Ohjelmistojen mallintaminen, syksy 2011, laskuharjoitus 6

huom: laskarit pidetään salissa B221

Kaikki alla olevat tehtävät tehdään etukäteen. Paikanpäällätehtävät tehtävät löytyvät laskarisivulta https://wiki.helsinki.fi/pages/viewpage.action?pageId=76261353

1. Asiakaspalaverin aikana järjestelmää tilaava asiakas kertoi seuraavaa:

   > Yrityksellämme on joukko ruokien ja tuotteiden asetteluun erikoistuneita työntekijöitä, joiden toimenkuvana on kerätä ideoita ja kuvia ketjumme kahviloiden ja kauppojen asetteluista. Tällä hetkellä käytämme sähköpostia digikameralla otettujen kuvien siirtämiseen, mutta haluaisimme jotain nykyaikaisempaa.
   >
   > Tavoitteenamme on saada teiltä järjestelmä johon työntekijämme voivat lisätä työpäivien aikana kerättyjä kuvia. Varustamme työntekijämme kännykkäkameroilla sekä normaalileilla digikameroilla. Järjestelmän tulee tukea kuvien vastaanottamista sekä MMS-viestinä suoraan kännyköistä, sekä www-sivuna näkyvän lähetyslomakkeen kautta.
   >
   > Jokaisella työntekijällä on käyttäjätunnus järjestelmään, jota käytetään myös lähetetyissä kuvissa tietona. Kuvia katsottaessa pitää siis näkyä kenen lähettämä kuva on. Jokaiseen käyttäjätunnukseen liittyy myös yksi tai useampi puhelinnumero, jonka avulla MMS-viestinä lähetettyjen kuvien lähettäjä saadaan myös selville.
   >
   > Järjestelmän pitää tarjota mahdollisuus kuvien listaamiseen eri kategorioihin perustuen, esimerkiksi sesonkeihin perustuen. Sesonkikategoriaan kuuluvat muunmuassa joulukuvat, pääsiäiskuvat jne. Jokainen kuva voi olla yhdessä tai useammassa kategoriassa, ja niitä pitää myös pystyä hakemaan kategorioiden perusteella.
   >
   > Käyttäjät voivat myös kommentoida toisten lisäämiä kuvia, sekä antaa niille suosituksia *like*-tyylisesti. Normaalien työntekijöiden lisäksi järjestelmää käyttävät myös ns. managerit, jotka eivät itse lähetä kuvia järjestelmään mutta voivat kommentoida niitä. Managereiden pitää myös pystyä luomaan raportteja lähetetyistä kuvista. Kerran viikossa sihteerit siirtävät viikon aikana valmistuneet raportit järjestelmästä erilliseen toiminnanohjausjärjestelmäämme.

   Tee ylläolevasta ohjelmakuvauksesta käyttötapausmalli. Eli etsi käyttäjät (kertaa viikon 1 kalvoilta mikä on käyttötapauksen käyttäjä eli englanniksi actor), listaa käyttötapaukaukset (tarkkaa kuvausta ei tarvita, nimi, käyttäjät sekä esiehdot riittävät) ja piirrä käyttötapausdiagrammi.

2. Tee ylläolevasta kuvauksesta määrittelyvaiheen luokkakaavio, joka tarkentaa kuvienhallintajärjestelmän käsitteistön ja käsitteiden suhteet.

Kuten viikon 4 luentokalvoilla 17 tähdennetään, **älä lisää toiminnallisuutta määrittelyvaiheen luokkakaavioon**, älä mieti kuka tekee ja mitä, keskity luokkien olioiden pysyväkuontoisiin yhteyksiin.

3. Ohjelmoinnin perusteiden laskareissa 2 ja 3 tehtiin kokonaislukujoukkojen käsittelyä varten komentotulkki. Perinteiden tapa toteuttaa komentotulkki on tehdä iso valintarakenne jossa on oma if-haara jokaista komentoa kohti. Rakenne on hieman ikävä ja laajennettavuudeltaan huono sillä uuden komennon lisääminen edellyttää valintarakenteen todennäköisesti muutosten tekemistä useampaan paikkaan koodia.

   Vaihtoehtoinen "olio-orientoitunut"tapa tehdä komentotulkki on tehdä jokaisesta esillisestä komennosta abstraktin luokan Komento aliluokkia.

   Osoittesta `https://wiki.helsinki.fi/display/ohma/Komentotulkki` löytyy ohjelma jossa kokonaislukujoukkoja käsitellään olio-orientoidun komentotulkin avulla (ohjelmassa ei ole valmiiksi toteutettu kuin muutama komento.).

   Kokeile ohjelmaa ja selvitä itsellesi sen toimintaperiaate. Ohjelman rakennetta ja toimintaperiaatetta ei välttämättä ole aluksi ihan helppo ymmärtää.

   Kuvaa ohjelman rakenne luokkakaaviona.

   Huom: oliolla olevan HashMap:in voi ajatella luokkakaavion suhteen olevan samanlainen kuin ArrayList, eli sitä ei kannattane luokkakaavioon merkitä omana luokkanaan.

4. Kuvaa sekvenssikaaviona mitä tapahtuu kun main-metodi luo IntJoukkoSovellusolion.

   Kuvaa sekvenssikaaviona mitä tapahtuu kun main käynistää sovelluksen ja käyttäjä antaa komentoriville seuraavat syötteet `lisaa <enter> a <enter> 3 <enter>` (eli käyttäjä pyytää lisäämään joukkoon nimeltä *a* luvun 3).

   Oliolla olevan HashMap:in voi ajatella myös sekvenssikaavion suhteen olevan samanlainen kuin ArrayList, eli sitä ei kannattane merkitä kaavioon omana oliona.

   bonus: toteuta ohjelmalle lisää komentoja, esim. seuraavat:

   - alkion poisto valitusta joukosta
   - uuden joukon luominen
   - joukkojen unioni, leikkaus ja erotus

5. Luennoilla ja kalvoissa on mainittu 3 oliosuunnittelun periaatetta:

   - **Single responsibility principle**

     Luokilla tulisi olla vain yksi selkeä vastuu. Vastuulla oikeastaan tarkoitetaan potentiaalista syytä muutokselle. Ei ole esimerkiksi viisasta laittaa käyttäjän syötteen lukemista minkään sovellusolion tehtäväksi. Syötteen lukeminen on oma selkeä vastuunsa ja sitä varten tulisi olla oma luokka. Sovellusoliot sitten tarpeen vaatiessa kutsuvat syötteenlukijaolioita.

- **Program to interfaces, not to concrete implementations**

  Älä ohjelmoi siten, että olet riippuvainen konkreettisista luokista, on järkevämpi olla riippuvainen ainoastaan rajapinnasta. Esim. lypsyrobootista (ks. viime viikon tehtävä *Maidon elämää) tulee monikäyttöisempi kun se määrittelee lypsettävän olevan joku rajapinnan Lypsava toteuttava olio:*

```
public void lypsa(Lypsava lypsettava) {
    // ...
    double maitoa = lypsettava.lypsa();
    sailio.lisaaSailioon(maitoa);
}
```

- **Favor composition over inheritance**

  Rakenna toiminnallisuutta monimutkaisen perintähierarkian sijaan liittämällä yhteen joukko selkeän vastuun omaavia "pieniä" olioita. Hyvänä esimerkkinä tästä ohjelmoinnin jatkokurssin viikon 3 MuistavaTuotevarasto jota käsiteltiin luennolla

  Tämän periaatteen soveltaminen on siis yksi keino, jonka avulla saadaan oliot noudattamaan single responsibility -periaatetta.

- *Muitakin periaatteita on olemassa, joihinkin niistä palataan kurssin aikana*

Tehtävät:

(a) Toteuttaako viime viikon tehtävä *Maidon elämä* kaikkia em. periaatteita? Jos ei niin mikä rikkoutuu?

(b) Toteuttaako tehtävien 3 ja 4 olio-orientoitu komentotulkki kaikkia em. periaatteita? Jos ei niin mikä rikkoutuu?

(c) Etsi ohjelmoinnin perusteiden ja jatkokurssin sekä tämän kurssin materiaaleista (luentomateriaalista ja laskareista) esimerkkejä (ainakin 3 esimerkkiä) jotka rikkovat jonkun ym. periaatteista. Näytä miten korjaisit tilanteen.

(d) Minkä takia em. periaatteet ovat tärkeitä? Saako niitä rikkoa? Missä tilanteissa?

6. Luentomonisteessa ja monisteessa mainituilla www-sivuilla on lueteltu joitain koodihajuja (code smell), jotka ovat siis kooditasolla näkyviä merkkejä laiskasta ohjelmointityylistä tai huonosta oliosuunnittelusta. Koodihaju yleensä johtaa jossain vaiheessa ongelmiin jos koodia on muutettava.

   Etsi omista ohjelmoinnin perusteiden ja ohjelmoinnin jatkokurssin laskarivastauksistasi ainakin 4 esimerkkiä, joissa esiintyy jotain koodihajuja.

7. **Anna kurssipalautetta** osoitteessa http://ilmo.cs.helsinki.fi/kurssit/servlet/Valinta

   Haluamme paljon palautetta: mikä toimi, mikä ei, mitä käsiteltiin liikaa mitä liian vähän, jne.