4. Luokan testaus ja käyttö olion kautta

Olion luominen luokasta

- Java-kielessä olio määritellään joko luokan edustajaksi tai taulukoksi.
- Olio on joukko keskusmuistissa olevia tietoja.
- Oliota käsitellään siihen epäsuorasti liittyvän viitetyyppisen muuttujan eli viitteen (reference) avulla.
- Viite määrittää olion identiteetin.
- Viite toteutetaan teknisesti yhden tai useamman suojatun osoittimen (pointer) avulla.
  - Javan viitteet eri asia kuin osoittimet ja C++:n viitteet.

Olion luominen luokasta

- Tunnuksen esittely varaa muistia viitteelle (viitetyyppiselle muuttujalle), mutta ei oliolle.
- Java alustaa automaattisesti viitteen tyhjäksi (null-arvo) tai ilmoittaa, että viite on alustettava.
- Oliolle varataan muistia new-operaatiolla, joka palauttaa viitteen.
- Luokkatyyppisen olion luomisen yhteydessä kutsutaan rakentajaa, joka on erityinen metodi, jonka avulla määrätään mistä luokasta olio luodaan ja kuinka attribuutit alustetaan.

Olion luominen luokasta

- Java luo luokalle automaattisesti tyhjän parametrittoman oletusrakentajan ja antaa attribuuteille niiden tyyppien määräämät alkuarvot.
  - Rakentajia käsitellään myöhemmin tarkemmin.
- Muuttuja ja olio liittyvät toisiinsa, kun new-operaation paluuarvona saatava viite sijoitetaan muuttujan arvoksi.
- Luokkatyyppisen olion metodeja kutsutaan muuttujan ja pistenotaation avulla.
  - String merkit = "abc";

    int merkkeja = merkit.length();
  - System.out.println(rontti.väri());

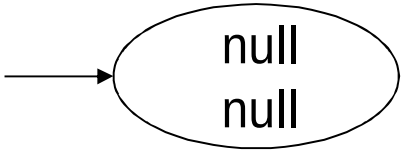
Olion luominen luokasta

- Kissa rontti = null;

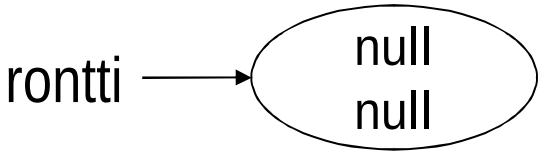
rontti ⟶ null

Olion viitteen esittely ja alustaminen tyhjäksi.

- rontti = new Kissa();



Lauseke new Kissa(); luo olion *Kissa*-luokasta, alustaa sen attribuutit ja palauttaa paluuarvona olioon liittyvän tunnuksettoman viitteen.



Sijoituksen seurauksena *rontti*-viite viittaa samaan olioon kuin paluuarvona saatu viite.

Luokalle oma tiedosto

- Ohjelma on toistaiseksi ajateltu yhdeksi luokaksi.
- Näin lähdekooditiedostojakin on ollut vain yksi.
- Siirrytään nyt tyypilliseen käytäntöön, jossa jokaisen luokan koodi erotetaan omaan tiedostoonsa.
- Näin toimien kukin luokka on selkeämmin täysin oma kokonaisuutensa myös tiedostojen tasolla.
- *Main*-metodin sisältävää luokkaa kutsutaan ajoluokaksi.
  - Käytetään tällä kurssilla usein toisen luokan testaamiseen.
  - Esimerkiksi *Kissa*-luokkaa (Kissa.java) testataan *KissaTesti*-luokassa (KissaTesti.java).

KissaTesti-luokka (KissaTesti.java)

```
public class KissaTesti {
   public static void main(String[] args) {
      // Viitteen esittely, muistinvaraus ja viitteen yhdistäminen
      // olioon yhdessä lauseessa.
      Kissa rontti = new Kissa();
      // Testataan metodeja kutsumalla niitä olion kautta pistenotaatiolla.
      rontti.äänteleä("Miau!");
      rontti.väri("musta");
      rontti.häntä("tavallinen");
      String rontinVäri = rontti.väri();
      System.out.println(rontinVäri);
      String rontinHäntä = rontti.häntä();
      System.out.println(rontinHäntä);
      …
   }
}
```

Testiluokan kääntäminen ja ajaminen

- Testiluokka on käännettävä yhdessä testattavan luokan kanssa. Tämä on tehtävissä eri tavoin.
- Kun molempien luokkien lähdekooditiedostot sijoitetaan samaan hakemistoon, kääntäjälle tarvitsee antaa vain testiluokan sisältävän tiedoston nimi.
  - javac KissaTesti.java
- Ohjelma ajetaan testiluokan nimellä.
  - java KissaTesti

Testiluokan kääntäminen ja ajaminen

- Luokan voi myös sisällyttää testiluokan käännökseen jostakin muusta hakemistosta joko polkumäärittelyllä tai *javac*-ohjelman *sourcepath*-parametrillä.
- Näin käännetty ohjelma suoritetaan siten, että *java*-ohjelmalle kerrotaan *classpath*-parametrin avulla tavukoodin hakemisto.