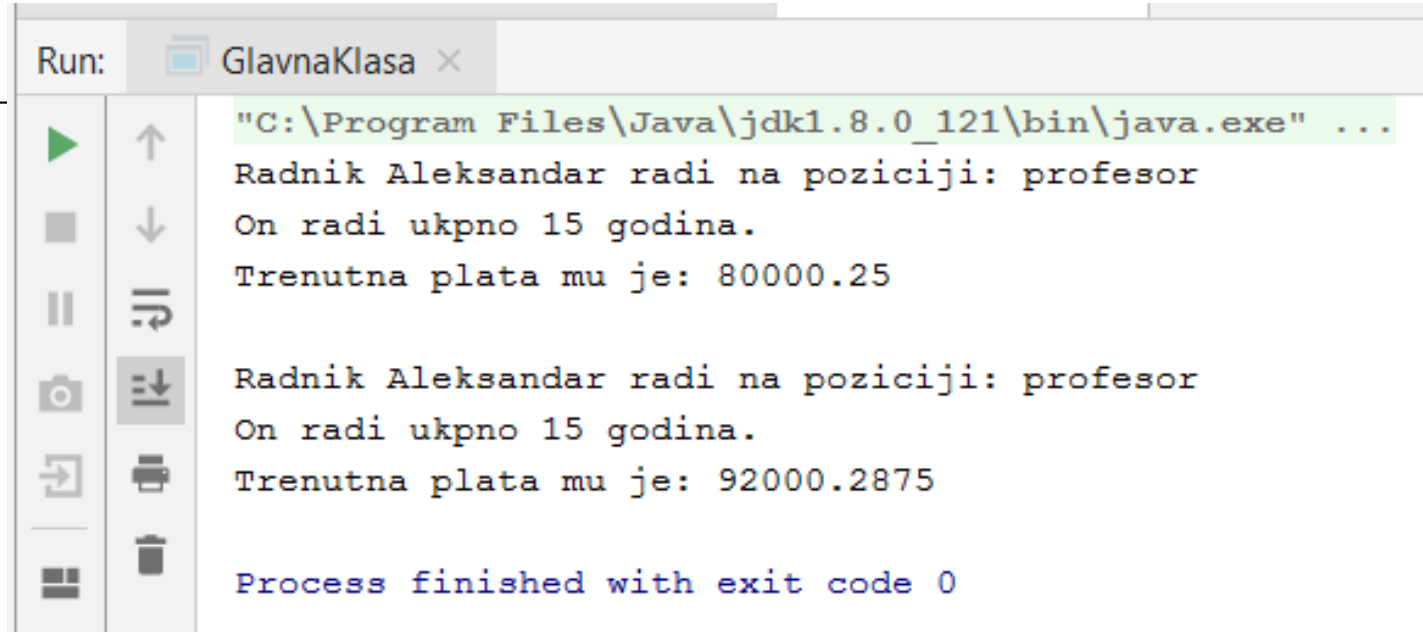
Више о кључној речи **`this`** на следећем предавању.

# Објектно оријентисано програмирање

```
public class Radnik {
    private String ime;
    private int staž;
    private double plata;
    public String pozicija;
    // Konstruktor
    public Radnik(String i, int s, double p) {
        ime = i;
        staž = s;
        plata = p;
    }
    public void povećajPlatu(double procenat) {
        plata = plata + plata * procenat / 100;
    }
    public void predstaviSe(){
        System.out.println("Radnik " + ime + " radi na poziciji: " + pozicija);
        System.out.println("On radi ukpno " + staž + " godina.");
        System.out.println("Trenutna plata mu je: " + plata);
    }
}
```

# Објектно оријентисано програмирање

```
public class GlavnaKlasa {
    public static void main(String[] args) {
        Radnik radnik = new Radnik("Aleksandar", 15, 80000.25);
        radnik.pozicija = "profesor";
        radnik.predstaviSe();
        System.out.println();
        radnik.povećajPlatu(15);
        radnik.predstaviSe();
    }
}
```



# Литература

Градиво осмог предавања :

- Поглавље 7:

https://singipedia.singidunum.ac.rs/izdanje/40716-osnove-java-programiranja

- Ако пратите препоручене видео лекције, градиво које смо обрадили у овом предавању се односи на видео лекцију 13.

https://www.youtube.com/playlist?list=PL-UTrxF0y8kK49N01V5ttb2Xaua7JfyXu

- Одабрана поглавља из књиге: Јава JDK9: Комплетан приручник
  - Аутор: *Herbert Schildt (може и старије издање JDK7).*
- Књига: Објектно оријентисани начин мишљења
  - Аутор: *Matt Weisfeld*
- Не морате да купујете наведене књиге.