Ohjelmistojen mallintaminen, syksy 2011, laskuharjoitus 1

Tämän viikon tehtävistä 1 ja 2 tehdään etukäteen kotona. Loput tehtävistä tehdään paikanpäällä.

1. Piirrä erilliselle A4-lapulle kuva tai tarvittaessa useampia kuvia Ohjelmoinnin perusteiden neljännen viikon laskuharjoituksissa esiintyneen minuuttilaskurista. Käytä kuvan pohjana ratkaisuehdotelmaa, jossa minuuttilaskuri kapseloi tuntilaskurin:

   `http://tinyurl.com/ohpe-lh4-hahmotelmia`

   Huom! Tässä ja seuraavassa tehtävässä sinun ei tarvitse (eikä oikeastaan pidäkään) vielä tietää mitään UMLstä tai mistään muustakaan kuvaustekniikasta. Piirrä vain kuva, joka mielestäsi kuvastaa minuuttilaskuria. Piirrokset palautetaan laskuharjoitustilaisuudessa, kirjoita piirrokseen myös opiskelijanumerosi!

2. Viimeiseltä sivulta löytyy pätkä Java-koodia. Piirrä kuva tai joukko kuvia jotka havainnollistavat ohjelman toimintaa.

   Myös tämä tehtävä palautetaan laskareissa nimellä ja opiskelijanumerolla varustettuna.

   Tämän ja edellisen tehtävän kuvia ei arvostella millään tavalla, haluamme ainoastaan tietää miten opiskelijat hahmottavat ohjelmakoodin visuaalisesti.

Laskuharjoituksissa paikanpäällä tehtävät tehtävät:

3. Lue wikissä oleva JUnit-ohje:
   `https://wiki.helsinki.fi/display/ohma/JUnit`

   ja **tee samalla testit itsekin**, jatkamme Varaston työstämistä vielä tehtävässä 6.

4. Tee Ohjelmoinnin perusteiden viikon 4 tehtävälle Tuntilaskuri seuraavat JUnit-testit:

   - luodun laskurin arvo on 0
   - kun etene-metodia on kutsuttu kerran, laskurin arvo on 1
   - kun etene-metodia on kutsuttu 2 kertaa, laskurin arvo on 2
   - kun etene-metodia on kutsuttu 24 kertaa, laskurin arvo on 0
   - kun etene-metodia on kutsuttu 25 kertaa, laskurin arvo on 1

   Tee jokaisesta testistä oma testimetodi. Muista nimetä testit kuvaavasti.

Jos olet kadottanut oman tuntilaskurisi, löydät koodin ohpen mallivastauksista:
`http://tinyurl.com/ohpe-lh4-hahmotelmia`

HUOM: testiluokan nimen on päätyttävä sanaan Test, eli tee testistäsi nimeltään esim TuntilaskuriTest

5. Tee Ohjelmoinnin perusteiden viikon 4 tehtävälle Minuuttilaskuri JUnit-testit. Testien tulee testata ainakin seuraavia asioita: minuuttien ja tuntien eteneminen sekä minuuttien ja tuntien pyörähtäminen takaisin nollaan ja eteneminen taas nollautumisen jälkeen.

6. Täydennä varaston (wiki-dokumentissa valmiina olevat) testit huomioimaan tapaukset, joissa varastoon yritetään laittaa liikaa tavaraa ja varastosta yritetään ottaa enemmän kuin siellä on

7. Lue wikissä oleva debuggeriohje:
   `https://wiki.helsinki.fi/display/ohma/debuggeri`

   ja **tee samalla ohjeen debuggausesimerkit itse**

8. Ohjelmoinnin perusteiden viikolla 6 käsiteltiin binäärihakua. Wikistä ja linkistä
   `https://wiki.helsinki.fi/pages/viewpage.action?pageId=76262468`
   löytyy hieman rikkinäinen binäärihaun toteutus.

   Etsi ensin arvo jolla binäärihaku ei toimi. Etsi sitten vika debuggeria hyväksikäyttäen ja korjaa vika.

   Korjauksen jälkeen käy debuggerilla ohjelman suoritus läpi muutamalla eri arvolla.

9. Demonstroi debuggerin avulla minuuttilaskurin (ks. tehtävä 5) minuutin ja tunnin pyörähdys nollaan. Tee sopiva pääohjelma, ja aseta useampia breakpointeja wikin debuggeriohjeen kohdan "useita breakpointeja ja suorituksen jatkaminen" tyyliin.

10. Kokeillaan testien debuggaamista wikin debuggeriohjeen mukaan. Askella debuggerilla läpi tehtävään 6 tekemiesi Varaston tesien suoritus. Tästä taidosta saattaa olla hyötyä automaattisesti palautettavien pajatehtävien yhteydessä.

```
public class Kioski {
    public Matkakortti ostaMatkakortti(String nimi) {
        Matkakortti uusiKortti = new Matkakortti(nimi);
        return uusiKortti;
    }

    public Matkakortti ostaMatkakortti(String nimi, int arvoAlussa) {
        Matkakortti uusiKortti = new Matkakortti(nimi);
        uusiKortti.kasvataArvoa(arvoAlussa);
        return uusiKortti;
    }
}
```

```
public class Matkakortti {
    private String omistaja;
    private double arvo;
    private int pvm;
    private int kk;

    public Matkakortti(String n){
        omistaja = n;  pvm = 0;  kk = 0;  arvo = 0;
    }

    public void kasvataArvoa(double kasvatus){ arvo += kasvatus; }

    public void vahennaArvoa(double vahennys){ arvo -= vahennys; }

    public double getArvo(){ return arvo; }

    public void uusiAika(int p, int k){
        kk = k;
        pvm = p;
    }
}
```

```
public class Lataajalaite {
    public void lataaArvoa(Matkakortti k, double ladattavaArvo) {
        k.kasvataArvoa(ladattavaArvo);
    }

    public void lataaAikaa(Matkakortti k, int pvm, int kk) {
        k.uusiAika(pvm, kk);
    }
}
```

```
public class Lukijalaite {
    private double RATIKKA = 1.5;
    private double HKL = 2.1;
    private double SEUTU = 3.5;

    public boolean ostaLippu(Matkakortti k, int tyyppi){
        double hinta = 0;
        if ( tyyppi == 0 ) hinta = RATIKKA;
        else if ( tyyppi ==1 ) hinta = HKL;
        else hinta = SEUTU;

        if ( k.getArvo()<hinta ) return false;
        k.vahennaArvoa(hinta);

        return true;
    }
}

public class Main {
    public static void main(String[] args) {
        Lataajalaite rautatietori = new Lataajalaite();
        Lukijalaite ratikka6 = new Lukijalaite();
        Lukijalaite bussi244 = new Lukijalaite();

        Kioski lippuLuukku = new Kioski();
        Matkakortti artonKortti = lippuLuukku.ostaMatkakortti("Arto");

        rautatietori.lataaArvoa(artonKortti, 3);

        ratikka6.ostaLippu(artonKortti, 0);
        bussi244.ostaLippu(artonKortti, 2);
    }
}
```