Ohjelmoinnin jatkokurssi, kurssikoe 4.5.2012

Vastaa tehtävät 1, 2 ja 3 **erillisille** konsepteille. Kirjoita jokaiseen konseptiin kurssin nimi, kokeen päivämäärä, oma nimi, nimikirjoitus ja opiskelijanumero. Vastaukset palautetaan tehtäväkohtaisiin pinoihin.

Vaikka jättäisit johonkin tehtävään vastaamatta, tulee vastauspaperi siinäkin tapauksessa palauttaa.

Huom: kokeen viimeisellä sivulla on pääohjelmarunko sekä lista HashMap:in ja ArrayList:in yleisimmistä komennoista.

1. Käsitteistöä (6p)

   Vastaa jokaiseen kohtaan lyhyesti ja ytimekkäästi. Koko tehtävän vastauksen pituudeksi riittää noin sivu.

   (a) Kerro keskeisimmistä alkeis- ja viittaustyyppisten muuttujien eroista.

   (b) Mitä tarkoitetaan rajapinnalla? Miten rajapintoja voi hyödyntää ohjelmoinnissa?

   (c) Mikä on abstrakti luokka?

   (d) Mitä tarkoitetaan polymorfismilla?

   (e) Mikä on tapahtumankäsittelijä?

   (f) Mikä on JButton?

2. Tiedostot ja kokoelmat (12 p)

   Tiedoston `sanat.txt` jokaisella rivillä on yksi suomen kielen sana. Tee ohjelma, joka lukee tiedoston ja selvittää yleisimmän sanan. Käytä ohjelmassasi HashMap- tai ArrayList-rakennetta sopivalla tavalla.

   Voit olettaa, että yleisin sana on yksikäsitteinen.

   Esimerkkitiedosto:

   ```
   apina
   cembalo
   apina
   banaani
   cembalo
   cembalo
   banaani
   ```

   Kun ohjelma lukee yllä olevan esimerkkitiedoston, tulee tulostuksen olla seuraava:

   ```
   Yleisin sana: cembalo (3 kpl)
   ```

3. Olio-ohjelmointi (12 p)

Tässä tehtävässä mallinnetaan putkistoa, jossa virtaa vettä. Virtaavan veden määrä, eli *virtaus*, esitetään double-arvona, joka on aluksi 0.0. Putkisto koostuu *osista*, joiden rajapinta mahdollistaa virtauksen lisäämisen:

```
public interface PutkistonOsa {
    void lisaaVirtausta(double virtaus);
}
```

Ohjelmoi seuraavanlaiset putkiston osat (kaikki luokista siis *toteuttavat* rajapinnan PutkistonOsa). Huomaa, että useimpien luokkien ei tarvitse tallentaa virtaustaan mihinkään.

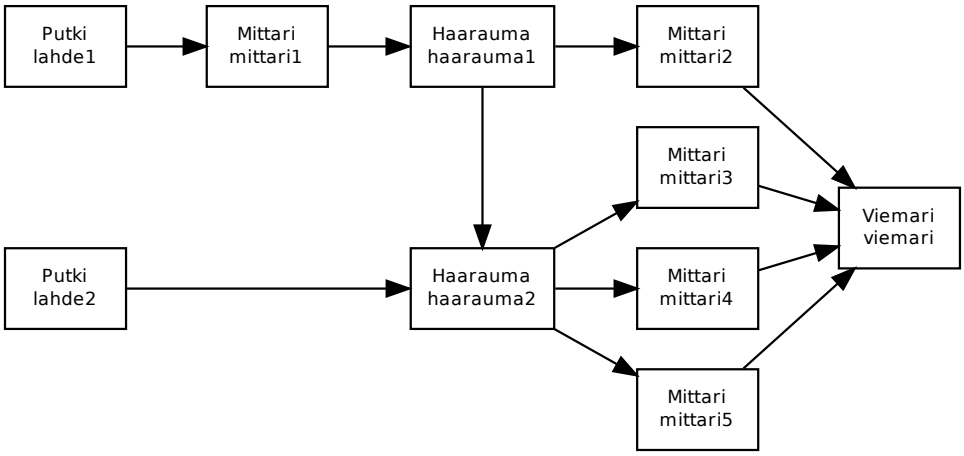
(a) (1 p) Luokka `Viemari`, joka ei tee virtauksellaan mitään.

(b) (3 p) Luokka `Putki`, joka saa konstruktorin parametrina seuraavan putkiston osan. Kun putkeen lisätään virtausta, se lisää saman verran virtausta seuraavaan putkiston osaan.

(c) (4 p) Luokka `Mittari`, joka *perii* luokan `Putki`. Mittari toimii kuin putki, mutta lisäksi se pitää kirjaa virtauksestaan. Tee virtauksen lukemista varten metodi `public double haeVirtaus()`.

(d) (4 p) Luokka `Haarauma`, joka saa konstuktorin parametrina listan putkiston osia. Haaraumaan lisätyn virtauksen tulee jakaantua tasaisesti annettuihin haaroihin. Jos haaroja on esimerkiksi kolme ja haaraumalle lisätty virtaus on 60, niin kuhunkin haaraan tulee lisätä virtausta 20.

Luokista tehdyistä olioista voidaan rakentaa esimerkiksi seuraavanlainen putkisto.



Seuraavalla sivulla on putkiston rakentava esimerkkiohjelma.

```
public class Main {
    public static void main(String[] args) {
        Viemari viemari = new Viemari();

        Mittari mittari2 = new Mittari(viemari);
        Mittari mittari3 = new Mittari(viemari);
        Mittari mittari4 = new Mittari(viemari);
        Mittari mittari5 = new Mittari(viemari);

        List<PutkistonOsa> haarat2 = new ArrayList<PutkistonOsa>();
        haarat2.add(mittari3);
        haarat2.add(mittari4);
        haarat2.add(mittari5);

        Haarauma haarauma2 = new Haarauma(haarat2);

        List<PutkistonOsa> haarat1 = new ArrayList<PutkistonOsa>();
        haarat1.add(mittari2);
        haarat1.add(haarauma2);

        Haarauma haarauma1 = new Haarauma(haarat1);

        Mittari mittari1 = new Mittari(haarauma1);

        Putki lahde1 = new Putki(mittari1);
        Putki lahde2 = new Putki(haarauma2);

        lahde1.lisaaVirtausta(123.45);
        lahde2.lisaaVirtausta(3000.0);

        System.out.println("Mittari 1: " + mittari1.haeVirtaus());
        System.out.println("Mittari 2: " + mittari2.haeVirtaus());
        System.out.println("Mittari 3: " + mittari3.haeVirtaus());
        System.out.println("Mittari 4: " + mittari4.haeVirtaus());
        System.out.println("Mittari 5: " + mittari5.haeVirtaus());
    }
}
```

Tulostus:

```
Mittari 1: 123.45
Mittari 2: 61.725
Mittari 3: 1020.575
Mittari 4: 1020.575
Mittari 5: 1020.575
```

Pääohjelmarunko

```
import java.util.Scanner;

public class KoeOhjelma {

    public static void main(String[] args) {
        Scanner lukija = new Scanner(System.in);

        int luku = Integer.parseInt( lukija.nextLine() );
        String merkkijono = lukija.nextLine();
    }
}
```

java.util.HashMap

- public HashMap<K,V>() luo tyhjän HashMap-olion, jossa `K`-tyyppiset oliot ovat avaimina ja niillä on `V`-tyyppisiä olioita arvoina.
- public V put(K key, V value) tallentaa HashMap-olioon avain-arvo-parin. Palautuva viite `V`-tyyppiseen olioon on viite vanhaan olioon, joka korvattiin tai `null`.
- public V get(Object key) palauttaa viitteen `V`-tyyppiseen olioon, joka liittyy `key`-avaimeen. Jos avaimella `key` ei löydy arvoa, palautetaan arvo `null`.
- public boolean containsKey(Object key) kertoo löytyykö HashMap:ista tiettyä avainta
- public V remove(Object key) poistaa avaimen `key` ja siihen liittyneen arvon. Palauttaa viitteen `V`-tyyppiseen olioon, joka on poistettu arvo tai `null`.
- public Set<K> keySet() palauttaa kaikkien HashMapissa olevien avaimien joukon.
- public Collection<V> values() palauttaa kaikkien HashMapissa olevien arvojen joukon.

java.util.ArrayList

- public ArrayList<T>() luo uuden ArrayList-olion jossa listan elementit ovat tyyppiä `T`.
- public boolean add(T x) lisää listan loppuun olion `x`.
- public boolean contains(Object o) tarkistaa onko listassa oliota `o`.
- T get(int i) palauttaa listan alkion indeksistä `i`.