2. Konstruktorid. Isendiloome

Teemad. *Konstruktorid. Isendiloome. Muutujate ja meetodite nähtavus.*

Pärast selle praktikumi läbimist oskab üliõpilane

- luua klassitüüpi objekte ehk isendeid;
- luua ning kasutada erinevaid konstruktoreid ja isendimeetodeid;
- kasutada piiritlejaid;
- kasutada võtmesõna `this`.

Praktikumijuhend

Java klass on konstruktsioon, mis ühendab tervikuks andmed (muutujad) ja koodi (meetodid). Klassi loomisega defineeritakse uus andmetüüp, mida saab kasutada üsna analoogiliselt lihttüüpidega (`int`, `char`, ...).

Klassikirjelduse põhjal loodud isendiga seostatavaid muutujaid nimetatakse isendimuutujateks (ka isendiväljadeks). Iga isend antud klassist sisaldab eraldi oma eksemplari klassi kirjelduses loetletud muutujatest. Muutujate ja meetodite nähtavust ning kasutust saab reguleerida piiritlejatega `public`, `private` ning `protected`. Neist `public` määrab piiranguteta, `private` klassisisese ning `protected` klassi- ja selle alamklasside-sisese kasutusõiguse.

Muutujat või meetodit, mis otseselt ei seostu antud klassi isendiga, nimetatakse vastavalt klassimuutujaks või -meetodiks. Eraldamaks neid meetodeid ja muutujaid otseselt isenditega seotutest, lisatakse nende kirjelduste ette piiritleja `static`, samuti ei ole nende kasutamiseks vajalik klassi isendite olemasolu.

Isendite loomiseks kasutatakse konstruktorit. Konstruktorit võib käsitleda kui erilist meetodit kolme tunnusega:

- konstruktori nimi langeb kokku klassi nimega;
- konstruktori nime ette ei kirjutata tagastustüüpi;
- konstruktori poole pöördumine toimub käsuga `new` antud klassi isendi loomisel ja konstruktor tagastab viida loodud isendile.

Alati leidub üks eriline konstruktor – vaikekonstruktor, mille abil saab isendeid luua ka siis, kui klassis pole kirjeldatud ühtegi konstruktorit.

Olgu meil klass `Isik`, milles on kaks välja:

```
class Isik {

  String nimi;       // isendiväli isiku nime jaoks
  double pikkus;     // isendiväli isiku pikkuse jaoks
}
```

Klassi isendeid luuakse käsuga `new`:

```
Isik a = new Isik();
```

Sellisel juhul jäävad isendi loomisel muutujad `nimi` ja `pikkus` väärtustamata. Tahtes nimetatud välju algväärtustada juba isendi loomisel, peame need tegevused ette nägema konstruktoris või klassikirjelduses. Niisuguse konstruktoriga täiendatud klass `Isik` võib välja näha järgmiselt:

```
class Isik {

  String nimi;          // isendiväli isiku nime jaoks
  double pikkus = 1.7;  // isendiväli isiku pikkuse jaoks

  // konstruktor
  Isik(String isikuNimi, double isikuPikkus) {
    nimi = isikuNimi;
    pikkus = isikuPikkus;
  }
}
```

Nüüd saab isendeid luua järgmiselt:

```
Isik a = new Isik("Juhan Juurikas", 1.99);
Isik b = new Isik("Madli Mallikas", 1.55);
```

Muutujaid a ja b käsitletakse siin kui tavalisi muutujaid, kuid nende tüübiks on klass `Isik`. Neid nimetatakse viidatüüpi muutujateks, sest nad sisaldavad viita klassi isendile. Loodud isendi muutujate ja meetodite poole pöördumiseks tuleb kasutada punkti:

```
a.pikkus = 1.95; // isendi a väljale pikkus omistatakse 1.95
```

Sellise otsekasutuse saab keelata, kui isendiväljade kirjeldamisel kirjutada muutuja nime ette piiritleja `private`. Tavaliselt väärtustatakse isendimuutujaid klassi konstruktorite ja meetodite abil. Privaatse isendimuutuja väärtuse tuvastamiseks on mõeldud nn piilumeetoded, mille ainsaks ülesandeks on tagastada vastava muutuja väärtus.

Kui konstruktori või meetodi formaalsete parameetrite nimed langevad kokku isendimuutujate nimedega, siis tuleb konstruktori või meetodi sees isendimuutujatele viitamisel kirjutada nende ette võtmesõna `this`:

```
Isik(String nimi, double pikkus) {
  // isendimuutujad nimi ja pikkus saavad väärtusteks
  // konstruktori parameetrite väärtused
  this.nimi = nimi;
  this.pikkus = pikkus;
}
```

Kui on vaja anda muutujatele teatavad kindlad algväärtused, siis üks võimalus selleks on kasutada parameetriteta konstruktorit: