Ohjelmistotekniikan menetelmät, kurssikoe syksy 2016

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

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

1. (a) (3p) Ohjelmistojen testaus jakaantuu muutamaan erilaiseen aktiviteettiin tai työvaiheeseen, joita kutsutaan myös testaustasoiksi. Kerro mitä testaustasoja on olemassa, ja mitä kullakin testaustasolla tarkoitetaan.

   (b) (2p) Laskareissa tutustuttiin Git-versionhallintajärjestelmään. Mitä versionhallinnalla tarkoitetaan, ja mitä hyötyä sen käyttämisestä on? Kirjoita aiheesta lyhyt, noin 0.5 sivun essee.

   (c) (2p) Kurssin aikana käsiteltiin keskeisiä oliosuunnittelun periaatteita. Nimeä näistä periaatteista mielestäsi kaksi tärkeintä ja selitä lyhyesti mitä ne tarkoittavat.

2. (a) (3p) Tehtäväpaperin lopusta löytyy katkelma Java-koodia. Takaisinmallinna koodi luokkakaaviona. Luokkakaavioon ei tarvitse merkitä metodien nimiä, yhteyksien laatu tulee merkitä mahdollisimman tarkasti!

   (b) (4p) Takaisinmallinna tehtäväpaperin lopun koodi sekvenssikaaviona. Sekvenssikaavio piirretään tilanteesta, jossa kutsutaan luokan Paaohjelma main-metodia.

3. (4p) Matkayhtiö tarjoaa matkoja kohteisiin ympäri maailman. Kaikkiin matkoihin liittyy kohde ja lennot (eli meno- ja paluulippu kohteeseen tiettyinä päivämäärinä ja tiettyyn aikaan). Osa matkoista on pakettimatkoja joiden hintaan kuuluu myös majoitus. Pakettimatkoja on kahta tyyppiä, rantalomamatkoja ja kulttuurimatkoja. Pakettimatkoihin kuuluu myös erilaista järjestettyä ohjelmaa joilla on tietty ajankohta ja paikka. Kaikilla kulttuurimatkoilla ohjelmassa on ainakin yksi opastettu kiertue ja kaikilla rantalomamatkoilla useampi jumppatuokio.

   Mallinna tilanne luokkakaaviona.

4. (4p) Mehtosen-taksi yrityksellä on useita ajoneuvoja, joihin kaikkiin on määrätty yksi kuski. Kun kuski on töissä hän voi olla merkittynä vapaaksi (vapaana ottamaan uusia asiakkaita), varatuksi (kiireinen toisen asiakkaan kanssa) tai tauolla (ei ota vastaan uusi asiakkaita).

   Kun asiakas haluaa saada taksikyydin hän käyttää puhelinapplikaatiota luodakseen uuden ajopyynnön, jossa asiakas kertoo, mistä hän tarvitsee taksikyydin ja mihin aikaan. Hallintahenkilökunnan jäsen käy listaa pyydetyistä taksikyydeistä läpi ja merkitsee ajopyynnöt toteutettavaksi kuskille, jotka ovat tarpeeksi lähellä ja jonka tila on merkitty vapaaksi. Kuski voi hyväksyä tai hylätä ajopyynnön listasta, jossa näkyy kaikki hänelle merkityt ajopyynnöt. Jos kuski merkitsee ajopyynnön hyväksytyksi, tulee asiakkaalle ilmoitus taksin arvioidusta saapumisajasta. Kun asiakas on poimittu kyytiin voi asiakas tarkastella puhelimestaan tähän mennessä ajettua matkaa ja matkasta kertynyttä maksua. Matkan päätyttyä asiakas maksaa matkan hinnan suoraan puhelimellaan. Maksun yhteydessä järjestelmä veloittaa asiakkaan luottokorttia käyttäen luottokunnan tarjoamaa rajapintaa. Kun matka on päättynyt ja se on maksettu voi asiakas antaa palautetta matkastaan asteikolla 1 - 5 tähteä.

   (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, luentomonisteen tekstuaalisten käyttötapauskuvausten tyyliin.

Tehtäviin liittyvä ohjelmakoodi:

```
public class Paaohjelma {

    public static void main(String[] args) {
        Kuvagalleria galleria = new Kuvagalleria();
        galleria.rekisteroiKayttaja("marko96", "marko@example.com");
        galleria.rekisteroiKayttaja("minna_m", "minna.meikalainen@example.com");
        galleria.lataaKuva("marko96", "uusi_pyora", new File("pyora.jpg"));
        galleria.lisaaArvioKuvalle("uusi_pyora", "marko96", 5);
        galleria.lisaaArvioKuvalle("uusi_pyora", "minna_m", 3);
        System.out.println("uusi_pyora kuvan arvioiden keskiarvo: "
                           + galleria.getKuvanArvioidenKeskiarvo("uusi_pyora"));
    }
}

public class Kuvagalleria {
    private Map<String, Kayttaja> kayttajat;
    private Map<String, Kuva> kuvat;

    public Kuvagalleria() {
        this.kayttajat = new HashMap<String, Kayttaja>();
        this.kuvat = new HashMap<String, Kuva>();
    }

    public void rekisteroiKayttaja(String kayttajanimi, String email) {
        Kayttaja uusikayttaja = new Kayttaja(kayttajanimi, email);
        kayttajat.put(kayttajanimi, uusikayttaja);
    }

    public void lataaKuva(String kayttajanimi, String kuvanNimi, File kuvaTiedosto) {
        Kayttaja lataaja = kayttajat.get(kayttajanimi);
        if (lataaja == null) {
            return;
        }
        Kuva uusiKuva = new Kuva(lataaja, kuvanNimi, kuvaTiedosto);
        kuvat.put(kuvanNimi, uusiKuva);
    }

    public void lisaaArvioKuvalle(String kuvannimi, String kayttajanimi, int arvio) {
        Kuva kuva = kuvat.get(kuvannimi);
        Kayttaja arvioija = kayttajat.get(kayttajanimi);
        if (kuva == null || arvioija == null) {
            return;
        }
        kuva.lisaaArvio(arvioija, arvio);
    }

    public double getKuvanArvioidenKeskiarvo(String kuvannimi) {
        Kuva kuva = kuvat.get(kuvannimi);
        if (kuva == null) {
            return 0.0;
        }
        return kuva.arvioidenKeskiarvo();
    }
}
```

```
public class Kuva {
    private Kayttaja lataaja;
    private String nimi;
    private File kuvatiedosto;
    private List<Arvio> arviot;

    public Kuva(Kayttaja omistaja, String nimi, File tiedosto) {
        this.lataaja = omistaja;
        this.nimi = nimi;
        this.kuvatiedosto = tiedosto;
        this.arviot = new ArrayList<Arvio>();
    }

    public void lisaaArvio(Kayttaja arvioija, int arvio) {
        for (Arvio a : arviot) {
            if (a.getArvioija().getNimi().equals(arvioija.getNimi())) {
                a.setArvio(arvio);
                return;
            }
        }
        Arvio uusiArvio = new Arvio(arvioija, arvio);
        arviot.add(uusiArvio);
    }

    public double arvioidenKeskiarvo() {
        if (arviot.isEmpty()) {
            return 0.0;
        }
        int arvioidenSumma = 0;
        for (Arvio a : arviot) {
            arvioidenSumma += a.getArvio();
        }
        return (double) arvioidenSumma / arviot.size();
    }
}

public class Arvio {
    private Kayttaja arvioija;
    private int arvio;

    public Arvio(Kayttaja arvioija, int arvio) {
        this.arvioija = arvioija;
        this.arvio = arvio;
    }

    public Kayttaja getArvioija() {
        return arvioija;
    }

    public int getArvio() {
        return arvio;
    }

    public void setArvio(int arvio) {
        this.arvio = arvio;
    }
}
```

```
public class Kayttaja {
    private String kayttajanimi;
    private String email;

    public Kayttaja(String kayttajanimi, String email) {
        this.kayttajanimi = kayttajanimi;
        this.email = email;
    }

    public String getNimi() {
        return kayttajanimi;
    }
}
```