Академија струковних
студија Шумадија
одсек у Крагујевцу

Увод у програмирање

Презентација 14

Академија струковних студија Шумадија

Одсек у Крагујевцу

Студијски програм Информатика

Крагујевац, 2021. година

Објектно оријентисано програмирање

Полиморфизам

- Реч полиморфизам у изворном облику, од грчких речи πολυ (много) и μορφία (облик) представља мноштво облика.
- Иако је полиморфизам повезан са наслеђивањем, често се у литератури засебно наводи и изучава.
- Појам полиморфизам подразумева могућност да се једна променљива било ког типа може користи за референцирање објеката различитих типова (наравно, ово се односи на класне типове).
- Такође, полиморфизам подразумева и позивање оног метода који је карактеристичан за тип објекта која та променљива референцира (сетимо се класе Облик).
- Захваљујући полиморфизму, исти позив методе може да се понаша другачије, у зависности од типа објекта на који се примењује.

Објектно оријентисано програмирање

- Полиморфизам функционише само са објектима изведене класе.
- Референца на објект изведене класе може да се чува у променљивој типа изведене класе али и у променљивој типа било које директне или индиректне базне класе.
- Полиморфизам се искључиво примењује на метеоде, а никако на атрибуте (поља) класе.
- Пројектовање класе за стварање потпуно различитих објеката је суштина ООП-а.
- У добро пројектованом систему, објекат би требало да "буде способан" да сам "одговори" на многа питања постављена о њему.
- Ова независност је један од основних механизама поновне употребе програмског кода.

Објектно оријентисано програмирање

- Када се објекту пошаље порука, објекат мора имати спреман и дефинисан метод који ће одговорити на поруку.
- Принцип подтипа - променљива класног типа може садржати референце на објекте свог декларисаног типа и сваког његовог подтипа.
- Принцип различитих метода које имају исто име:
  - Преоптерећење (*overload*) метода – исто име, различити потписи. Која верзија проптерећене методе се позива се одређује на основу листе параметара у позиву методе. Разрешава се у фази превођења програма - статичко (рано) везивање.
  - Нађачавање (*override*) метода – исто име, исти потпис (исти број, редослед и тип параметара) у поткласама. Одлука коју методу треба позвати се доноси у време извршавања на основу стварног типа објекта за који је метода позвана - динамичко (касно) везивање.
- Преоптерећене методе смо радили раније на предавањима, а сада посматрамо пример класе "Кућни љубимац" - променљива љубимац може током извршавања програма указивати на било који тип.

Објектно оријентисано програмирање

```
public class KućniLjubimac {
    public void predstaviSe(){
        System.out.println("Ja sam kućni ljubimac.");
    }
}
```

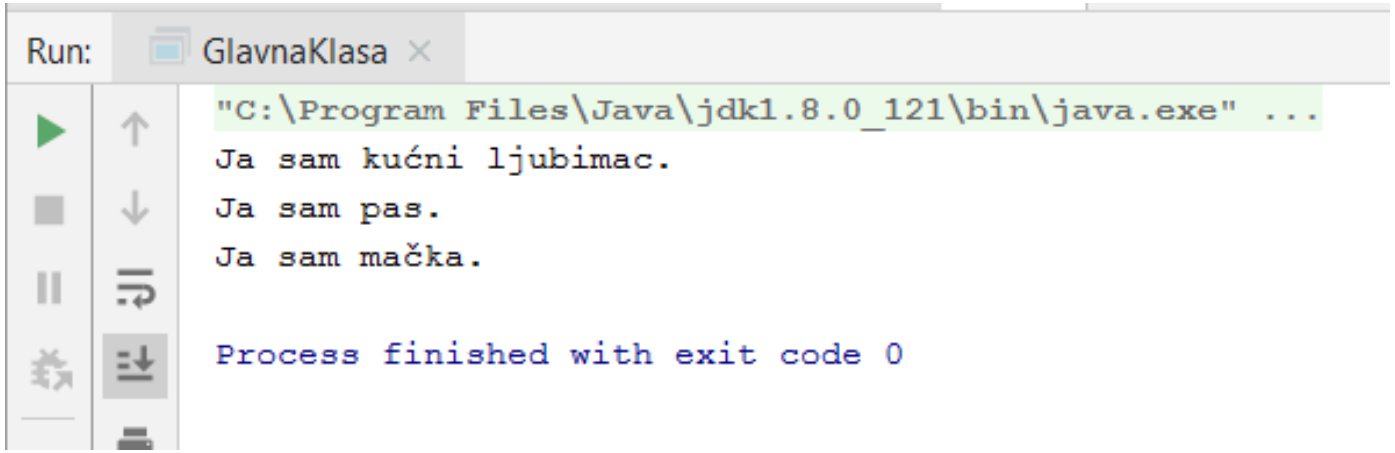
```
public class Pas extends KućniLjubimac{
    @Override
    public void predstaviSe() {
        System.out.println("Ja sam pas.");
    }
}
```

```
public class Mačka extends KućniLjubimac {
    @Override
    public void predstaviSe() {
        System.out.println("Ja sam mačka.");
    }
}
```

Објектно оријентисано програмирање

```
public class GlavnaKlasa {
    public static void main(String[] args) {
        KućniLjubimac kućniLjubimac1 = new KućniLjubimac();
        KućniLjubimac kućniLjubimac2 = new Pas();
        KućniLjubimac kućniLjubimac3 = new Mačka();

        kućniLjubimac1.predstaviSe();
        kućniLjubimac2.predstaviSe();
        kućniLjubimac3.predstaviSe();
    }
}
```



Објектно оријентисано програмирање

Динамичко везивање се обавља по једноставном правилу:

- Надјачан метод који се позива за неку класну промењљиву зависи од стварног типа објекта на који указује промењљива, без обзира на декларисан тип промењљиве.
- На ефикасан начин се може манипулисати великим бројем објеката исте хијерархије класа.
- Нпр., замислимо да има 1000 кућних љубимаца:

```
KućniLjubimac[] kućniLjubimci = new KućniLjubimac[1000];
for (KućniLjubimac kućniLjubimac : kućniLjubimci){
    kućniLjubimac.predstaviSe();
}
```

Објектно оријентисано програмирање

Интерфејси

- У Јава програмском језику једна класа може проширивати само једну класу (није дозвољено вишеструко наслеђивање класа).
- Да би се ипак некако искористиле погодности вишеструког наслеђивања у Јави се користе интерфејси.
- Интерфејс је референцијални тип података који подсећа на апстрактну класу у којој су сви методи апстрактни.
- Употребом интерфејса може се потпуно одвојити начин приступа подацима од саме реализације класе.
- Интерфејс се дефинише као и класа, једино се уместо кључне речи *class* користи кључна реч *interface*.
- Један интерфејс могу да примене више класа.
- Такође, једна класа може да примени више интерфејса.

Објектно оријентисано програмирање

- Значи, интерфејс се састоји од скупа апстрактних метода – метода без имплементације.
- Методе у интерфејсу су увек *public* и *abstract*, константе увек *public*, *static* и *final* и то се не пише експлицитно.
- Дакле, интерфејс у Јави није класа, већ скуп могућности које класе које га имплементирају морају имати.
- Сваки метод декларисан унутар интерфејса мора да има дефиницију унутар класе која имплементира интерфејс (уколико желимо да креирамо објекат те класе).
- Уколико се неки метод интерфејса не дефинише у класи која га имплементира, класа мора бити декларисана као апстрактна.
- Интерфејсом се дефинише ШТА метода треба да ради али НЕ и КАКО то треба да ради.

Објектно оријентисано програмирање

- Декларација интерфејса не имплицира никакву реализацију.
- Интерфејси подржавају динамичко разрешавање метода – у тренутку извршавања.
- Интерфејси се могу проширивати, као и класе, наслеђивањем.
- Кроз наслеђивање интерфејса могу се додати потребне методе – слично као и у наслеђивању класа.
- Интерфејс добијен наслеђивањем захтева у имплементацији реализацију свих метода:
  - метода из наслеђеног интерфејса и
  - методе декларисане у процесу наслеђивања.
- Интерфејс се може користити као тип података за референцу промењљиве!

Објектно оријентисано програмирање

- Уз дефиниције свих метода интерфејса, свака класа која имплементира интерфејс мора садржати и експлицитну декларацију да она имплементира дати интерфејс.
- Ово се постиже писањем службене речи *implements* и имена датог интерфејса иза имена класе која се дефинише.
- Иако су слични апстрактним класама, интерфејси нису класе (то је већ напоменуту и битно је да се запамти).
- Интерфејси се не могу користити за конструисање објекта.
- Дефиницијом интерфејса се у програму технички уводи нови класни тип података.
- Вредности тог класног типа су објекти класа које имплементирају дефинисани интерфејс.

Објектно оријентисано програмирање

- Ако не можемо креирати објекте из интерфејса шта можемо да урадимо?
- Користимо промењљиву за смештање референце на објекат произвољне класе која имплементира тај интерфејс.
- То значи да ту промењљиву можемо користити како бисмо полиморфно позивали методе декларисане у интерфејсу.

```
1   public interface Vozilo {
4
5       public void ubrzaj();
6
7       public void uspori();
8
9       public void promeniStepenPrenosa();
10  }
```

Објектно оријентисано програмирање

```
3   public class Bicikl implements Vozilo {
4
5       public int stepenPrenosa;
6       public int brzinaKretanja;
7
8       @Override
9       public void ubrzaj() {
10          if (brzinaKretanja < 10) {
11              System.out.println("Okreći pedale brže!");
12          } else {
13              System.out.println("Ne menjaj ništa, sve je OK!");
14          }
15      }
16
17      @Override
18      public void uspori() {
19          if (brzinaKretanja > 60) {
20              System.out.println("Postaje rizično, uspori!");
21          } else {
22              System.out.println("Ne menjaj ništa, sve je OK!");
23          }
24      }
25
26      @Override
27      public void promeniStepenPrenosa() {
28
29          if (stepenPrenosa < 1 && brzinaKretanja < 30) {
30              System.out.println("Promeni brzinu kako bi lakše ubrzao");
31          } else {
32              System.out.println("Ne menjaj ništa, sve je OK!");
33          }
34
35      }
36  }
```

Објектно оријентисано програмирање

```
3   public class Automobil implements Vozilo {
4
5       public int stepenPrenosa;
6       public int brzinaKretanja;
7
8       @Override
9       public void ubrzaj() {
10          if (brzinaKretanja < 20) {
11              System.out.println("Dodaj gas ali ne previše!");
12          } else {
13              System.out.println("Ne menjaj ništa, sve je OK!");
14          }
15      }
16
17      @Override
19      public void uspori() {
20          if (brzinaKretanja > 100) {
21              System.out.println("Postaje rizično, uspori!");
22          } else {
23              System.out.println("Ne menjaj ništa, sve je OK!");
24          }
25      }
26
27      @Override
28      public void promeniStepenPrenosa() {
29
30          if (stepenPrenosa < 2 && brzinaKretanja < 20) {
31              System.out.println("Promeni brzinu kako bi lakše ubrzao");
32          } else {
33              System.out.println("Ne menjaj ništa, sve je OK!");
34          }
35
36      }
37  }
```

Објектно оријентисано програмирање

```
3   public class GlavnaKlasa {
4
5       public static void main(String[] args) {
6           Bicikl mojBicikl = new Bicikl();
7           Automobil mojAuto = new Automobil();
8
9           mojBicikl.brzinaKretanja = 80;
10          mojBicikl.stepenPrenosa = 2;
11
12          mojAuto.brzinaKretanja = 10;
13          mojAuto.stepenPrenosa = 2;
14
15          System.out.println("Bicikl:");
16
```

```
16
17          mojBicikl.ubrzaj();
18          mojBicikl.uspori();
19          mojBicikl.promeniStepenPrenosa();
20
21          System.out.println("\n");
22
23          System.out.println("Automobil:");
24
25          mojAuto.ubrzaj();
26          mojAuto.uspori();
27          mojAuto.promeniStepenPrenosa();
28      }
29  }
```

Литература

- Одабрана поглавља из књиге: Јава JDK9: Комплетан приручник
  - Аутор: *Herbert Schildt (може и стариje издање JDK7).*
- Књига: Објектно оријентисани начин мишљења
  - Аутор: *Matt Weisfeld*
- *Предавање: ООП, др Зоран Величковић*
- *Предавање: ООП, др Дејан Живковић*