Ohjelmoinnin jatkokurssi, kurssikoe 29.4.2013

Kirjoita jokaiseen palauttamaasi konseptiin kurssin nimi, kokeen päivämäärä, oma nimi, nimikirjoitus ja opiskelijanumero. Vastaa tehtävät 1 ja 4 omille konsepteilleen. Tehtävät 2 ja 3 voivat olla samalla konseptilla.

Huom: kokeen viimeisellä sivulla on pääohjelmarunko sekä lista HashMap:in ja ArrayList:in yleisimmistä komennoista. Kokeessa nähtävä poikkeus *IllegalStateException* toimii kuten kurssilla nähty *IllegalArgumentException*.

1. Essee aiheesta rajapinnat (3p)

   Kerro mitä rajapinnat ovat, miten niitä käytetään ja mitä hyötyjä niiden käytöstä on. Anna konkreettisia koodiesimerkkejä. Vastauksen pituus ei saa ylittää kahta sivua.

2. Sanakirja (9p)

   Ensin ohjelma kyselee käyttäjältä sanoja ja niiden käännöksiä (käyttäjän syötteet *italicsilla*):

   ```
   Syötä sanoja ja niiden käännöksiä. Tyhjä syöte lopettaa:
   sana: olut
   käännös: beer
   sana: pallo
   käännös: ball
   sana: jalka
   käännös: foot
   sana: jalka
   käännös: leg
   sana:
   ```

   Kun käyttäjä antaa tyhjän syötteen, siirtyy ohjelma toiseen vaiheeseen, eli varsinaiseen "sanakirjamoodiin":

   ```
   Kysele sanakirjalta käännöksiä. Tyhjä syöte lopettaa:
   sana: käsi
   käännös ei tiedossa!
   sana: pallo
   ball
   sana: jalka
   foot, leg
   sana:
   Näkemiin...
   ```

   Tee siis ohjelma, joka toimii esimerkkitulosteiden kuvaamalla tavalla. Huomioi jo nyt seuraavassa osassa tehtävään tehtävä laajennus ja tee ohjelmasi rakenteesta sen verran siisti, että laajennus onnistuu.

   Huomaa, että sanalla (kuten esimerkissä sanalla "jalka") voi olla useampia käännöksiä! Jos et tiedä miten saat sanakirjan toteutettua siten, että yhdellä sanalla on useita käännöksiä, voit tehdä ohjelman siten että yhdellä sanalla voi olla ainoastaan yksi käännös. Tässä tapauksessa voit saada tehtävästä maksimissaan 7 pistettä.

3. (3p) Sanakirjan laajennus

   Laajenna sanakirjaasi siten, että ennen ensimmäistä vaihetta sanakirjaan voidaan lukea sanoja tiedostosta. Sanat ovat tiedostossa joka toisella rivillä, ensin sana ja sen jälkeen käännös. Seuraavassa 3 käännöstä sisältävä tiedosto *sanat.txt*:

```
olut
beer
vesi
water
kenkä
shoe
```

Ohjelman laajennetun version tulisi toimia seuraavasti:

```
Anna luettavan sanakirjatiedoston nimi: sanat.txt
luettiin tiedostosta 3 käännöstä:
olut -> beer
vesi -> water
kenkä -> shoe
Syötä sanoja ja niiden käännöksiä. Tyhjä syöte lopettaa:
sana:
Kysele sanakirjalta käännöksiä. Tyhjä syöte lopettaa:
sana: olut
käännös: beer
sana:
Näkemiin...
```

4. (14p) Sensoreita

   Käytössämme on seuraava rajapinta:

```
public interface Sensori {
    boolean onPaalla();  // palauttaa true jos sensori on päällä
    void paalle();       // käynnistää sensorin
    void poisPaalta();   // sulkee sensorin
    int mittaa();        // palauttaa sensorin lukeman jos sensori on päällä
                         // jos sensori ei päällä heittää poikkeuksen IllegalStateException
}
```

   (a) Tee luokka `Lampomittari` joka toteuttaa rajapinnan `Sensori`

   Aluksi lämpömittari on poissa päältä. Kutsuttaessa metodia `mittaa`, kun mittari on päällä, mittari arpoo luvun väliltä *min*...*max* ja palauttaa sen kutsujalle. Jos mittari ei ole päällä, heitetään poikkeus `IllegalStateException`. Arvot *min* ja *max* lämpömittari saa konstruktorin parametreina. Esim. jos luodaan lämpömittari kutsulla `new Lampomittari(-20, 35)`, ovat mitatut lämpötilat väliltä -20...35.

   Arvonta onnistuu `Random`-luokan avulla seuraavaan tyyliin:

```
Random arpa = new Random();
int arvottuLuku = arpa.nextInt(10);
// muuttujan arvottuLuku arvo nyt 0...9
```

   (b) Tee luokka `MuistavaLampomittari`, joka *perii* luokan lämpömittari ja laajentaa sen toiminnallisuutta metodilla `public List<Integer> kaikkiMittaukset()`, joka palauttaa listan kaikista mittarilla tehdyistä mittauksista. Luokan konstruktori ottaa parametreina mittarin pienimmön ja suurimman mahdollisen lämpötilan.

   (c) Tee luokka `MaksimiSensori` joka toteuttaa rajapinnan `Sensori`

   MaksimiSensori sisältää useita sensoreita. Rajapinnan `Sensori` lisäksi MaksimiSensorilla on metodi `public void lisaaSensori(Sensori lisattava)` jonka avulla MaksimiSensorin hallintaan lisätään uusi sensori.

MaksimiSensori on päällä silloin kuin *kaikki* sen sisältävät sensorit ovat päällä. Kun MaksimiSensori käynnistetään, täytyy kaikkien sen sisältävien sensorien käynnistyä jos ne eivät ole käynnissä. Kun MaksimiSensori suljetaan, täytyy ainakin yhden sen sisältävän sensorin mennä pois päältä. Saa myös käydä niin että kaikki sen sisältävät sensorit menevät pois päältä.

MaksimiSensorin metodi `mittaa` palauttaa sen sisältämien sensoreiden lukemien *maksimin*. Jos MaksimiSensorin metodia `mittaa` kutsutaan sensorin ollessa poissa päältä, tai jos MaksimiSensorille ei vielä ole lisätty yhtään sensoria heitetään poikkeus `IllegalStateException`.

Seuraavassa sensoreja käyttävä esimerkkiohjelma (huomaa, että luokkien konstruktorit ovat parametrittomia):

```
public static void main(String[] args) {
    MuistavaLampomittari kumpula = new MuistavaLampomittari(-30, 30);
    kumpula.paalle();
    System.out.println("lämpötila Kumpulassa "+kumpula.mittaa() + " astetta");

    Sensori kaisaniemi = new Lampomittari(-30, 30);
    Sensori helsinkiVantaa = new Lampomittari(-35, 30);

    MaksimiSensori paakaupunki = new MaksimiSensori();
    paakaupunki.lisaaSensori(kumpula);
    paakaupunki.lisaaSensori(kaisaniemi);
    paakaupunki.lisaaSensori(helsinkiVantaa);

    paakaupunki.paalle();
    System.out.println("maksimilämpötila Pääkaupunkiseudulla "+paakaupunki.mittaa() + " astetta");

    System.out.println("lämpötila Kumpulassa "+kumpula.mittaa() + " astetta");

    System.out.println("Kumpulassa mitatut lämpötilat:")
    for ( int lukema : kumpula.kaikkiMittaukset() ) {
       System.out.prinltn( "  " + lukema);
    }
}
```

tulostuu (tulostetut lukuarvot riippuvat tietenkin arvotuista lämpötiloista, oletuksena että pääkaupunkiseudulle tehty mittaus palautti Kumpulan osalta lukeman -11):

```
lämpötila Kumpulassa -7 astetta
maksimilämpötila Pääkaupunkiseudulla -5 astetta
lämpötila Kumpulassa -9 astetta
Kumpulassa mitatut lämpötilat:
  -7
  -11
  -9
```

Muistutus pääohjelmarungosta ja `Scanner`-luokan käytöstä, sekä HashMap- ja ArrayList-kokoelmaluokista

```
import java.util.Scanner;

public class Ohjelma {

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

       // muistutus Scannerin käytöstä:
       // int luku = Integer.parseInt( lukija.nextLine() );
       // String rivi = 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`.