15. Viitteiden taulukot

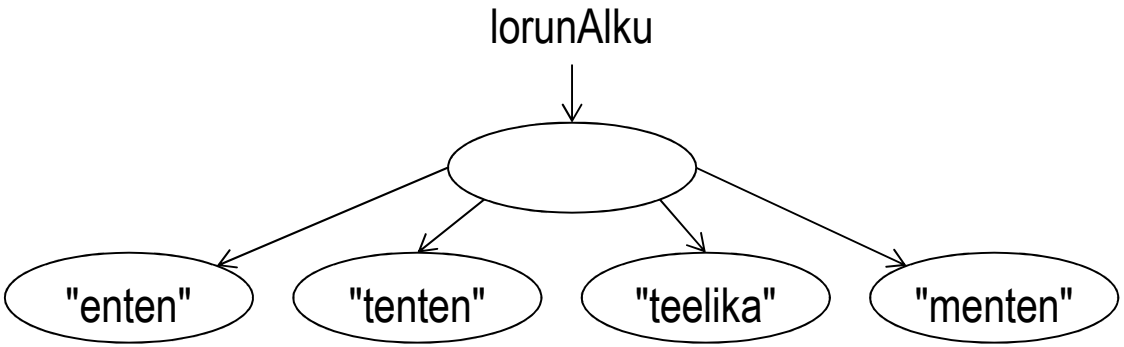
Sisällys

- Viitteet taulukon alkioina.
- Kääreluokat.
- Automaattinen käärintä ja purku.

Viitteet taulukon alkioina

- Taulukko voi koostua viitetyyppisistä alkioista.
  - Taulukon alkio eli viite liittyy olioon tai **null**-arvoon.
  - Alustamaton alkio liittyy oletus-arvoisesti **null**-arvoon.
- *String*-tyyppisistä viitteistä koostuvaa taulukkoa on käytetty jo aiemmalla kurssilla ilman tarkempaa pohdintaa.

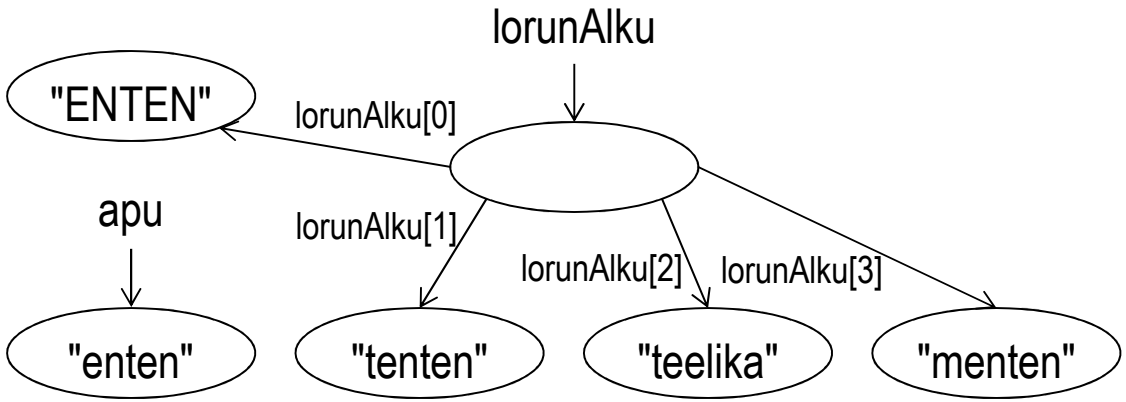
```
// Java luo kustakin merkkijonoliteraalista
// automaattisesti olion ja liittää kunkin
// viiteen vastaavaan olioon.
String[] lorunAlku = {"enten", "tenten",
"teelika", "menten"};
```



Viitteet taulukon alkioina

- Tietty alkio yksilöitään tutulla [ ]-notaatiolla, joka palauttaa nyt tiedon asemasta viitteen, jonka kautta päästään tarvittaessa käsittelemään viitteeseen liittyvää oliota (tietoa).
- Alkioiden sijoitus- ja vertailu on viitteiden sijoitusta ja vertailua.

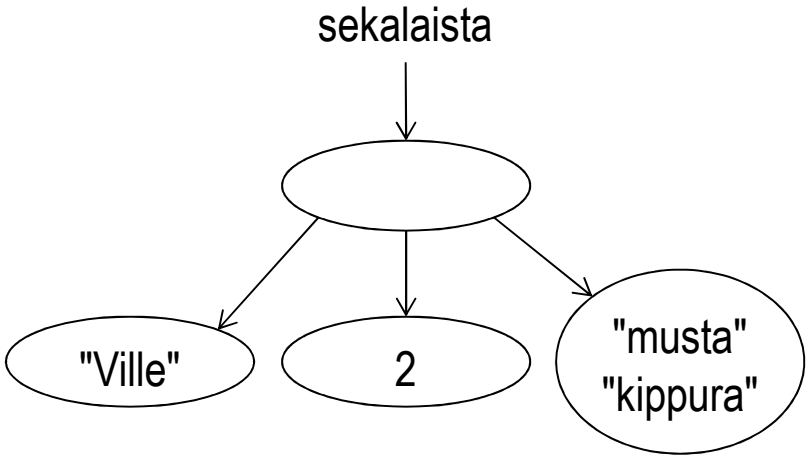
```
String[] lorunAlku = {"enten", "tenten",
"teelika", "menten"};
// Sijoitetaan ensimmäiseen viitteeseen
// liittyvään olioon uusi viite.
String apu = lorunAlku[0];
// Liitetään ensimmäinen viite uuteen olioon.
lorunAlku[0] = "ENTEN";
// Tarkistetaan, että taulukon alkuperäinen
// ensimmäinen olio on turvassa ja että
// ensimmäiseen alkioon liittyy uusi olio.
System.out.println(apu); // enten
System.out.println(lorunAlku[0]); // ENTEN
```



Viitteet taulukon alkioina

- Monimuotoisuus ilmenee myös siten, että taulukkoon voidaan lisätä alkion tyypin kanssa yhteensopivia viitteitä, jolloin taulukko voi omistaa erilaisia olioita.
- *Object*-tyyppisten viitteiden taulukkoon voi sijoittaa minkä tahansa tyyppisen viitteen ja samalla taulukosta voi olla viite minkä tahansa tyyppiseen olioon.

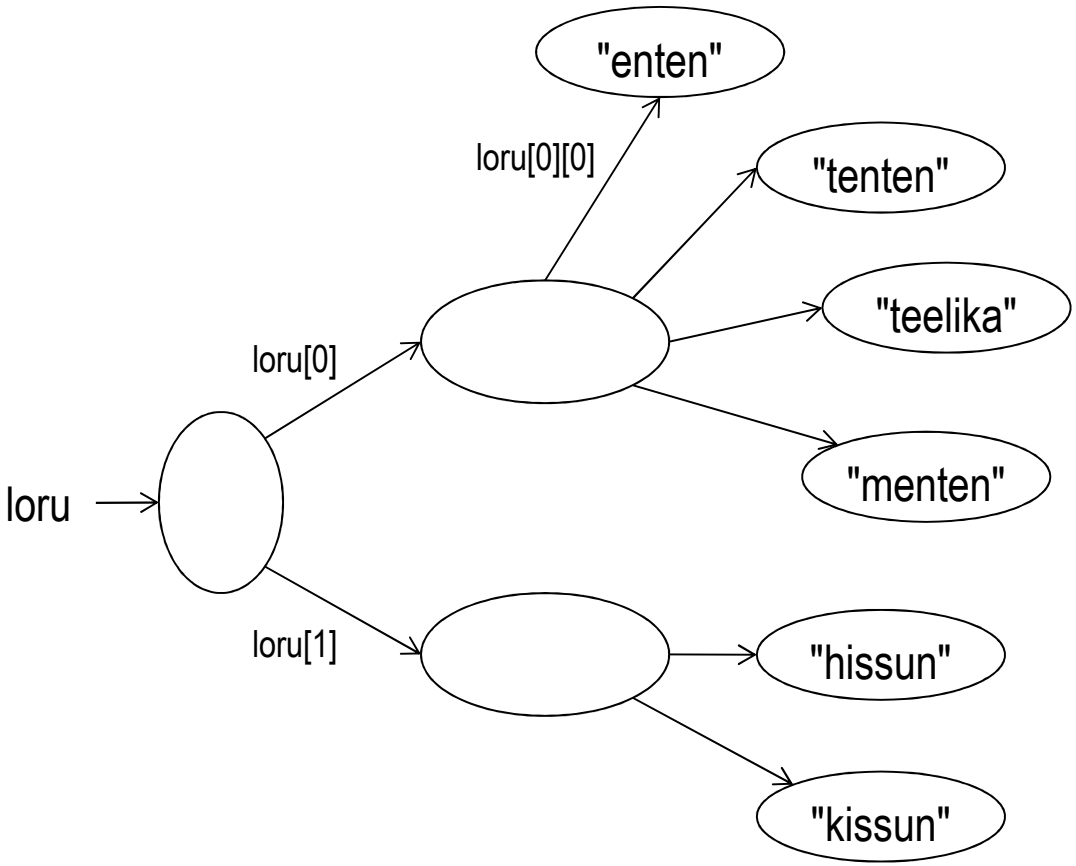
```
String nimi = new String("Ville");
Integer ikä = Integer.valueOf(2);
Kissa ville = new Kissa();
// Alustetaan taulukko luettelemalla
// siihen tulevat viitteet.
Object[] sekalaista = { nimi, ikä, ville };
```



Viitteet taulukon alkioina

- Kaksiulotteisen taulukon rivit ovat yksiulotteisia taulukoita, joihin kaksiulotteinen taulukko-olio viittaa.
  - Riveittäistä käsittelyä voidaan tehostaa viitteiden avulla.
- Taulukon rivien ei ole pakko olla saman mittaisia.

```
String[][] loru = {{"enten", "tenten", "teelika",
"menten"}, {"hissun", "kissun"}};
```



Käärеluokat

- Javan alkeistyypeillä on vastaavat käärеluokat.
  - **boolean** ja *Boolean*
  - **byte** ja *Byte*
  - **char** ja *Character*
  - **double** ja *Double*
  - **float** ja *Float*
  - **int** ja *Integer*,
  - **long** ja *Long*
  - **short** ja *Short*
- Käärеluokkia käyttämällä voidaan tarvittaessa ohjelmoida täysin oliomaisesti.
- Käärеluokat ovat osa *java.lang*-pakkausta ja siten aina käytettävissä ilman **import**-lausetta.

Automaattinen käärintä ja purku

- Java muuntaa alkeistyyppisen arvon automaattisesti olioksi (autoboxing), kun arvo sijoitetaan viitetyyppiseen muuttujaan, jonka tyyppi vastaa alkeistyyppiä.
- Purku kääreluokan oliosta alkeistyyppiseksi arvoksi (unboxing) tapahtuu sijoittamalla olioon liittyvä viite alkeistyyppiseen muuttujaan, jonka tyyppi vastaa viitetyyppiä.

```
// Esitellään viite, luodaan
// käärimällä Integer-tyyppinen
// olio, jonka sisältää tiedon 2
// ja liitetään viite olioon.
Integer käärittyIkä = 2;
// Olio voidaan luoda tarvittaessa
// myös "tehdasmetodilla".
Integer uusiIkä = Integer.valueOf(3);
// Kääretyyppien rakentajat
// vanhenivat Javan versiossa 9.
// Tuore kääntäjä varoittaa alla
// annetusta lauseesta.
Integer huonoIkä = new Integer(20);
// Puretaan olion arvo alkeisarvoksi.
int purettuIkä = käärittyIkä;
```

- Käärintä ja purku toimii vain alkeistyyppien ja niiden kääreluokkien yhteydessä.

Automaattinen käärintä ja purku

- Automaattiset muunnokset nopeuttavat saman kääreluokan viitteistä koostuvien taulukoiden käsittelyä.
  - Olioiden luomisia ja tyyppimuunnoksia tarvitaan vähemmän.
- Käärimistä ja purkamista voidaan hyödyntää myös Javan kokoelmaluokissa.
- Hidastaa ohjelmaa.

```
// Pitemmin lausuen.
Integer[] luvut = {Integer.valueOf(1),
Integer.valueOf(2), Integer.valueOf(3)};
int eka = luvut[0].intValue();
luvut[2] = Integer.valueOf(4);
// Sama lyhemmin lausuen.
Integer[] luvut = {1, 2, 3};
int eka = luvut[0];
luvut[2] = 4;
```