Downcast

Yliluokasta takaisin aliluokaksi

Kuten aikaisemmilta kursseilta muistamme, olio-tyyppisten muuttujien arvot ovat aina viittauksia johonkin muistissa olevaan luokan ilmentymään. Vaikka luokalle suoritettaisiin upcast, eli käsitellään laajempaa alaluokkaa suppeana yliluokkana, viittaa muuttuja edelleen laajempaan aliluokkaan. Tämän vuoksi yliluokka-muodossa oleva muuttujaa voidaan käsitellä myös aliluokkana, eli suorittaa downcast.

Downcast-operaation syntaksi on

```
Aliluokka uusiMuuttuja = (Aliluokka) vanhaMuuttuja;
```

Downcast käytännössä

Downcastin toimivuutta käytännössä voidaan tarkastella edellä upcastin yhteydessä esitetyn Auto-esimerkin kautta. Jos jokin ilmentymä (KilpaAuto) on tilapäisesti esitetty yliluokan muodossa (Auto), voidaan se myös muuttaa takaisin aliluokaksi (KilpaAuto).

Esimerkki

Edellisessä esimerkissä esitettiin metodi, joka tulostaa kaikkien autojen tiedot. Metodi toimii riippumatta siitä, onko parametri `Auto` vai Autosta periytetty `KilpaAuto`.

```
public void tulostaAutonTiedot(Auto auto)
{
  System.out.println("Auton merkki on " + auto.getMerkki() +
     "ja väri on " + auto.getVäri());
}
```

Oletetaan, että tiedämme metodia kirjoittaessa, että kaikki autot joiden merkki on "ferrari", ovat kilpa-autoja. Tämän tiedon perusteella voimme muuttaa `Auto`-tyyppisen parametrin `KilpaAuto`-tyyppiseksi ja tulostaa `KilpaAuton` erikoistiedot:

```
public void tulostaAutonTiedot(Auto auto)
{
  System.out.println("Auton merkki on " + auto.getMerkki() +
     "ja väri on " + auto.getVäri());

  // Jos auton merkki on ferrari, tiedämme että
  // se on kilpa-auto. Tulostetaan tällöin myös
  // huippunopeus
  if (auto.getMerkki().equals("ferrari"))
  {
    KilpaAuto ferrari = (KilpaAuto) auto;
    System.out.println("Auton huippunopeus on " +
      ferrari.getHuippuNopeus());
  }
}
```

Downcast-operaation kanssa on kuitenkin oltava varovainen. Jos annetun parametrin tyyppi olisikin `Auto` eikä `KilpaAuto`, mutta merkki olisi silti "ferrari", aiheuttaisi annettu koodi `ClassCastException`-poikkeuksen ja ohjelman suoritus kaatuisi virheeseen.

Instanceof

Luokan tyypin tunnistaminen

Edellä esitettiin, miten Auto tunnistettiin KilpaAutoksi sen merkin perusteella. Tämä ei kuitenkaan ole kovin käytännöllinen tapa. Olisi paljon parempi, jos olisi jokin automaattinen tapa, millä näkisi viittaako annettu Auto-tyyppinen parametri itse asiassa KilpaAuto-tyyppiseen muuttujaan. Ratkaisu tähän ongelmaan on `instanceof` operaattori.

Instanceof operaattorilla voidaan tarkistaa viittaako annettu muuttuja johonkin tietyn tyyppiseen muuttujan. Operaattorin syntaksi on `muuttuja instanceof LuokanNimi`.

Esimerkki

Tarkistetaan `tulostaAutonTiedot`-metodissa onko annettu muuttuja normaali `Auto`-luokan ilmentymä vai onko se `KilpaAuto`-luokan ilmentymä:

```
public void tulostaAutonTiedot(Auto auto)
{
  System.out.println("Auton merkki on " + auto.getMerkki() +
    " ja väri on " + auto.getVäri());

  // Tutkitaan viittaako auto-muuttuja KilpaAuto-
  // luokan ilmentymään
  if (auto instanceof KilpaAuto)
  {
    KilpaAuto ferrari= (KilpaAuto) auto;
    System.out.println("Auton huippunopeus on " +
      ferrari.getHuippuNopeus());
  }
}
```

Käyttötarkoitus

Mihin upcastia ja downcastia tarvitaan?

Esimerkissä esitettiin, miten yhdellä metodilla saadaan tulostettua kaikkien Auto-tyyppisten luokkien tiedot riippumatta siitä, ovatko ne Autoja, KilpaAutoja, KuormaAutoja tms. Auto-luokasta periytettyjä tietotyyppejä. Ilman down- ja upcast