Ohjelmistojen mallintaminen, kurssikoe 14.12.2011

Vastaukset palautetaan tehtäväkohtaisiin pinoihin. Vaikka jättäisit johonkin tehtävään vastaamatta, tulee vastauspaperi siinäkin tapauksessa palauttaa. Toivomme että palautat myös lunttilappusi, käytämme niitä tieteellisessä tutkimuksessa, arvosteluun niillä ei ole mitään vaikutusta.

Kirjoita jokaiseen palauttamaasi konseptiin kurssin nimi, kokeen päivämäärä, nimi, nimikirjoitus ja opiskelijanumero. Merkitse opiskelijanumerosi myös lunttilappuun.

1. (6p) Selitä lyhyesti mutta kattavasti ja kokonaisin lausein seuraavat käsitteet
   (a) ohjelmiston määrittely
   (b) yksikkötestaus
   (c) single responsibility -periaate
   (d) code smell eli koodihaju
   (e) refaktorointi
   (f) TDD eli Test Driven Development
2. Luokka- ja sekvenssikaavioita.
   (a) (4p) Afrikan tähti on lautapeli, jossa 2-6 pelaajaa liikkuu eri puolilla Afrikkaa ja yrittävät löytää Afrikan tähden, eli ison timantin.
   Pelilauta koostuu ruuduista. Ruudut voivat olla liikkumisruutuja tai aarrepaikkoja. Näiden lisäksi pelissä on kaksi lähtöruutua. Aarrepaikkoihin liittyy mahdollisesti yksi aarre. Jokaisesta ruudusta voi siirtyä vähintään kahteen eri ruutuun. Yli kolmeen ruutuun ei voi siirtyä mistään ruudusta. Liikkumisruudut ovat joko merta tai maantietä. (Unohdamme lentoreitit tällä kertaa.)
   Aarrepaikoissa voi siis olla aarre. Aarteita on erilaisia: Afrikan tähti (joita koko pelissä on vain yksi), smaragdeja, rubiineja, hevosenkenkiä. Aarre voi myös olla rosvo. Pelaajan pelinappula on kulloinkin tietyssä ruudussa. Pelaajalla on myös tietty määrä rahaa.
   Mallinna tämä hieman yksinkertaistettu Afrikan tähti luokkakaaviona.
   (b) (1+4p) Tehtäväpaperin lopusta löytyy katkelma Java-koodia. Takaisinmallinna koodi luokka- ja sekvenssikaaviona. Sekvenssikaavio piirretään tilanteesta, jossa kutsutaan luokan Paaohjelma main-metodia. Luokkakaavioon ei tarvitse merkitä metodien nimiä.
3. Aiemmin vesiputousmallin nimiin vannonut yritys on päättänyt siirtyä soveltamaan ketteriä menetelmiä. Yritys päättää toteuttaa omiin tarpeisiinsa ketterää projektinhallintaa tukevan tietojärjestelmän. Yritykseen juuri palkattu Agile Coach, dosentti Arto Vihavainen kertoo seuraavassa mitä kaikkea tietojärjestelmällä tulisi pystyä tekemään.

   Ketterässä ohjelmistokehityksessä projektin toteutus jakautuu useisiin 1-2 viikon jaksoihin eli iteraatioihin. Product owner valitsee kuhunkin iteraatioon toteutettavaksi joukon backlogissa eli tehtävälistalla olevia tehtäviä. Product owner myös lisää uusia tehtäviä backlogiin sitä mukaa kun niitä generoidaan asiakaspalavereissa. Jokaiseen tehtävään liittyy joukko testaajien määrittelemiä hyväksymätestejä. Jossain tapauksessa hyväksymätesti voi liittyä useampaan tehtävään. Ohjelmoijat arvioivat kunkin tehtävän vaatiman työmäärän. Jokainen iteraation aikana toteutettavaksi valittu tehtävä annetaan yhden tai useamman ohjelmoijan sekä vähintään yhden testaajan vastuulle. Scrum master huolehtii, että tehtävälistaan on merkitty kunkin tehtävän vastuulliset tekijät. Tehtävälistalla oleviin tehtäviin liittyy status: ei aloitettu, aloitettu, testattavana, valmis. Scrum master päivittää tehtävälistalla olevien tehtävien statusta. Iteraation alussa iteraatioon valittujen tehtävien statuksesi tulee aloitettu ja tavoite on, että iteraation lopussa kaikkien tehtävien status on valmis. Iteraation päätteeksi Scrum master luo raportin, joka sisältää tiedot iteraatiossa toteutetuista tehtävistä. Scrum master siirtää valmiin raportin yrityksen toiminnanohjausjärjestelmään.

(a) (4p) Laadi kuvatusta tietojärjestelmästä karkean tason käyttötapausmalli, eli etsi käyttäjät ja käyttötapaukset. Määrittele kunkin käyttötapauksen kulku lyhyesti (maksimissaan rivi per käyttötapaus) tekstinä. Mainitse myös jokaisen käyttötapauksen yhteydessä siihen liittyvät käyttäjät sekä esiehdot. Piirrä myös käyttötapauskaavio.

(b) (1p) Kuvaa yksi käyttötapauksista tarkemmalla tasolla, ns. Cockburnin käyttötapauspohjan tai luentomonisteen tekstuaalisten käyttötapauskuvausten tyyliin.

(c) (4p) Laadi järjestelmän kuvauksen perusteella määrittelyvaiheen (eli kohdealueen) luokkakaavio. Merkitse yhteyksiin kytkentärajoitteet ja nimeä yhteydet ja yhteysroolit tarvittaessa. Ilmeisimmät attribuutit luokille kannattaa merkitä. Muista, että toiminnallisuutta ei kannata määrittelyvaiheen luokkamalliin laittaa!

Tehtävään 2 (b) liittyvä ohjelmakoodi:

```
public class Paaohjelma {
    public static void main(String[] args) {
        Henkilostorekisteri rekisteri = new Henkilostorekisteri();

        Henkilo arto1 = new Henkilo("Vihavainen", 1500, "1234-12345");
        rekisteri.lisaa(arto1);
        Henkilo arto2 = new Henkilo("Wikla", 4500, "4455-123123");
        rekisteri.lisaa(arto2);

        rekisteri.asetaPalkka("Vihavainen", 2100);
        rekisteri.suoritaPalkanmaksu();
    }
}

public class Henkilo {
    private String nimi;
    private int palkka;
    private String tilinumero;

    public Henkilo(String nimi, int palkka, String tilinumero) {
        this.nimi = nimi;
        this.palkka = palkka;
        this.tilinumero = tilinumero;
    }

    public String getNimi() {
        return nimi;
    }

    public void setPalkka(int palkka) {
        this.palkka = palkka;
    }

    public int getPalkka() {
        return palkka;
    }

    public String getTilinumero() {
        return tilinumero;
    }
}
```

```
public class Henkilostorekisteri {
    private ArrayList<Henkilo> henkilot;
    private PankkiRajapinta pankki;

    public Henkilostorekisteri() {
        henkilot = new ArrayList<Henkilo>();
        pankki = new PankkiRajapinta();
    }

    public void lisaa(Henkilo henkilo){
        henkilot.add(henkilo);
    }

    public void suoritaPalkanmaksu(){
        for (Henkilo henkilo : henkilot) {
            String nimi = henkilo.getNimi();
            String tiliNro = henkilo.getTilinumero();
            int palkka = getPalkka();
            pankki.maksaPalkka(nimi, tiliNro, palkka);
        }
    }

    public void asetaPalkka(String nimi, int uusiPalkka){
        Henkilo h = etsiHenkilo(nimi);

        if ( h!=null )
            h.setPalkka(uusiPalkka);
    }

    private Henkilo etsiHenkilo(String nimi) {
        for (Henkilo henkilo : henkilot) {
            if ( henkilo.getNimi().equals(nimi) )
                return henkilo;
        }

        return null;
    }

}


public class PankkiRajapinta {

    public void maksaPalkka(String nimi, String tilinumero, int summa) {
        // suorittaa maksun verkkopankin internet-rajapinnan avulla
        // yksityiskohdat piilotettu
    }
}
```