IDK2015 Objektorienteeritud programmeerimine. Martin Rebane ja Inna Švartsman 2014/2015

3. september 2014

OOP

Objektorienteeritud programmerimine võimaldab ehitada suuri tarkvarasüsteeme nii, et süsteemi osad on üksteisest sõltumatud. Kood liigendatakse erinevateks klassideks, millest igaüks tegeleb ühe loogilise osaga.

Näide: lihtne foorumi tarkvara lubab postitada foorumisse ja kasutajaks registreeruda. Sellises süsteemis võivad muuhulgas olla loodud järgmised objektitüübid: foorum, teema, postitus, kasutaja.

Java

Java programmeerimiskeel erineb paljudest teistest enamlevinud programmeerimiskeeltest selle poolest, et programmi lähtekood kompileeritakse esmalt baitkoodiks (*bytecode*). Seejärel käivitab Java virtuaalmasin (*virtual machine* ehk VM) selle ning alles siis läheb programm tööle. Praktikumis katsetasime seda. Esiteks kompileerisime Java programmi baitkoodiks käsuga:

```
javac  -Hello.java
```

seejärel käivitasime baitkoodi Java virtuaalmasina abil:

```
java Hello
```

Levinuim virtuaalmasin on Oracle Java Hotspot, mida kasutame ka enda praktikumides.

Klass ja objekt

Kui kirjutame Java koodi, siis kirjutame **klasse**. Kui virtuaalmasin meie koodi tööle paneb, tekitab ta klassidest **objektid**. Objekt on konkreetne eksemplar klassist.

Java objektid säilitavad andmeid oma sisemises olekuna ja suhtlevad omavahel, kutsudes välja meetodeid. Igal objektil on kolm olulist omadust:

tüüp – millisest klassist objekt loodud on – see võib olla Javasse sisseehitatud tüüp (nt HashMap) või enda defineeritud tüüp (nt MyClass).

olek – milline on andmete seis objektis (muutujate väärtused).

käitumine – millised meetodid sellel objektil defineeritud on.

Ühest klassist võime luua palju erinevaid objekte, milledel saab olema erinev olek, kuid sama tüüp ja käitumine. Näide erinevast olekust:

```
MyClass a = new MyClass(3);
MyClass b = new MyClass(5);
```

Samast klassist (MyClass) loodi kaks sama tüüpi erinevat objekti, andes kummagi konstruktorile ette erineva väärtuse – seega on neil ka erinev olek.

Meetod

Meetod on funktsioon, mille abil loodav objekt reaalselt midagi teeb. Objekti võimalik käitumine on piiratud selles leiduvate meetoditega. Igal meetodil on signatuur, mis kirjeldab meetodi nime

ning parameetreid, mis tuleb meetodile välja kutsudes kaasa anda (*arguments*). Signatuurile eelneb tavaliselt nähtavuse definitsioon (*modifier*) ja tagastatava väärtuse tüüp (*return type*). Näide:

```
public String getNameById(int userId) {
}
```

Selles näites on nähtavus public (avalik), meetod tagastab String-tüüpi väärtuse, meetodi nimi on getNameById ning meetodi parameetriks on int-tüüpi väärtus.

Konstruktor

Konstruktor on meetod, mis kutsutakse välja objekti loomisel automaatselt. Konstruktoril on alati sama nimi mis klassil. Ühes klassis võib olla mitu konstruktorit kui neil on erinev signatuur. Konstruktoril peab olema sama nimi, mis klassil, seega saavad erineda ainult meetodi parameetrite tüübid ja arv. Näide

```
public class MyClass() {
// 1. konstruktor
public MyClass (int myNum) {
}

// 2. konstruktor
public MyClass(String myName) {
}
}
```

Muutuja

Muutujaid kasutame info säilitamiseks. Peamiseks omaduseks on muutuja tüüp. Vastavalt muutuja tüübile saab sinna ka infot salvestada. Näiteks int tüüpi muutujasse saab salvestada ainult int tüüpi väärtust (integeri).

Muutujad võivad olla primitiivset tüüpi (Javasse sisse ehitatud) või objektid (võivad olla nii osa Javast kui kasutaja defineeritud). Primitiivsed tüübid on int, long, double, char, byte, short, boolean, float. Teatud mööndustega võib siia hulka lugeda ka Stringi.

Uue muutuja loomine:

```
// primitiivsest tüübist, ei kasutata new keywordi
int i = 7;

// objektitüübist, kasutame new keywordi
HashMap<Integer, Integer> a = new HashMap<>();
```

Getter ja setter

Getter ja setter on kokkuleppelised funktsioonid, mida kasutatakse klassimuutujale väärtuse omistamiseks või väärtuse küsimiseks. Kui meil on defineeritud muutuja *int age*, siis võiks tal olla järgmised getter ja setter:

```
// setter
public void setAge(int i) {
	age = i;
}

// getter
public int getAge() {
	return age;
}
```

Nimed on kokkuleppelised, aga hea tava on kasutada nimekuju getMuutujaNimi.