Tietorakenteet, laskuharjoitus 8, 21-26.3

1. Javassa kaikki luokat perivät luokalta `Object` metodit `hashCode` ja `equals`. Metodeita käytetään jos olioa käytetää `HashMap`:n avaimena tai olio talletetaan `HashSet`:iin. `hashCode`:n tuloksen avulla etsitään alkion paikka hajautustaulukosta ja `equals`:in avulla katsotaan löytyykö paikasta alkavalta listalta etsitty alkio.

   Oletusarvoiset toteutukset `hashCode` ja `equals` metodeille toimivat odotetulla tavalla esim. `String`-olioille. Jos toteutetaan omia luokkia joita käytetään hajautusavaimena, on metodin kuitenkin yleensä ylikirjoitettava.

   Oletetaan, että on määritelty luokka jonka avulla voidaan kuvata kordinaatteja:

```
public class Kordinaatti {
    private int x;
    private int y;

    public Kordinaatti(int x, int y) {
        this.x = x;
        this.y = y;
    }

    public String toString() {
        return "("+x+", "+y+")";
    }
}
```

   Kordinaatteja talletaan `HashSet`:inä toteutettuun joukkoon:

```
        HashSet<Kordinaatti> kordinaatit = new HashSet<Kordinaatti>();

        kordinaatit.add( new Kordinaatti(1, 1) );
        kordinaatit.add( new Kordinaatti(5, 3) );

        System.out.println( kordinaatit.contains( new Kordinaatti(1, 1)) );
        System.out.println( kordinaatit.contains( new Kordinaatti(1, 2)) );
        System.out.println( kordinaatit.contains( new Kordinaatti(5, 3)) );
```

   Ohjelman tulostus on nyt `false false false` sillä oletusarvoinen toteutus `hashCode`:ille ja `equals`:ille pitää olioita samoina vain jos vertaillaa olioa itsensä kanssa, samansisältöisyys ei riitä.

   Kuitenkin haluaisimme esimerkissämme, että mitkä tahansa kaksi kordinaattiolioa, joilla on sama sisältö (eli x- ja y-kordinaatit) tunnistetaan samaksi. Tämä saadaan aikaan ylikirjoittamalla luokan `Kordinaatti` metodit `hashCode` ja `equals` sopivalla tavalla.

   Ylikirjoita metodit. Ohjeita seuraavassa:
   `https://wiki.helsinki.fi/display/javak11/Javan+Set+ja+Map`

   Varmista, että ohjelma tulostaa `true false true`.

   On helppo generoida huono, mutta toimiva hajautusfunktio. Yritä kuitenkin tehdä funktiostasi mahdollisimman hyvä.

2. HashMap:in avaimena käytettävä `Luokka` osoite kaipaa myös metodien `hashCode` ja `equals` ylikirjoittamista:

```
public class Osoite {
    private String katu;
    private int postinumero;
    private String kaupunki;

    public Osoite(String katu, int postinumero, String kaupunki) {
        this.katu = katu;
        this.postinumero = postinumero;
        this.kaupunki = kaupunki;
    }

    public static Osoite muodosta(String katu, int postinumero, String kaupunki){
        return new Osoite(katu, postinumero, kaupunki);
    }
}
```

Seuraavassa käyttöesimerkki. Osoitteisiin on esimerkissä liitetty kordinaatti.

```
HashMap<Osoite, Kordinaatti> sijainnit = new HashMap<Osoite, Kordinaatti>();
sijainnit.put( new Osoite("Mannerheimintie 1", 100, "Helsinki"),
               new Kordinaatti(100, 200));
sijainnit.put( new Osoite("Helsinginkatu 10", 500, "Helsinki"),
               new Kordinaatti(150, 195));

System.out.println( sijainnit.get( new Osoite("Mannerheimintie 1", 100, "Helsinki")) );
System.out.println( sijainnit.get( new Osoite("Helsinginkatu 10", 500, "Helsinki")) );
```

Toteuta siis metodit `hashCode` ja `equals` siten, että saman sisältöiset osoiteoliot tulkitaan samaksi. Pyri jälleen hyvään hajautusfunktioon.

Toimiessaan oikein ohjelma tulostaa `(100, 200) (150, 195)`

Bonus: muuta toteutustasi siten, että osoitteen kirjoitusasu ei ole case sensitive, eli myös seuraavat toimisivat:

```
System.out.println( sijainnit.get( new Osoite("MANNERHEIMINTIE 1", 100, "HELSINKI")) );
System.out.println( sijainnit.get( new Osoite("helsinginkatu 10", 500, "helsinki")) );
```

3. Toteuta hajautusrakenne ylivuotoketjuja tai avointa hajautusta käyttäen. Voit olettaa, että hajautustaulun avaimet ovat merkkijonoja. Tässä tehtävässä riittää toteuttaa operaatiot `search` ja `insert`. Riittää, että taulukkosi pysyy koko ajan samankokoisena, eli uudelleenhajautusta ei tarvita. Toteuta hajautusfunktio mahdollisimman järkevällä tavalla.

   Tee toteutuksestasi sellainen, että jokainen avain talletetaan rakenteeseen korkeintaan kertaalleen. Eli ennenkuin lisäät avaimen rakenteeseen, tarkista että se ei jo ole rakenteessa ennestään.

   Jos tee ylivuotoketjuja käyttävän hajautustaulun, voit toteuttaa ylivuotoketjut Javan valmiiden tietorakenteiden (LinkedList tai ArrayList) avulla. Huomaa kuitenkin, että jos haluat luoda taulukollisen esim. LinkedList:eja, ei seuraava toimi:

```
LinkedList<String>[] taulukko = new LinkedList<String>[3];
```

   Täytyy kirjoittaa:

```
LinkedList<String>[] taulukko = new LinkedList[3];
```

Joillakin Javan versioilla tämäkään ei tosin toimi. Googlella löytyy erilaisia ratkaisuja esim. hakusanoilla `java generic array`

Huom: ei ole välttämättä suositeltua käyttää geneerisiä tyyppejä ja taulukoita sekaisin, ks: `http://download.oracle.com/javase/tutorial/extra/generics/fineprint.html`

4. Testaa edellisen tehtävän hajautustauluasi lisäämällä sinne Aleksis Kiven romaanin Seitsemän veljestä kaikki sanat. Romaani on saatavilla tekstimuodossa seuraavassa osoitteessa:

   `http://www.gutenberg.org/files/11940/11940-8.txt`

   Voit kokeilla erilaisia hajautusfunktioita. Jos toteutuksesi perustuu ketjutukseen, selvitä mikä on pisin ylivuotoketjun pituus ja keskimääräinen ylivuotoketjun pituus.

   Jos toteutuksesi perustuu avoimeen hajautukseen, tarkastele suorituskykyä eri mittaisilla taulukoilla ja eri törmäyksenhallintastrategioilla.

   Tiedoston lukeminen ja sen käsittely sana sanalta onnistuu Javalla seuraavasti:

```
import java.io.File;
import java.util.Scanner;

public class Main {
    public static void main(String[] args) throws Exception {
         // avataan tiedosto seitseman.txt
         Scanner teksti = new Scanner(new File("seitseman.txt"));
         // tai      = new Scanner(new File("seitseman.txt"), "ISO-8859-1");
         // jos tiedosto on ISO-8859-1-formaatissa

         // jatketaan niin kauan kuin sanoja jäljellä
         while ( teksti.hasNext() ) {
            String mj = teksti.next();  // haetaan seuraava sana
            System.out.println(mj);
         }
    }
}
```

5. Vertaa oman hajautusrakenteesi, Javan hajautustaulun HashSet:in ja Javan tasapainoitetun hakupuun TreeSet:in suorituskykyä. Jos et tehnyt edellistä tehtävää, saat tästä rastin vertaamalla HashSet:iä ja TreeSet:iä. Ohje suorituskykymittauksille löytyy laskareiden 3 tehtäväpaperissa.

   Tee seuraavat mittaukset:

   - Lisää rakenteiseen Seitsemän veljeksen sanat ja mittaa tähän kuluva aika
   - Edellisen jälkeen testaa jokaisesta Dostojevskin romaanin Rikos ja rangaistus englanninkielisen käännöksen sanasta löytyykö se Seitsemän veljeksen sanojen joukosta ja mittaa tähän kuluva aika. Käy läpi kaikki Rikoksen ja rangaistuksen sanat alusta loppuun, sama sana saa siis toistua kyselyssä niin monta kertaa kuin se tulee tekstissä vastaan.

   Rikos ja rangaistus löytyy osoitteesta:

   `http://www.gutenberg.org/files/2554/2554-8.txt`