TITE2050 Olio-ohjelmointi **Kevät 2016**

Harjoitustyö

Voit ehdottaa harjoitustyön aiheeksi haluamaasi aihetta. Aiheelle täytyy kuitenkin hakea hyväksyntä ennen työn virallista aloitusta. Oman aiheen tulee olla vähintään suunnilleen yhtä haastava ja laaja kuin tässä esitetty aihe. Harjoitustyön saa tehdä yksin, tai korkeintaan kolmen hengen ryhmissä.

Oletusaiheena on tehdä urheilukilpailujen pistelaskujärjestelmäksi soveltuva ohjelma. Ohjelman käyttöliittymänä voi toimia konsoli eli ns. komentokehoite. Ohjelmalle voi halutessaan tehdä graafisen käyttöliittymän (GUI) Javan Swing-kirjaston avulla. Tämä on kuitenkin huomattavasti työläämpi tai vaativampi vaihtoehto.

Ohjelman käynnistyessä käyttäjä voi joko luoda uuden kilpailun tai lukea aiemmin syötetyn kilpailun tiedot tiedostosta tai SQL-tietokannasta. Jos kyseessä on uusi kilpailu, käyttäjä antaa ensin kilpailun nimen. Sen jälkeen käyttäjä voi hyödyntää seuraavia toimintoja.

- Käyttäjä voi lisätä kilpailuun uusia kilpailijoita. Jokainen kilpailija esitetään oliona luokasta `Kilpailija`, joka perii luokan `Henkilo`. Jokaisella kilpailijalla on yksilöllinen kilpailija-numero, jota ei voida vaihtaa kilpailun aikana. Sellaiset tiedot, jotka koskevat myös ei-kilpailevia henkilöitä (esimerkiksi toimitsijoita), tulee tallentaa luokan `Henkilo` oliokohtaisiin attribuutteihin. Henkilön suku- ja etunimen tulee täyttää seuraavat ehdot.
    - Nimen tulee sisältää vähintään yksi ja korkeintaan 64 merkkiä.
    - Nimi saa sisältää vain isoja ja pieniä kirjaimia sekä yhden tai useamman tavuviivan.

  Kilpailijoita tulisi voida myös poistaa. Kilpailija voi osallistua useaan eri lajiin, mutta tästä huolimatta ohjelman tulisi käyttää vain yhtä `Kilpailija`-luokan oliota yhtä henkilöä kohden.

- Aiemmin syötettyjen kilpailijoiden tietoja voidaan muuttaa, jos esimerkiksi nimi oli aiemmin syötetty väärin.

- Käyttäjä voi määritellä kilpailuun uusia kilpailtavia lajeja. Jokaiselle lajille tulee antaa nimi. Käyttäjä voi määritellä mihin lajeihin eri kilpailijat osallistuvat. Kukin kilpailija voi osallistua useaan eri lajiin.

- Käyttäjä voi määritellä kuhunkin lajiin useita sarjoja. Sarjalle voidaan määrätä rajoituksia sen suhteen, ketkä saavat kilpailla kyseisessä sarjassa. Esimerkiksi, sarjassa *Yleinen* saavat kilpailla kaikki. Sarjassa *Naiset* saavat kilpailla vain naiset. Sarjassa *Y20* saavat kilpailla kaikki alle 20-vuotiaat, ja sarjassa *N20* saavat kilpailla vain alle 20-vuotiaat naiset. Seniori-sarjassa *Y50* saavat kilpailla vain 50 vuotta täyttäneet. Lajiin osallistuva kilpailija osallistuu automaattisesti kaikkiin sarjoihin joihin hän voi osallistua. Kilpailija saa vain yhden tuloksen lajia kohti, ja hän osallistuu tuolla samalla tuloksella jokaiseen sarjaan jossa hän kilpailee. Jokaiselle sarjalle laaditaan erillinen tulosluettelo.

- Käyttäjä voi määritellä joukkueita sekä niihin kuuluvat kilpailijat. Joukkue voi osallistua lajeihin kuten yksittäinen kilpailija. Joukkue voi kilpailla annetussa sarjassa jjos kaikki siihen kuuluvat kilpailijat kilpailevat kyseisessä sarjassa. Joukkueen pistemäärä on siihen kuuluvien kilpailijoiden pistemäärien summa. Joukkueille laaditaan tulosluettelot kuten yksittäisille kilpailijoille. Yhtä joukkuetta varten ohjelmassa tulee olla vain yksi `Joukkue`-luokan olio.

- Käyttäjä voi syöttää kilpailijoiden tuloksia eri lajeissa.

- Kun kaikki tarpeelliset tulokset on syötetty, käyttäjä voi tulostaa eri lajien ja eri sarjojen tulosluetteloita. Näissä luetteloissa lajiin osallistuneet kilpailijat luetellaan heidän pistemäärän mukaisessa järjestyksessä (myös pistemäärä ja sijoitus mainitaan). Suurempi pistemäärä on parempi.

- Käyttäjä voi tallentaa syöttämänsä tiedot tiedostoon tai SQL-tietokantaan. Tiedot voidaan myöhemmin lukea takaisin ohjelmaan. Jos tiedot tallennetaan tiedostoon, käyttäjä voi halutessaan suojata tiedoston salasanalla, siten että tiedoston sisällön lukemiseen tarvitaan kyseinen salasana. Käytetty tallennusmuoto tulee dokumentoida siten, että toinen ohjelmoija tai ohjelmoijaryhmä voisi sen perusteella kirjoittaa ohjelman, joka lukee näitä tiedostoja. (Dokumentoinniksi ei riitä se, että *käytimme Javan version x.y* `Object*Stream`*-luokkaa*.)

- Käyttäjä voi suorittaa näitä tehtäviä erilaisissa järjestyksissä, eli aina ei ole pakko edetä ennalta määrätyssä järjestyksessä syötä kilpailijat → syötä lajit → syötä tulokset → tulosta.

- Ohjelma paketoidaan suoritettavaksi JAR-paketiksi. Jos paketin nimi on esimerkiksi `pistelasku.jar`, ohjelma tulee voida käynnistää komennolla `java –jar pistelasku.jar`. JAR-paketin tulee sisältää `.class`-tiedostojen lisäksi myös ohjelman lähdekoodi (`.java`-tiedostot).

- Ohjelman toteutus (eli lähdekoodi) tulee dokumentoida riittävällä tarkkuudella käyttäen joko JavaDoc tai Doxygen -muotoisia kommentteja. Dokumentointia tarvitaan lähinnä ohjelman eri osien rajapinnoille, sekä erikoisille tai omalaatuisille toteutuksille tai suunnittelu-ratkaisuille.

- Ohjelman lähdekoodiin tulee merkitä kuka tai ketkä ovat tehneet kyseisen toteutuksen (opiskelijanumero ja nimi).

Toteutuksen tulee myös täyttää seuraavat yleiset vaatimukset.

- Kaikki ei-vakio attribuutit tulee olla joko *suojattuja* (*protected*) tai *yksityisiä* (*private*).
- Lähdekoodin tunnisteissa, kuten luokkien ja muuttujien nimissä, saa käyttää vain ASCII-merkistön merkkejä. Toisin sanoen, ääkkösiä ei saa käyttää muuttujien tai luokkien nimissä.
- Jokaisen luokan tulee olla pakkauksessa.
- Jokainen luokka tulee tallentaa omaan tiedostoonsa, siten että tiedoston nimi on sama kuin luokan nimi.

Seuraavat kohdat tehtävänannosta eivät ole pakollisia, mutta niiden puuttuminen vaikuttaa arvosanaan. Mitä ”yksinkertaisempia” asioita toteutuksesta puuttuu ja mitä enemmän puutteita löytyy, sitä suurempi on vaikutus arvosanaan.

- Kilpailijoiden nimien pituus- ja (muut) muotovaatimukset.
- Kilpailijoiden poistaminen.
- Sarjat voidaan jättää toteuttamatta. Eli ikään ikuin jokaisessa lajissa kilpailtaisiin vain yksi sarja.
- Joukkueet voidaan jättää toteuttamatta.
- Tietojen tallentaminen tiedostoon tai SQL-tietokantaan.
- Tiedoston suojaus salasanalla.

Myös toteutuksen yleinen huolellisuus vaikuttaa arvosanaan. Esimerkiksi, jos ohjelma odottaa käyttäjän syöttävän kokonaisluvun, mutta käyttäjä antaakin merkkijonon ja tämä aiheuttaa ohjelman ”kaatumisen”, arvosana saattaa heikentyä.

Toteutuksessa voi halutessaan käyttää `Lue`-luokkaa, sekä muita `uva`-pakkauksesta löytyviä luokkia. Erityisesti `uva.BER`-pakkaus voi olla hyödyllinen, jos tietojen tallentamiseen käytetään BER-enkoodausta.