6. Ilmentymä- ja luokkapiirteet

Sisällys

- Ilmentymä- ja luokka-attribuutit.
- Ilmentymä- ja luokkametodit.
- Metodien ja muun luokan sisällön järjestäminen.

Ilmentymä- ja luokka-attribuutit

- Oliokielessä voidaan yleensä valita ovatko attribuutin arvot oliokohtaisia tai kaikille luokan olioille yhteisiä.
- Javan normaalit attribuutit ovat ilmentymä-attribuutteja (instance variable), joiden arvo on oliokohtainen.
- **Static**-määreellä esitellyn attribuutin eli luokka-attribuutin (class variable) arvon omistaa luokka.
  - Oliot jakavat luokka-attribuutin arvon.
  - Olion tekemä muutos attribuutin arvoon näkyy kaikille olioille.
  - Julkista luokka-attribuuttia voidaan käyttää luokan ulkopuolella suoraan luokan nimen kautta ilman, että on tarpeen luoda oliota.

Luokka-attribuutit

- Esitellään attribuutti, jonka arvo on yhteinen kaikille kissoille. Haukottelu tarttuu.
  - **private static boolean** haukottelee;
- Luokan ulkopuolella tarvittavat attribuutit esitellään usein **static**-määreellä, jotta tunnus voidaan antaa ilman, että on tarvetta luoda olio.
  - // Kissa-luokassa määritelty julkinen vakioitu luokka-attribuutti.

    public static final int MAXELOT = 9;
  - // Luokan nimen kautta viitattu vakio kissaa testaavassa luokassa.

    System.out.println(… + Kissa.MAXELOT + …);

Luokka-attribuutit

- Kissa rontti = **new** Kissa("musta", "tavallinen");
  Kissa ville = **new** Kissa("mustavalkoinen", "levoton");
  // Kumpikaan kissa ei haukottele.
  System.out.println(rontti.haukottelee()); // false
  System.out.println(ville.haukottelee()); // false
  // Kun Rontti alkaa haukottelemaan, haukottelee myös Ville.
  rontti.haukottelee(**true**);
  System.out.println(rontti.haukottelee()); // true
  System.out.println(ville.haukottelee()); // true

Luokka-attribuutit

- Käytä tällä kurssilla vain vakioituja luokka-attribuutteja.
  - Mitä enemmän luokka-attribuutteja käytetään, sitä vähemmän oliomaiseksi ohjelma muuttuu, koska oliokohtaisten tietojen määrä vähenee.
- Vahingossa tehty luokka-attribuutti on tyypillinen ohjelmointivirhe.
  - Tutki ohjelmasi vakioimattomat attribuutit **static**-määreiden varalta, jos olion tila vaikuttaa muuttuvan ilman, että sen arvoja on asetettu.

Ilmentymä- ja luokkametodit

- **Static**-määre vaikuttaa metodiin erityisesti olio-ohjelmoinnissa keskeisessä periytymismekanismissa.
- **Static**-määreellä esitellyn metodin eli luokkametodin (class method) omistaa luokka, jolloin metodin periytyminen on tehokkaampaa, mutta vähemmän ilmaisuvoimaista.
- Olio-ohjelmoinnissa käytetään pääosin tavallisia metodeja eli ilmentymämetodeja (instance method), koska niitä voidaan tarkentaa ja käyttää periytymisen yhteydessä luokkametodeja joustavammin.
  - Tee luokkametodi vain, jos metodin käyttö ilman oliota vaikuttaa luontevalta nyt ja tulevaisuudessa.
  - Periytymisestä ja metodin tarkentamisesta kerrotaan pian lisää.

Luokkametodit

- Kutsuttavissa ilman olion luomista suoraan luokan nimen kautta.
- Luokkametodeja voidaan kutsua myös olion kautta, koska oliolla on aina luokkansa piirteet.
  - Ilmentymämetodien kutsuminen on mahdollista vain olion kautta.

- // Math-luokassa.

  public static int max(**int** a, **int** b)
- // Kutsu Math-luokan kautta.

  int suurin = Math.max(1, 2);

Luokkametodit

- Luokkametodissa voidaan käsitellä vain luokka-attribuutteja ja kutsua luokkametodeja, koska ilmentymäpiirteet eivät ole saatavilla ilman oliota.
- Ajoluokan metodeissa on käytettävä **static**-määrettä, jos ohjelmoidaan rakenteisesti Lausekielinen ohjelmointi II – kurssin tapaan, koska *main*-metodi on luokkametodi.

Luokkapiirteet

- Olio-ohjelmoinnissa **static**-määrettä tarvitaan metodeissa lähinnä, kun
  - kootaan yhteen toimintoja, joiden käyttö on sujuvampaa ilman olion esittelyä (esimerkiksi *Math*-luokka),
  - halutaan määritellä metodi, jota ei ole tarvetta määritellä uudelleen tarkemmin ja jonka toiminta riippuu siten ainoastaan sen saamista parametreista,
  - halutaan optimoida ohjelmaa.
- Kurssilla luokkametodeja tarvitaan vain, kun kirjoitetaan metodeja ajoluokkaan tai luokasta muotoutuu pakottavasta syystä metodien kokoelma ilman attribuutteja.
  - Tarpeeton **static**-määreen käyttö metodeissa on huonoa olio-ohjelmointia.

Luokan sisällön järjestäminen

- Yleensä luokan sisältö järjestetään jollakin tavoin, koska halutaan tehdä yhdenmukaista koodia.
- Attribuutit ennen metodeja, koska olio-ohjelmoinnissa tiedot ovat keskiössä.
- Esimerkkijärjestys:
  - Vakiomuotoiset attribuutit.
  - Luokka-attribuutit.
  - Ilmentymäattribuutit.
  - Rakentajat.
  - Luokkametodit.
  - Ilmentymämetodit.
  - *Main*-metodi.

Luokan sisällön järjestäminen

- Rakentajat aina ennen muita metodeja.
- Muut metodit:
  - Java-kielen kehittäjät suosittelivat järjestämään toiminnallisuuden mukaan: yhteenkuuluvat osat koodia lähellä toisiaan.
  - Näkyvyyden mukaan ryhmitellen.
  - Aakkosjärjestys.
- Yrityksillä omat yksityiskohtaiset tyylioppaansa.
- Oli järjestys mikä tahansa, tärkeintä on johdonmukaisuus!