

Map

## Javan kokoelmat

- Olemme toistaiseksi käyttäneet listoja tai taulukoita, kun olemme halunneet käsitellä useita saman tyyppisiä asioita.
- Javassa on myös muita **tietorakenteita** tiedon varastointiin ja käsittelyyn.
- Java Collections Framework sisältää mm:
  - **List**, esim. ArrayList
  - **Set**
  - **Map**
  - Muuta: Queue, Stack, ym jota ei käsitellä

*Kokoelma on yksinkertaisesti olio, joka kokoaa alkioita yhteen.* <sup>1</sup>

<sup>1</sup> "A collection — sometimes called a container — is simply an object that groups multiple elements into a single unit"

<https://docs.oracle.com/javase/tutorial/collections/intro/index.html>

## Map eli hajautustaulu

Datan tallentaminen avainten ja arvojen pareina

## Analogia: JSON-tiedostomuoto

```
[{  
  "etunimi": "Lindsey",  
  "sukunimi": "Drillingcourt",  
  "email": "ldrillingcourt0@so-net.ne.jp",  
  "puhelin": "132-414-7730"  
}, {  
  "etunimi": "Zilvia",  
  "sukunimi": "Zamboniari",  
  "email": "zzamboniari1@dell.com",  
  "puhelin": "445-276-2785"  
}, {  
  "etunimi": "Moses",  
  "sukunimi": "Fairhead",  
  "email": null,  
  "puhelin": "681-240-4656"  
}, {  
  "etunimi": "Devondra",  
  "sukunimi": "Bridywater",  
  "email": "cyberchimps.com",  
  "puhelin": "306-422-3408"  
}]
```

## Map-tietorakenteet

- **Map** on eräs ohjelmoinnissa paljon käytetyistä tietorakenteista.
- **Map**:ia käytetään kun halutaan käsitellä tietoa avain-arvo –pareina. Avaimen perusteella voidaan **lisätä**, **hakea** ja **poistaa** avaimen liittyvä arvo.
- Siis toisin kuin listoissa, arvoja ei käsitellä numeeristen indeksien avulla, vaan voimme määritellä avaimiksi esimerkiksi merkkijonoja.

avain      arvo  
"00710" → "Helsinki"  
"90014" → "Oulu"

"33720" → "Tampere"  
"33014" → "Tampere"

```
import java.util.HashMap;  
  
// ...  
  
HashMap<String, String> postinumerot =  
    new HashMap<String, String>();  
  
postinumerot.put("00710", "Helsinki");  
postinumerot.put("90014", "Oulu");  
postinumerot.put("33720", "Tampere");  
postinumerot.put("33014", "Tampere");
```

## Mapin tyypin määrittely

- Hajautustaulua luodessa tarvitaan kaksi tyyppiparametria:
  - avainmuuttujan tyyppi ja
  - lisättävän arvon tyyppi.
- Tyyppiparametrit määritellään kulmasulkeisiin pilkulla eroteltuna
- Viereisessä esimerkissä sekä avainmuuttujan että lisättävän arvon tyyppi on **String**

Muuttujan tyyppinä voidaan käyttää joko rajapintaa **Map** tai luokkaa **HashMap**.

Kulmasuluissa ensimmäinen tyyppi on **avaimen tyyppi**, toinen **tallennettavien arvojen tyyppi**.

```
Map<String, String> tietovarasto =  
    new HashMap<String, String>();  
  
tietovarasto.put("avain", "arvo");  
  
String merkkijono = tietovarasto.get("avain");  
  
System.out.println(merkkijono);
```

## Avainten ja arvojen hakeminen Map:ista

Map:eihin voidaan tallentaa ainoastaan viittaustyyppisiä arvoja. Siksi int-tyypin sijaan käytetään Integer-tyyppiä.

```
Map<String, Integer> opintopisteet = new HashMap<String, Integer>();
```

```
// Lisätään arvoja tietyille avaimille:
```

```
opintopisteet.put("swd1tn001", 5);
```

```
opintopisteet.put("swd1tn002", 5);
```

```
// Haetaan yksi arvo:
```

```
int pisteet = opintopisteet.get("swd1tn002");
```

```
System.out.println(pisteet); // 5
```

```
// Haetaan kaikki avaimet:
```

```
Set<String> avaimet = opintopisteet.keySet();
```

```
System.out.println(avaimet); // [swd1tn002, swd1tn001]
```

## Koodaustehtävä: lempinimet

- Tässä tehtävässä sinun tehtäväsi on luoda `TreeMap<String, String>` ja lisätä siihen henkilöiden nimiä ja lempinimiä.
- Henkilön nimi toimii avaimena ja lempinimi arvona. Lisää nimet kokoelmaan oikealla esitetyssä järjestyksessä.
- Lopuksi tulosta kokoelma sellaisenaan.

Nimi (avain)	Lempinimi (arvo)
Teemu	The Finnish Flash
Jari	Litti
Kaisa-Leena	Kappa

```
{Jari=Litti, Kaisa-Leena=Kappa, Teemu=The  
Finnish Flash}
```



Mitä jos lisäämme arvon olemassa olevalle avaimelle?

Mitä jos haemme arvoa avaimella, jota Mapista ei löydy?

## Uuden arvon asettaminen

- Hajautustaulussa on jokaista avainta kohden korkeintaan yksi arvo.
- Jos hajautustauluun lisätään uusi avain-arvo -pari, missä avain on jo aiemmin liittynyt toiseen hajautustauluun tallennettuun arvoon, vanha arvo katoaa hajautustaulusta.

```
Map<String, String> numerot = new HashMap<>();
numerot.put("Uno", "Yksi");
numerot.put("Dos", "Zwei");
numerot.put("Uno", "Ein");

String kaannos = numerot.get("Uno");

System.out.println(kaannos);
```

## Arvojen poistaminen tai tarkastaminen (remove ja containsKey)

```
HashMap<String, String> countries = new HashMap<>();

countries.put("Suomi", "Finland");
countries.put("Ruotsi", "Sweden");
countries.put("Norja", "Norway");

countries.containsKey("Ruotsi"); // true

countries.remove("Ruotsi");

countries.containsKey("Ruotsi"); // false
```

## Null-viittaukset mapilta

- Jos mapista haetaan arvoa avaimella, jota ei löydy, palautuu tuloksena `null`-arvo
- Mikäli `null`-arvon sijasta halutaan käyttää jotain toista arvoa oletusarvona, voidaan käyttää `getOrDefault`-metodia
- Oheisessa esimerkissä ohjelma kaatuisi, mikäli oletusarvoa ei olisi annettu:
  - `null` -arvoa ei voida sijoittaa `int` -tyyppiseen muuttujaan!

```
Map<String, Integer> pistelaskuri = new HashMap<>();
pistelaskuri.put("Matti", 10);

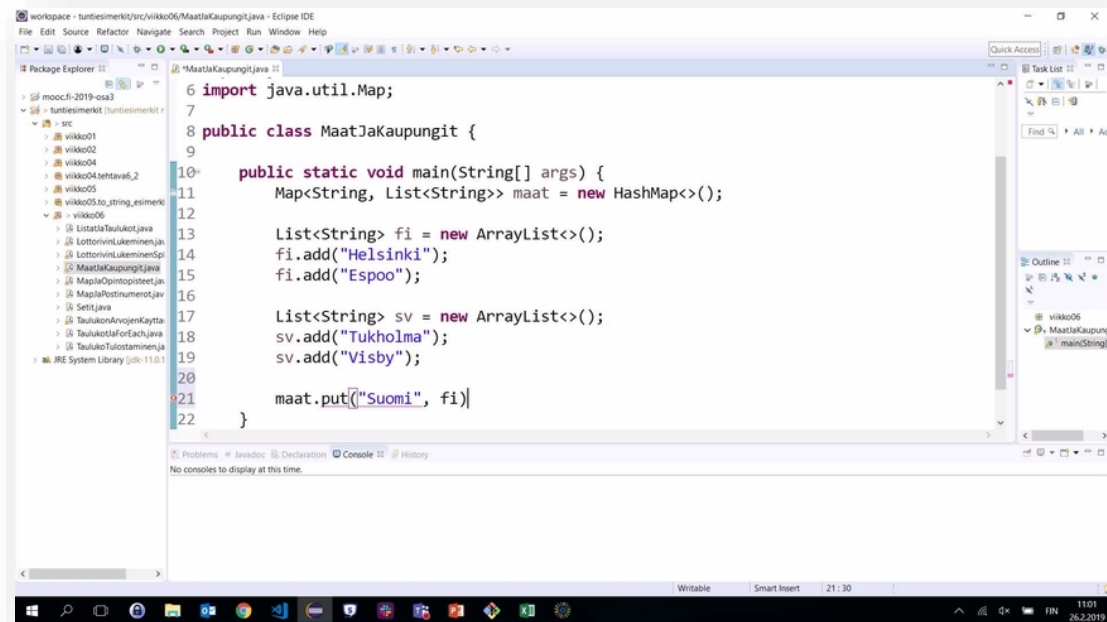
// Muuttuja matti saa arvon 10:
int matti = pistelaskuri.getOrDefault("Matti", 0);
// Muuttuja teppo saa arvon 0, avaimella ei arvoa:
int teppo = pistelaskuri.getOrDefault("Teppo", 0);

// Seuraava käsky kaataa ohjelman, koska int ei ole
// viittaustyyppinen muuttuja ja Kaija-avaimelle ei
// löydy arvoa:
int kaija = pistelaskuri.get("Kaija");
```

Palauttaa null-arvon, jota ei voida sijoittaa alkeistyyppiseen int-muuttujaan: ohjelma kaatuu

## Demo kokoelmista ja viittauksista:

[https://video.haaga-helia.fi/media/HashMap%2C+useiden+arvojen+tallentaminen+samalle+avaimelle/0\\_1xnwercr](https://video.haaga-helia.fi/media/HashMap%2C+useiden+arvojen+tallentaminen+samalle+avaimelle/0_1xnwercr)



```
6 import java.util.Map;
7
8 public class MaatJaKaupungit {
9
10     public static void main(String[] args) {
11         Map<String, List<String>> maat = new HashMap<>();
12
13         List<String> fi = new ArrayList<>();
14         fi.add("Helsinki");
15         fi.add("Espoo");
16
17         List<String> sv = new ArrayList<>();
18         sv.add("Tukholma");
19         sv.add("Visby");
20
21         maat.put("Suomi", fi);
22     }
```

The screenshot shows the Eclipse IDE interface. The Package Explorer on the left displays a project structure with packages 'moo.fi-2019-osa3' and 'tuntiesimerkit'. The main editor shows the Java file 'MaatJaKaupungit.java' with the code above. The Outline view on the right shows the class 'MaatJaKaupungit' and its 'main' method. The status bar at the bottom indicates 'Writeable', 'Smart Insert', and the time '21:30'.

## *Syventävää tietoa:* usean arvon tallentaminen samalle avaimelle

- Map:issa voidaan säilyttää vain yhtä arvoa kutakin avainta kohden.
- Säilytettävät arvot voivat kuitenkin olla kokoelmia.
  - Kokoelmat ovat olioita siinä missä muutkin oliot.
- Map:issa voidaan siis (tavallaan) säilyttää samalla avaimella useita arvoja, kun käsittelemme Mapin sisällä listoja tai muita kokoelmia.

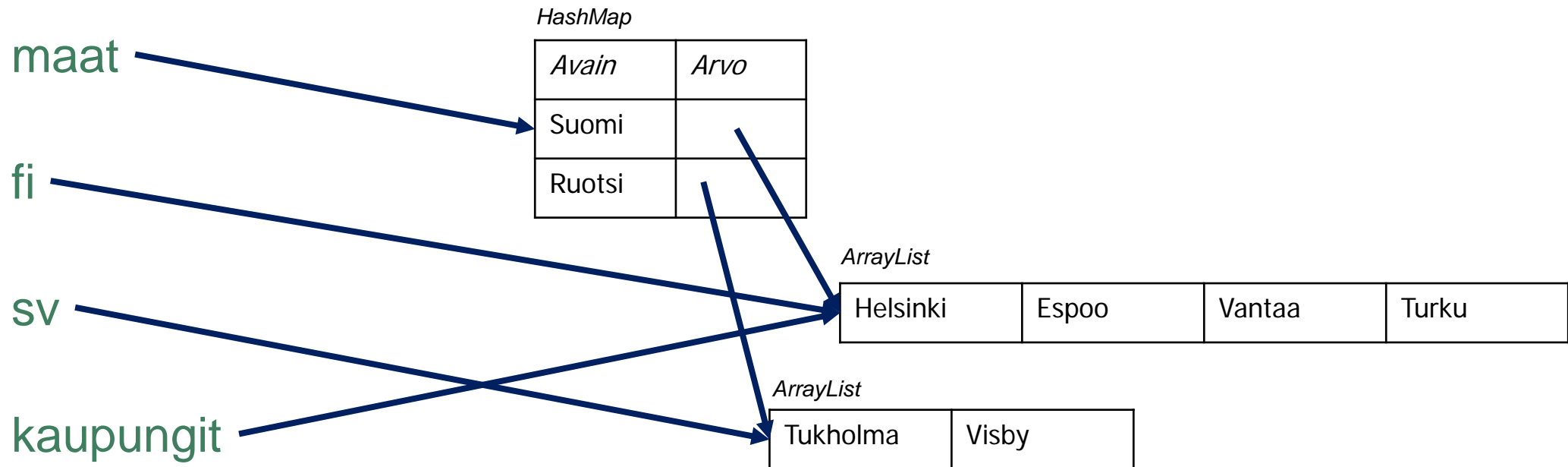
Oikealla oleva esimerkki tulostaa:

```
{Suomi=[Helsinki, Espoo, Vantaa],  
Ruotsi=[Tukholma, Visby]}
```

```
Map<String, List<String>> maat = new HashMap<>();  
  
List<String> fi = new ArrayList<String>();  
fi.add("Helsinki");  
fi.add("Espoo");  
fi.add("Vantaa");  
  
List<String> sv = new ArrayList<String>();  
sv.add("Tukholma");  
sv.add("Visby");  
  
maat.put("Suomi", fi);  
maat.put("Ruotsi", sv);  
  
System.out.println(maat);  
  
List<String> kaupungit = maat.get("Suomi");
```

## Muuttujat

## Oliot



Edellisen kalvon tietorakenteiden visualisointi: muistissa on vain kaksi listaa, joihin viitataan useilla muuttujilla ja map:in sisällä.

## *Syventävää tietoa:* **Map**:in koko sisällön läpikäynti

- Mapin sisältö voidaan käydä helposti läpi joko avainten, arvojen tai avain-arvo – parien osalta
  - `keySet()` palauttaa kaikki mapin avaimet
  - `values()` palauttaa kaikki mapin arvot
  - `entrySet()` palauttaa avaimet ja arvot pareina
    - Jokaisella `Entry`-oliolla on yksi avain ja arvo

```
Set<String> avaimet = data.keySet();
Collection<Integer> arvot = data.values();
Set<Entry<String, Integer>> parit = data.entrySet();

// Käydään läpi kaikki avaimet:
for (String avain : avaimet) {
    System.out.println(avain);
}

// Käydään läpi kaikki arvot:
for (Integer arvo : arvot) {
    System.out.println(arvo);
}

// Käydään läpi kaikki avain-arvo -parit:
for (Entry<String, Integer> pari : parit) {
    System.out.println("Avain: " + pari.getKey());
    System.out.println("Arvo: " + pari.getValue());
}
```