Ohjelmistojen mallintaminen, syksy 2011, laskuharjoitus 2

Viikon 2 laskareita ei pidetä mikrosaleissa, käytössä ovat opetusohjelmaan merkatut salit.

Tämän viikon tehtävistä 1-6 tehdään etukäteen kotona. Loput tehtävistä tehdään paikanpäällä.

Tehtävät 1 ja 2 käsittelevät viikon 1 kalvojen 34-67 asiaa. Tehtäviät 3-6 käsittelevät viikon 2 asiaa, erityisesti kannattaa kerrata kalvot 13-33 ja 44-48.

1. Ohjelmoinnin, Tietorakenteiden ja Tietokantojen harjoitustyöissä opiskelijan on pidettävä kirjaa käyttämistään työtunneista. Aina kun opiskelija tekee tehtäviä harjoitustyöhön liittyen, on hänen kirjatattava ylös käyttämänsä työaika ja tieto millaiseen tehtävään aika käytettiin. Työtehtävät jakautuvat seuraaviin kuuteen luokkaan: määrittely, suunnittelu, ohjelmointi, testaus, dokumentointi ja muuta.

   Tehtävänä on suunnitella harjoitustöiden tuntikirjanpitoa tukeva tietojärjestelmä.

   Harjoitustyöt tehdään ryhmässä, johon kuuluu useita opiskelijoita ja yksi ohjaaja. Kaikki ryhmään kuuluvat opiskelijat tekevät saman aineen (Ohjelmointi, Tietorakenteet tai Tietokannat) harjoitustyötä. Kukin opiskelija tekee työnsä itsenäisesti. Ryhmän ohjaaja luo jokaiselle ryhmäläiselle käyttäjätunnuksen järjestelmään. Käyttäjätunnuksen luonnin yhteydessä opiskelijan tiedot (mm. osoite, aloitusvuosi, pääaine) kopioidaan OODI-järjestelmästä. Harjoitustyön aikana ohjaaja seuraa ryhmäläisten etenemistä järjestelmän avulla. Opintoesimiehen vastuulla on luoda kaikille ohjaajille sopivat käyttäjätunnukset järjestelmään sekä listätä järjestelmään tiedot jokaisesta ryhmästä. Ryhmän tiedot (ohjaaja, tapaamisajat, alku- ja loppupäivämäärä) kopioidaan laitoksen kurssikirjanpitojärjestelmästä.

   Tunnista tietojärjestelmän käyttäjät ja mieti minkälaisia käyttötapauksia järjestelmällä tulisi olla. Käyttötapauksesta riittää mainita nimi ja käyttötapauksen kulku lyhyesti (kuten luennon 1 kalvolla 42). Tee myös käyttötapausdiagrammi.

2. Kuvaa edellisen tehtävän käyttötapauksista kolme tarkemmalla tasolla.

   Käytä kuvaamiseen Alistair Cockburnin käyttötapauspohjaa, joka löytyy osoitteesta www.cs.helsinki.fi/u/mluukkai/ohmas10/usecase.pdf, tai samaa tyyliä kun viikon 1 luentokalvon sivulla 44 olevan käyttötapauksen *Opiskelija ilmoittautuu kurssille* tarkemman version yhteydessä on käytetty.

3. Mallinna Ohjelmoinnin perusteiden viikon 4.3 tehtävä kaksipyttyinen Viljavarasto Java-koodi luokkakaaviona

- Viljavarastotehtävä löytyy osoitteesta:
  http://www.cs.helsinki.fi/u/wikla/ohjelmointi/perus/s2011/harjoitukset/4/
- Java-koodi (esimerkkiratkaisu):
  https://wiki.helsinki.fi/display/s2011paja/Ratkaisuehdotelmia

Tee Viljavaraston luokkakaavio tarkasti (merkitse kaikki metodit ja attribuutit). Jatkossa riittää että luokkakaavioon merkataan kulloisenkin käyttötilanteen suhteen tärkeimmät metodit/attribuutit. Yleensä ei merkata kuin korkeintaan julkiset metodit ja attribuutit, ei aina niitäkään.

4. Mallinna Ohjelmoinnin jatkokurssin viikon 1 tehtävän 3 Lastiruuma Java-koodi luokkakaaviona

   - Tehtävä löytyy osoitteesta
     http://www.cs.helsinki.fi/u/wikla/ohjelmointi/jatko/s2011/harjoitukset/1/
   - Java-koodi (esimerkkiratkaisu, ilmestyy vasta ma 7.10 klo 23.59):
     https://wiki.helsinki.fi/display/s2011paja/ohja+viikko+1

5. Viimeiseltä sivulta löytyy viime viikon tehtävän 2 hieman laajennettu versio. Piirrä koodia vastaava luokkakaavio.

6. Piirrä edellisen tehtävän luokkakaavion mukainen oliokaavio, joka kuvaa main():in viimeisellä rivillä olevan tilanteen.

Laskuharjoituksissa paikanpäällä tehtävät tehtävät:

jakaantukaa 3-5 hengen ryhmiin

7. Oluen verkkomyynti sallitaan Suomessa vuodesta 2013 alkaen. Kumpula Biershopin toimitusjohtaja Arto Vihavainen haluaa avata verkkokaupan heti vuoden alussa ja on päättänyt pyytää meitä toimittamaan sovelluksen. Arton visio sovelluksesta on seuraava:

   Haluamme markkinoille niin pian kuin mahdollista. Sen takia onkin tärkeää että ensin laitetaan kaupan perustoiminnallisuus kuntoon, lisätoimintoja ehdimme viritellä sivuille myöhemminkin.

   Pääsivulle sijoitetaan firman logo sekä linkit yhteystietoihin ja itse kauppaan. Kaupassa näytetään lista ostettavissa olevista tuotteista. Jokaisesta tuotteesta näytetään nimi ja hinta. Klikkaamalla tuotetta voisi päästä tuotteen tarkempaan kuvaukseen, kuvauksena on linkki Ratebeer.com-sivulle. Toiminnallisuuden on tarkoitus olla normaalin webbikaupan kaltainen, eli klikkaamalla tuotteen yhteydessä olevaa nappia, menevät tuotteet asiakkaan ostoskoriin. Olisi hyvä jos ostoskorin saldo ja siellä olevien tuotteiden määrä on koko ajan näkyvissä. Varmaan myös nappi jolla tyhjennetään ostoskori on tarpeen.

Kun ostokset on kerätty koriin, pääsee asiakas kassalle nappia "maksa ostokset" klikkaamalla. Kassalla ollessa ostoskorin sisältö näytetään. Kassalla tuotteita on mahdollista poistaa ostoskorista, koko kori on mahdollista tyhjentää tai voidaan myös palata takaisin tekemään lisää ostoksia. Kassalla ollessa asiakas täyttää sivulle osoitteensa ja luottokorttinumeron. Kun asiakas klikkaa "suorita maksu", veloitetaan asiakkaan korttia ja ilmoitetaan jos veloitus onnistui. Asiakas saa sitten tuotteensa postitse kotiin. Jos luottokorttinumero on virheellinen tai maksu ei muusta syystä onnistu, ohjaa järjestelmä asiakkaan takaisin kassalle. Jos ostotapahtuma onnistuu, asiakas ohjataan pääsivulle. Ostoskori tietenkin tyhjenee kun ostokset on tehty onnistuneesti.

Sovellukselle tarvitaan myös erillinen hallintasivu jonka kautta ylläpitäjän on mahollista lisätä tuotteita ja niiden tietoja järjestelmään. Jokaisen tuotteen varastosaldo on tietenkin oleellista olla järjestelmällä tiedossa. Onnistuneen ostoksen tulee päivittää saldoa. Kun uusia tuotteita saapuu varastoon, ylläpitäjä kasvattaa saldoa.

Kaikki ostokset on siis pakko tehdä luottokortilla. Luottokorttimaksut tapahtuvat Luottokunnan verkkorajapintaa hyödyntäen. Asiakas ei tietenkään näe mitään, ohjelma hoitaa luottokunnan kanssa kommunikoimisen sisäisesti. Postituksia varten meillä on oma tietojärjestelmä. Verkkokauppa on siihen yhteydessä verkon välityksellä, ja jokaisesta onnistuneesta myyntitapahtumasta pitää lähteä tiedot (toimitusosoite ja ostetut tuotteet) postituksen tietojärjestelmään. Ylläpitäjän on hyvä päästä tarvittaessa näkemään lista kaikista luottokunnan kautta tehdyistä tehdyistä maksuista ja postitustietojärjestelmän kautta tehdyistä toimituksista.

- Tunnistakaa verkkokaupan käyttäjät
- Listatkaa kaikki verkkokaupan käyttötapaukset. Käyttötapauksesta riittää mainita nimi, käyttäjät, esiehto ja jälkiehto sekä käyttötapauksen kulku lyhyesti (kalvon 42 tapaan)
- **HUOM:** suositaan pienen osatoiminnallisuuden sisältäviä käyttötapauksia, esim. "asiakas valitsee tuotteita ja täyttää laskutustietonsa ja suorittaa maksun"on liian monta toiminnallisuutta sisältävä käyttötapaus ja kannattaa jakaa useaan erilliseen käyttötapaukseen
  - isompi toiminnallisuus koostetaan pienten käyttötapausten avulla
  - pienelle käyttötapaukselle tulee merkitä esiehdoksi ne aiemmat käyttötapaukset jotka oletetaan suoritetuksi ennen kuin käyttötapaus voidaan suorittaa (esim. ostoksen voi poistaa ostoskorista vain jos siellä on jo ostos)
- Piirtäkä käyttötapauskaavio

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

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

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

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

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

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

    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 a) {
        k.kasvataArvoa(a);
    }

    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 HKLLaitehallinto {
        ArrayList<Lataajalaite> lataajat = new ArrayList();
        ArrayList<Lukijalaite> lukijat = new ArrayList();

        lisaaLataaja(Lataajalaite lataaja){ lataajat.add(lataaja); }

        lisaaLukija(Lukijalaite lukija){ lataajat.add(lukija); }
}

public class Main {
    public static void main(String[] args) {
        HKLLaitehallinto laitteet = new HKLLaitehallinto();

        Lataajalaite rautatietori = new Lataajalaite();
        Lukijalaite ratikka6 = new Lukijalaite();
        Lukijalaite bussi244 = new Lukijalaite();

        laitehallinto.lisaaLataaja(rautatietori);
        laitehallinto.lisaaLukija(ratikka6);
        laitehallinto.lisaaLukija(bussi244);

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

        rautatietori.lataaArvoa(artonKortti, 3);

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