Ohjelmistotekniikan menetelmät, kurssikoe 16.12.2015

Vastaukset palautetaan tehtäväkohtaisiin pinoihin. Vaikka jättäisit johonkin tehtävään vastaamatta, tulee vastauspaperi siinäkin tapauksessa palauttaa.

Kirjoita jokaiseen palauttamaasi konseptiin kurssin nimi, kokeen päivämäärä, nimi, nimikirjoitus ja opiskelijanumero.

1. Testaus

   (a) (3p) Mitä tarkoitetaan ohjelmiston yksikkö-, integraatio-, järjestelmä- ja regressiotestaamisella?

   (b) (2p) Tee JUnit-testit, jotka testaavat tehtäväpaperin lopussa olevasta luokasta **Palkkio** seuraavat asiat:
   - metodin *summa* paluuarvo on (normaalitapauksissa) konstruktorien parametrien tulo
   - metodin *summa* paluuarvo on nolla jos jompikumpi tai molemmat konstruktorin parametreista ovat arvoltaan negatiivisia

   (c) (1p) Tee JUnit-testi, joka varmistaa että luokan **Henkilo** metodi *palkatYhteensa* laskee palkan oikein tapauksessa, jossa henkilön metodia *lisaaPalkkio* on kutsuttu muutamaan kertaan järkevillä parametrien arvoilla.

   Tehtäväpaperin lopussa on pieni JUnit-esimerkki muistinvirkistykseksi.

2. Luokka- ja sekvenssikaavioita.

   (a) (3p) Terraariossa on monia erilaisia eliöitä. Eliöt ovat joko kasveja tai liikkuvia eliöitä. Kukin eliö kuuluu pysyvästi johonkin lajiin. Osa lajeista on toistensa vihollisia. Yhdellä lajilla voi olla useita eri vihollislajeja. Liikkuvat eliöt muodostavat laumoja. Laumaan kuuluu useita eliöitä, mutta eliö voi kerrallaan kuulua vain yhteen laumaan. Liikkuvat eliöt toteuttavat liikkumisominaisuuden. Liikkumisominaisuus säätelee eliön liikkumista terraariossa. Myös laumat toteuttavat oman liikkumisominaisuutensa joka vaikuttaa jokaisen lauman sisältämän eliön liikkeeseen.
   Mallinna terraario 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 statukseksi 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) (3p) 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.

4. (4p) Laadi tehtävän 3 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!

Tehtäviin liittyvä ohjelmakoodi:

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

        hallinto.lisaaHenkilo("Arto", "assistentti");
        hallinto.lisaaHenkilo("Sasu", "professori");

        hallinto.lisaaTunteja("Arto", 3);
        hallinto.lisaaTunteja("Sasu", 20);

        hallinto.raportoi();
    }
}


public class HenkilostoHallinto {
    private ArrayList<Henkilo> henkilot;
    private Palkanlaskin palkanlaskin;

    public HenkilostoHallinto() {
        henkilot = new ArrayList<>();
        palkanlaskin = new Palkanlaskin();
    }

    public void lisaaHenkilo(String nimi, String virka) {
        Henkilo h = new Henkilo(nimi, virka);
        henkilot.add(h);
    }

    public void lisaaTunteja(String nimi, int tunnit) {
        for (Henkilo henkilo : henkilot) {
            if ( henkilo.getNimi().equals(nimi)) {
                String virka = henkilo.getVirka();
                int tuntipalkka = palkanlaskin.tuntiPalkka(virka);
                Palkkio palkkio = new Palkkio(tunnit, tuntipalkka);
                henkilo.lisaaPalkkio(palkkio);
            }
        }
    }

    public void raportoi() {
        for (Henkilo henkilo : henkilot) {
```

```
            henkilo.raportoi();
        }
    }

}


public class Henkilo {
    private String nimi;
    private String virka;
    private ArrayList<Palkkio> palkkiot;

    public Henkilo(String nimi, String virka) {
        this.nimi = nimi;
        this.virka = virka;
        palkkiot = new ArrayList<>();
    }

    public String getNimi() {
        return nimi;
    }

    public String getVirka() {
        return virka;
    }

    public void lisaaPalkkio(Palkkio palkkio) {
        palkkiot.add(palkkio);
    }

    public int palkatYhteensa() {
        int palkkaSumma = 0;
        for (Palkkio palkkio : palkkiot) {
            palkkaSumma += palkkio.summa();
        }

        return palkkaSumma;
    }

    public void raportoi() {
        System.out.println(nimi+ " "+virka+ " palkkoja "+palkatYhteensa()+" euroa");
    }
}

public class Palkkio {
    private int tunnit;
    private int tuntipalkka;

    public Palkkio(int tunnit, int tuntipalkka) {
        this.tunnit = tunnit;
        this.tuntipalkka = tuntipalkka;
    }

    public int summa() {
        return tunnit*tuntipalkka;
    }
}
```

```
public class Palkanlaskin {
    public int tuntiPalkka(String virka) {
        if (virka.equals("professori")) {
            return 50;
        } else if (virka.equals("lehtori")) {
            return 30;
        } else if (virka.equals("assistentti")) {
            return 20;
        } else if (virka.equals("pajaohjaaja")) {
            return 1;
        }

        return 0;
    }
}

public class HenkiloTest {

    @Test
    public void nimiOikein(){
        Henkilo h = new Henkilo("Leo", "pajaohjaaja");
        assertEquals("pajaohjaaja", h.getVirka());
    }

}
```