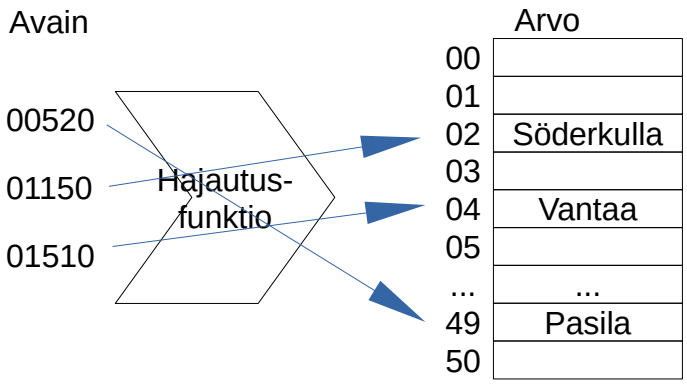
Hash map -tietorakenne

Hash map tietorakenne on toiselta nimeltään hajautustaulu (hash table). Kyseessä on siis tietorakenne aivan kuten ArrayList. Myös erilaiset puumaiset tietorakenteet ovat tietorakenteita, kuten vaikkapa binääripuu (binary tree). Hajautustaulu on siis yksi vaihtoehto tietorakenteeksi. Nyt ryhdymme tarkastelemaan millainen hash map eli hajautustaulu on ja milloin sitä kannattaa käyttää.

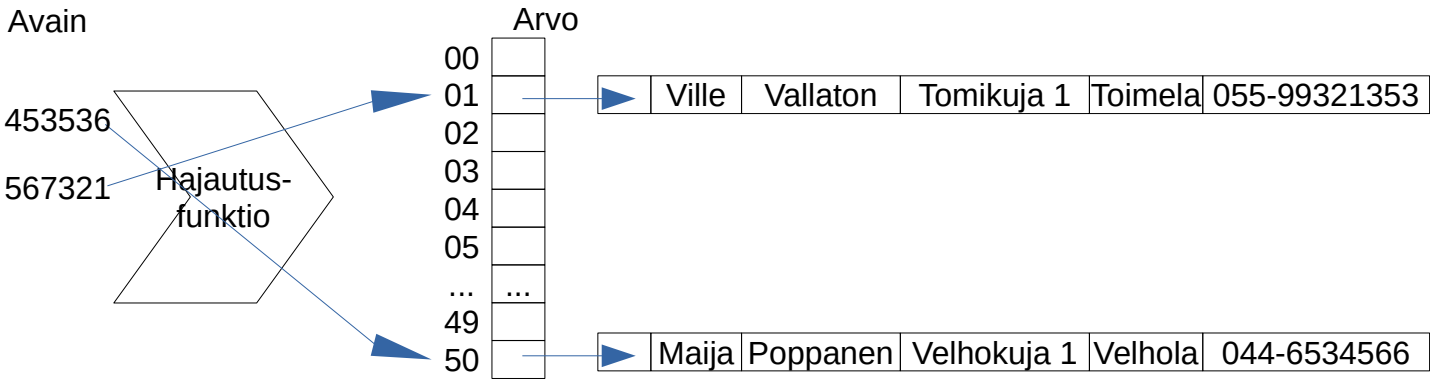
Hajautustaulua kannattaa käyttää aina kun dataa talletetaan avain ja sen data-arvo parina (key, value). Esimerkiksi vaikkapa postinumero ja sitä vastaava postitoimipaikka (00520, Pasila). Postinumero ja sitä vastaava postitoimipaikka ovat hyvä esimerkki avain ja sitä vastaavasta data-arvosta. Muitakin esimerkkejä on. Myös asiakasnumero ja sitä vastaavat asiakkaan tiedot ovat tällainen pari (453536, Maija Poppanen Velhokuja 1 Velhola puhelin 044-6534566).

Hajautustaulusta tekee merkittävän sen nopeus – se on yksi nopeimmista tietorakenteista. Se on nopeampi keskimäärin kuin hakupuut (search trees) tai taulukot. Siksi se onkin eniten käytetty tietorakenne erilaisissa sovelluksissa ArrayList:in ohella. Hajautustaulukon nopeus siis houkuttelee käyttämään sitä aina. On kuitenkin tärkeitä ”ongelmia” hajautustaulukossa. Niiden vuoksi hajautustaulukkoa kannattaa käyttää vain silloin kun vältytään kaikilta näiltä ongelmilta. Silloin kannattaa valita jokin muu tietorakenne. Esimerkiksi ArrayList on todella suosittu tietorakenne myös. Ja muitakin vaihtoehtoja löytyy (esim. puumaiset tietorakenteet).

Nyt voimme tarkastella, mitä ”ongelmia” hajautustaulukkoon liittyy. Siten voi välttyä niiltä, jos käyttää hajautustaulukkoa. Hajautustauluun liittyy sen hajautusfunktioon liittyvät niin sanotut ”törmäykset” (hash collisions). Tässä tarkoitetaan sitä, että hajautusfunktio generoi saman arvon useammalle avaimen arvolle. Eli tavallaan tapahtuu siis ”törmäys” kun toisen avaimen arvon viittamaan paikkaan tallennetaankin toisen avaimen arvon viittaamaa dataa. Hajautusfunktiossa tämä on tyypillisesti otettu huomioon.



Kuva 1. Postinumero ja postitoimipaikka parina.



Kuva 2. Asiakasnumero ja asiakastiedot parina.

Hajautus taulukon ”törmäysten” (hash collision) välttämisessä on tärkeää käyttää hyvin toimiviksi todettuja hajautusfunktioita. Hyvin toimivat hajautusfunktiot käyttävät tasaisemmin käytettävissä olevaa avain, arvo aluetta hajautuksen tekemiseen. Todennäköisyys ”törmäykselle” on siis vähäisempi. Törmäyksen tapahtuessa, tallennetaan ketjuttaen. Ketjuttaminen puoletaan tarkoittaa hitautta kun tietoa haetaan.

Map -luokka

Java kielessä hajautustaulu tietorakenteen saa käyttöön `java.util.Map` ja `java.utilHashMap` kirjastoilla. Ne on `import`attava tiedoston alussa:

```
import  java.util.Map;
import  java.util.HashMap;

public class OhjelmaSovellus1 {
      public static void main(String[] args) {
            Map<String, String> postitoimipaikat =
                  new HashMap<String, String>();
      }
}
```

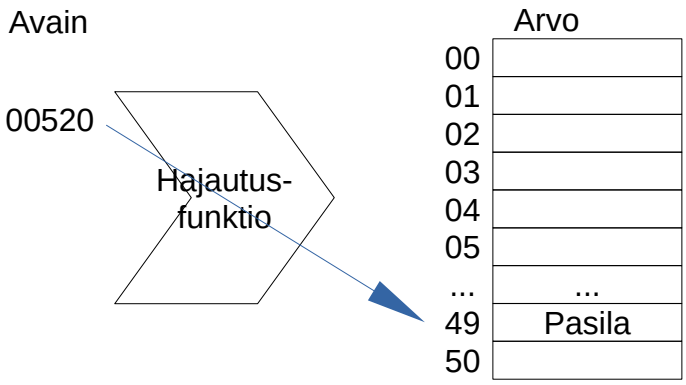
Nyt on saatu luotua postitoimipaikat niminen hajautustaulu tietorakenne. Se toisin on juuri nyt tyhjä. Voisimme lisätä sinne vaikkapa Pasilan postitoimipaikan ja käyttää avaimena sen postinumeroa (00520). Tämä onnistuu HashMap luokan put -metodilla:

```
import  java.util.Map;
import  java.util.HashMap;

public class OhjelmaSovellus1 {
      public static void main(String[] args) {
            Map<String, String> postitoimipaikat =
                  new HashMap<String, String>();

            postitoimipaikat.put(”00520”, ”Pasila”);
      }
}
```

Nyt meillä on postitoimipaikat tietorakeneessa ensimmäinen tietue:



Kuva 3. Postinumero ja postitoimipaikka parina.

Voisimme halutessamme lisätä uusia postinumero, postitoimipaikka pareja hajautustauluumme `put(avain, arvo)` -metodin avulla. Hajautustaulussa olevia tietoja puolestaan voimme

palauttaa (luea/hakea) `get(avain)` metodilla. Voisimme hakea esimerkiksi postinumeroa 00520 vastaavan data-arvon – siis postitoimipaikan:

```
import  java.util.Map;
import  java.util.HashMap;

public class OhjelmaSovellus1 {
      public static void main(String[] args) {
            Map<String, String> postitoimipaikat =
                  new HashMap<String, String>();

            postitoimipaikat.put(”00520”, ”Pasila”);
            System.out.println(postitoimipaikat.get(”00520”));
      }
}
```

Ohjelmamme tulostaisi nyt Eclipsen konsoli-ikkunaan: Pasila.

Myös olioita voisimme käyttää hajautustaulussa ”data-arvoina”. Voisimme luoda hajautustaulun asiakkaat, jossa avaimena olisi asiakasnumero ja sitä vastaavana ”data-arvona” asiakas olio. Asiakas luokka täytyisi tietysti myös tehdä, jotta siitä asiakas olioiden luonti onnistuisi:

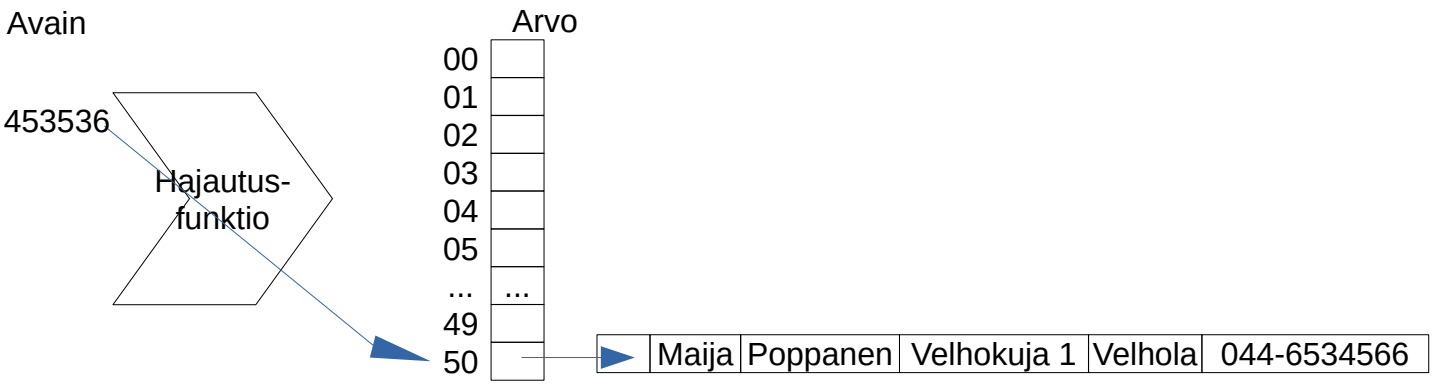
```
import  java.util.Map;
import  java.util.HashMap;

public class OhjelmaSovellus1 {
      public static void main(String[] args) {
            Map<Integer, Asiakas> asiakkaat =
                  new HashMap<Integer, Asiakas>();

            Asiakas asiakas1 = new Asiakas(”Maija”, ”Poppanen”, ”Velhokuja 1”,
                  ”Velhola”, ”044-6534566”);

            asiakkaat.put(453536, asiakas1);
      }
}
```

Nyt meillä on myös asiakkaat niminen hajautustaulukko tietorakenne. Siinä on yksi avain, data-arvo pari, Maija Poppasen asiakasnumero on avaimena ja sitä vastaavana ”data-arvona” on olio, josta Maija Poppasen asiakastiedot löytyvät. Tämä tietorakenne näyttäisi tällä hetkellä seuraavanlaiselta:



Kuva 4. Asiakasnumero ja asiakastiedot parina.

Yllä olevassa kuvassa on esimerkin mukaisesti esitetty, miten hajautusfunktio on ”laskenut” asiakasnumeroa `453536` vastaavan tallennuspaikan indeksi arvon (50). Todellisuudessa hajautusfunktion laskenta riippuu käytetyn hajautustaulun hajautusfunktiosta ja sen versiosta. Mutta toiminta idea on yllä esitetyn kuvan kaltainen.