JAVA PROGRAMMĒŠANAS PAMATI

Mārtiņš Ceske
Vjačeslavs Pēteris Vasiļevskis

LATVIJAS UNIVERSITĀTE
BIZNESA, VADĪBAS UN EKONOMIKAS FAKULTĀTE

VADĪBAS UN UZŅĒMĒJDARBĪBAS MĀCĪBU CENTRS

ESF projekts Nr. 8.4.1.0/16/I/001
"Nodarbināto personu profesionālās kompetences pilnveide"

NACIONĀLAIS ATTĪSTĪBAS PLĀNS 2020
EIROPAS SAVIENĪBA
Eiropas Sociālais fonds
IEGULDĪJUMS TAVĀ NĀKOTNĒ

VUMC

Java programmēšanas pamati
2021

5.Nodarbība

OBJEKTI ATMIŅĀ

VUMC

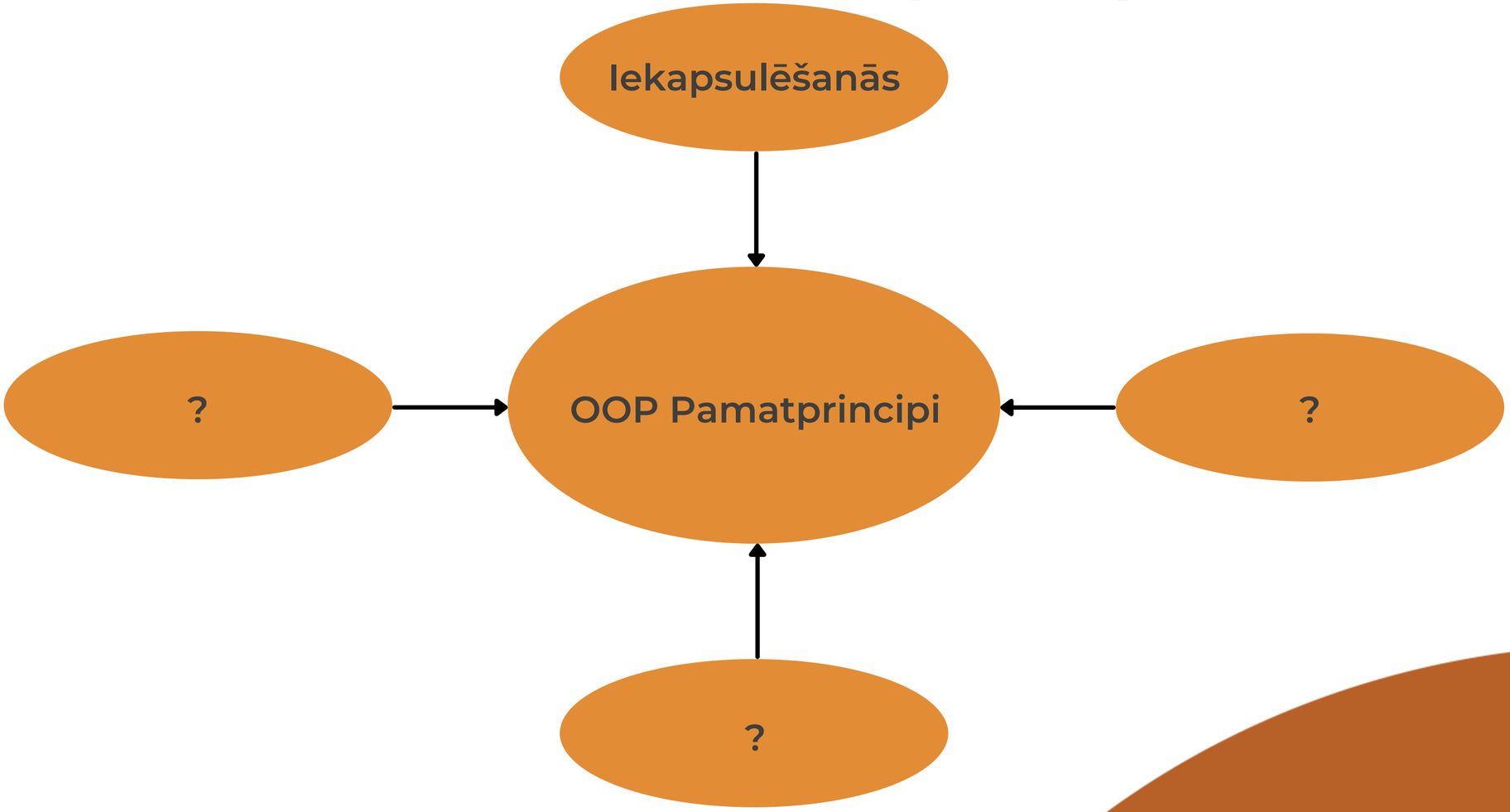
Java programmēšanas pamati
2021

OBJEKT-ORIENTĒTĀS PROGRAMMĒŠANAS PRINCIPI

OBJEKTU ORIENTĒTĀ PROGRAMMĒŠĀNA IR ĪPAŠI SLIKTA DOMA, KAS VARĒJA RASTIES TIKAI KALIFORNIJĀ

Edsger Dijkstra

ČETRI OBJEKT-ORIENTĒTĀS PROGRAMMĒŠANAS (OOP) PILĀRI



IEKAPSULĒŠANA

- Datu un uzvedības saistīšana vienā vienībā
- Piekļuve datiem notiek netieši, izmantojot klasē definētās metodes
- Īsteno datu slēpšanas principu

PIEKĻUVES MODIFIKATORI

⚠ Nosaka, kuras klases var piekļūt dotajai klasei un tās laukiem, konstruktoriem un metodēm

---

⚠ Klasēm, laukiem, konstruktoriem un metodēm varbūt viens no četriem piekļuves modifikatoriem

- *private*
- *default* (privāts pakotnei)
- *protected*
- *public*

“PRIVATE” MODIFIKATORS

⚠ Privātam elementam var piekļūt tikai kods, kas atrodas tajā pašā klasē

⚠ Deklarējamie koda elementi

- Lauki (mainīgie)
- Metodes
- Konstruktori

⚠ Neatbalstītie koda elementi

- Klases

“DEFAULT” MODIFIKATORS

⚠ Pakotnes privātam elementam var piekļūt kods no tās pašas klases vai no tās pašas pakotnes, kurā atrodas kods

---

⚠ Deklarējamie koda elementi

- Lauki (mainīgie)
- Metodes
- Konstruktori
- Klases

“PUBLIC” MODIFIKATORS

⚠ Publiskam elementem var piekļūt no jebkuras koda daļas, neatkarīgi no tā atrašānās vietas

---

⚠ Deklarējamie koda elementi

- Lauki (mainīgie)
- Metodes
- Konstruktori
- Klases

VIENKĀRŠS SKAITĪTĀJS

Stāvoklis

Tekošā skaitītāja vērtībai
nevar piekļūt tieši

Uzvedība

Skaitītāju var palielināt,
samazināt un dzēst

Var iestatīt skaitītāja vērtībai
jebkuru pozitīvu skaitli (citādi iestatīt 0)

Var tikt izveidots tikai no koda,
kas atrodas vienā pakotnē
ar skaitītāja klasi

1. VIENKĀRŠS SKAITĪTĀJS: BEZ TIEŠAS PIEKĻUVES STĀVOKLIM

Paslēpj skaitītāja iekšējo stāvokli, norādot tam “private”

```
public class BasicCounter {

    private int counter;

    public int getCounter() {
        return counter;
    }
}
```

Dod iespēju piekļūt tam no ārpuses, pievienojot “geteru” (get metodi)

2. VIENKĀRŠS SKAITĪTĀJS: PRIMĀRĀ UZVEDĪBA

```
public class BasicCounter {
    ...
    public void increment() {
        counter++;
    }
    public void decrement() {
        counter--;
    }
    public void clear() {
        counter = 0;
    }
}
```

Kontrolē skaitītāju bez tiešas piekļuves tam

3. VIENKĀRŠS SKAITĪTĀJS: SEKUNDĀRĀ UZVEDĪBA

Tikai pats skaitītājs zina par nepieciešamajām vērtības pārbaudēm

```
public class BasicCounter {
    ...
    public void setCounter(int counter) {
        if (isPositive(counter)) {
            this.counter = counter;
        } else {
            clear();
        }
    }

    private boolean isPositive(int value) {
        return value > 0;
    }

}
```

4. VIENKĀRŠS SKAITĪTĀJS: VEIDOŠANAS IEROBEŽOJUMI

Ja piekļuves modifikators nav norādīts, tad var piekļūt šim elementam kodā,kas atrodas tajā pašā pakotnē

```
public class BasicCounter {

    ...

    BasicCounter() {

    }

    ...

}
```

Tukšs konstruktors

REZULTĀTS


```
public class BasicCounter {
    private int counter;
    BasicCounter() {
    }

    public int getCounter() {
      return counter;
    }

    public void setCounter(int counter) {
      if (isPositive(counter)) {
          this.counter = counter;
      } else {
          clear();
      }
    }

    public void increment() {
      counter++;
    }

    public void decrement() {
      counter--;
    }

    public void clear() {
      counter = 0;
    }

    private boolean isPositive(int value) {
      return value > 0;
    }
}
```

VUMC

Java programmēšanas pamati
2021

OBJEKTU VIENLĪDZĪBA UN IDENTITĀTE

OBJEKTS UN *“HEAP"* ATMIŅA”

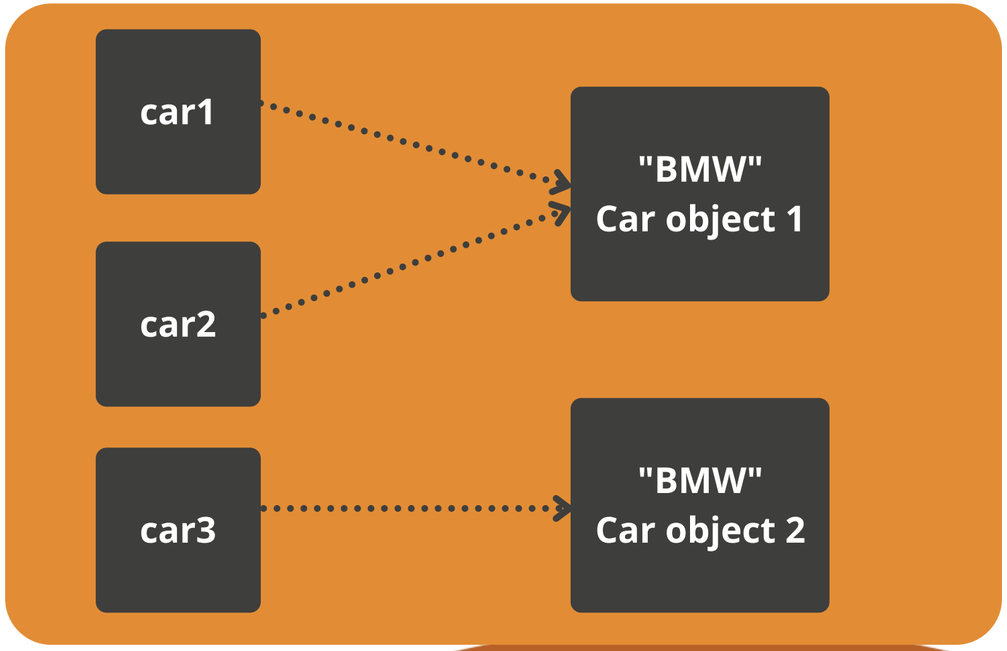
- Kad objekts tiek izveidots, tas tiek saglabāts “hīpatmiņā” (*“heap”* - kaudze)
- Dators objektam izveido adresi atmiņā, pēc kuras tas varēs atrast šo objektu
- Katram jaunizvedotam (*new*) objektam tiek piešķirta jauna adrese

NORĀDE UN OBJEKTI *“HĪPATMIŅĀ”*

Izejas kods

```
Car car1 = new Car("BMW");
Car car2 = car1;
Car car3 = new Car("BMW");
```

Objekti atmiņā



NORĀDES VIENLĪDZĪBA

Divu operandu salīdzināšanai tiek izmantots “==” operators, kas nosaka vai operandi ir vienādi savā starpā

Kad tiek izmantots ar norādes tipu, tad mēs varam redzēt vai abi mainīgie norāda uz vienu un to pašu objektu *“hīpatmiņā”*

NORĀDES VIENLĪDZĪBA: PIEMĒRS

```
Car car1 = new Car("BMW");
Car car2 = car1;
Car car3 = new Car("BMW");

if (car1 == car1) { //true, vienāds
}

if (car1 == car2) { //true, vienāds
}

if (car1 == car3) { //false, nav vienāds
}
```

LOĢISKĀ VIENLĪDZĪBA

Katrai klasei ir noklusētā *equals* (vienlīdzības) metode, kurai tiek nodots salīdzināmais objekts.

Salīdzina objektu lauku vērtības, nevis objektu norādes (adreses atmiņā)

LOĢISKĀ VIENLĪDZĪBA: PIEMĒRS


```
Car car1 = new Car("BMW");
Car car2 = car1;
Car car3 = new Car("BMW");

if (car1.equals(car1)) { //true, vienāds
}

if (car1.equals(car2)) { //true, vienāds
}

if (car1.equals(car3)) { //false, nav vienāds
}
```

TAS PATS, BET ATŠĶIRĪGS UN TOMĒR TAS PATS

```
public boolean equals(Object obj) {
    return (this == obj);
}
```

Pēc noklusējuma, tiek salīdzināts pēc norādes

Objekta klases noklusētā *"equals"* metodes implementācija

LOĢISKĀ VIENLĪDZĪBA: PĀRRAKSTI NOKLUSĒTO METODI

Tā kā noklusētās metodes implementācija neko nezina par dotās klases datiem (laukiem), tad pēc noklusējuma tiek salīdzinātas norādes

Nosaka, kādi klases dati ir jāsalīdzina un kā šie dati ir jāsalīdzina

LOĢISKĀ VIENLĪDZĪBA: METODES PĀRRAKSTĪŠANA


```
public class Car {
  private String brand;
  ...
  @Override
  public boolean equals(Object o){
    if (this == o) return true;
    if (o == null || getClass() != o.getClass()) return false;
    Car car = (Car) o;
    return Objects.equals(brand, car.brand);
  }
  ...
}
```

Pārbauda vai abi norāda uz vienu objektu

Pārbauda vai arguments nav null, un vai abi ir vienas klases instances

Pārbauda vai abi norāda uz vienu objektu

Nosaka, kuri klases lauki ir jāsalīdzina

LOĢISKĀ VIENLĪDZĪBA PĒC *"EQUALS"* PĀRRAKSTĪŠANAS


```
Car car1 = new Car("BMW");
Car car2 = car1;
Car car3 = new Car("BMW");

if (car1.equals(car1)) { //true, vienāds
}

if (car1.equals(car2)) { //true, vienāds
}

if (car1.equals(car3)) { //true, vienāds
}
```

VILTĪGS JAUTĀJUMS: *STRING* OBJEKTA IZVEIDE

Izveido *String* objektu bez atslēgvārda *new*

```
String artist = "Justin Bieber";
String band = new String("Nickelback");
```

Izveido *String* objektu ar atslēgvārdu *new*

VIENLĪDZĪBAS ATŠĶIRĪBA

Norādes vienlīdzība

```
String s1  = "Cat";
String s2 = "Cat";
String s3 = new String("Cat");
if (s1 == s2) { //true, vienads
}
if (s1 == s3) { //false, nav vienads
}
```

Loģiskā vienlīdzība

```
String s1  = "Cat";
String s2 = "Cat";
String s3 = new String("Cat");
if (s1 == s2) { //true, vienads
}
if (s1 == s3) { //true, vienads
}
```

VUMC

Java programmēšanas pamati
2021

OBJEKTA TEKSTUĀLA ATTĒLOŠANA

OBJEKTA IZVADE KONSOLĒ: GARI

Pirmkods

```
SmartPhone phone = new SmartPhone("Apple", "iPhone X");

System.out.println("Brand: " +phone.getBrand());
System.out.println("Model: " + phone.getModel());
```

Konsoles izvade

```
Brand: Apple
Model: iPhone X

Process finished with exit code 0
```

OBJEKTA IZVADE KONSOLĒ: ĪSI

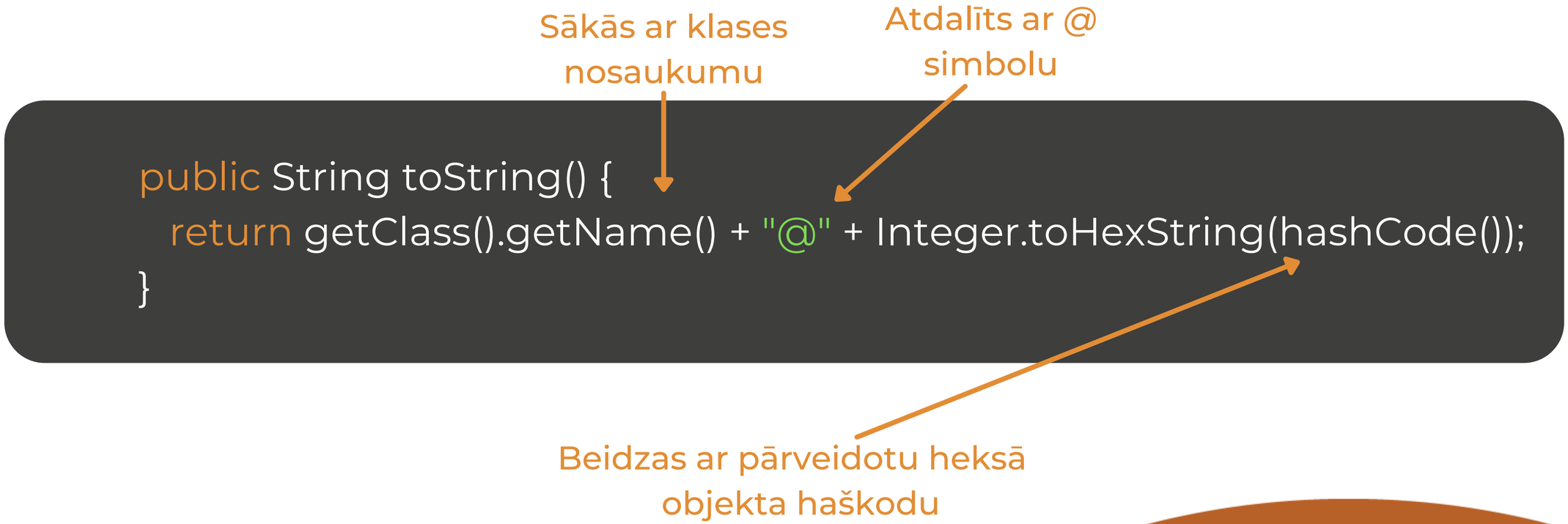
Pirmkods

```
SmartPhone phone = new SmartPhone("Apple", "iPhone X");

System.out.println(phone);
```

Konsoles izvade

```
lv.lessons.l5.SmartPhone@1540e19d

Process finished with exit code 0
```

NOKLUSĒTĀ *“toString”* METODE



PĀRRAKSTI NOKLUSĒTO *“toString”*


```
public class SmartPhone {

  private String brand;
  private String model;
  ...

  @Override
  public String toString() {
    return "SmartPhone{brand='" + brand + '\'' +  ", model='" + model + '\'' +  '}';
  }

}
```

IZVADI OBJEKTU

Pirmkods

```
SmartPhone phone = new SmartPhone("Apple", "iPhone X");

System.out.println(phone);
```

Konsoles izvade

```
SmartPhone{brand='Apple', model='iPhone X'}

Process finished with exit code 0
```

ATSAUCES

 https://dzone.com/articles/object-identity-and-equality-i java

---

 https://docs.oracle.com/javase/8/docs/api/java/lang/ Object.html#toString-

---

 https://users.soe.ucsc.edu/~eaugusti/archive/102-winter16/misc/howToOverrideEquals.html

VADĪBAS UN UZŅĒMĒJDARBĪBAS MĀCĪBU CENTRS

VUMC

Java programmēšanas pamati
2021

ESF projekts Nr. 8.4.1.0/16/I/001
"Nodarbināto personu profesionālās kompetences pilnveide"

NACIONĀLAIS ATTĪSTĪBAS PLĀNS 2020

EIROPAS SAVIENĪBA
Eiropas Sociālais fonds

IEGULDĪJUMS TAVĀ NĀKOTNĒ