Tiina Tuomisto & Ella Rauma

URHEILUPÄIVÄKIRJA

HARJOITUSTYÖN RAPORTTI

1.12.2020

TIEA1130 Oliosuuntaunut suunnittelu ja ohjelmointi

Kokkolan yliopistokeskus Chydenius

Jyväskylän yliopisto

Sisällysluettelo

1. Johdanto

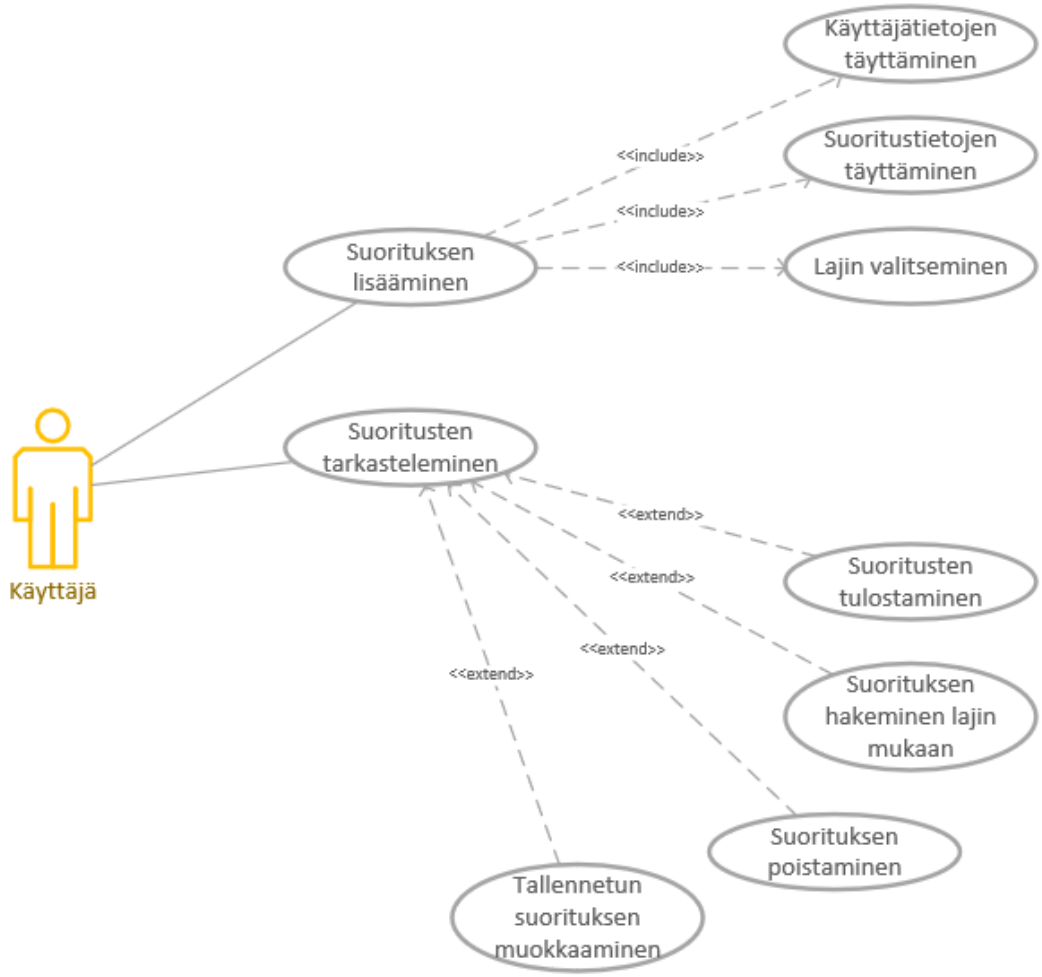
Oliosuuntautunut suunnittelu ja ohjelmointi -kurssi piti sisällään harjoitustyön, jonka toimeksiantona oli tuottaa ohjelmisto. Aihealueen ja tarkemmat toiminnallisuudet saivat opiskelijat valita itse, mutta keskeistä oli ohjelmiston oliopohjaisuus sekä graafisen käyttöliittymän teko. Halusimme toteuttaa reaalimaailmaankin hyödyllisen sovelluksen, joka liittyy urheiluun. Toteutimme urheilupäiväkirjan, jonka avulla käyttäjä voi pitää kirjaa urheilusuorituksistaan. Ohjelmointikielenä ohjelmistossamme on käytetty Javaa ja käyttöliittymä on toteutettu NetBeans IDE 11.2 -versiolla. Tämä päiväkirjan ensimmäinen versio sisältää runsaasti jatkokehitysmahdollisuuksia, mutta on jo itsessään toimiva kokonaisuus urheilusuoritusten seurantaan.

2. Palvelun kuvaus

Tässä kappaleessa kuvaamme ohjelmistoa sen käyttötapausten kautta, esittelemme käyttötapauskaavion ja merkittävimmän käyttötapauskuvauksen sekä syvennymme luokkakaavioon.

2.1. Käyttötarkoitus ja käyttötapauskaavio

Ohjelmisto on tarkoitettu urheilusuoritusten seuraamiseen. Käyttäjä voi syöttää omat tietonsa, valita urheilulajin, päivämäärän, paikan, urheilusuorituksen keston sekä intensiteetin ja tallentaa suorituksen tietokantaan. Käyttäjä voi lisäksi selata menneitä urheilusuorituksia, muokata niitä sekä poistaa jo tallennetun suorituksen. Ohjelmaan on tallennettu valmiiksi tietyt suosituimmat urheilulajit, mutta näiden ohella käyttäjä voi valita ”muu”-osion, mikäli lajia ei löydy listasta.



Kuva 1. Käyttötapauskaavio

2.2. Käyttötapauskuvaus

Ohjelmassa on kaksi pääasiallista käyttötapausta: suorituksen lisääminen sekä suoritusten tarkasteleminen. Niistä voidaan johtaa useampi muu käyttötapaus, kuten edellisen osion käyttötapauskaaviosta nähdään. Tarkastelemme tässä osiossa suorituksen lisäämistä käyttötapauksen kuvauksen näkökulmasta.

Käyttäjätapauksen nimi	Suorituksen lisääminen
Suorittaja	Käyttäjä (Tavoitteellisesti urheileva henkilö)
Esitilanne	Käyttäjä on tehnyt urheilusuorituksen ja haluaa kirjata sen urheilupäiväkirjaansa
Kuvaus	Käyttäjä syöttää omat sekä suorituksensa tiedot järjestelmään
Jälkitilanne	Käyttäjä tallentaa suorituksen ja tarkastaa, että se on tallentunut oikein
Poikkeukset	Järjestelmä ei sisältänyt käyttäjän suorittamaa lajia, joten hän joutui valitsemaan ”muu”

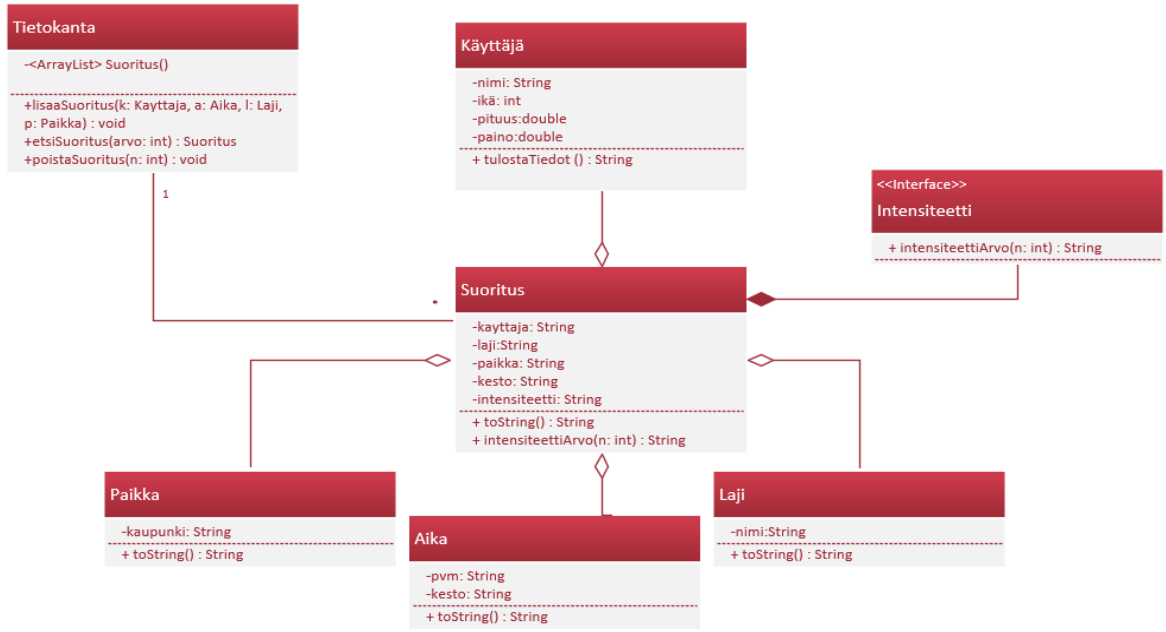
Taulukko 1. Käyttötapauskuvaus

3. Ohjelmisto

Tässä osiossa perehdymme ohjelmiston luokkarakenteeseen, käyttöliittymään sekä koodin näkökulmasta keskeisimpiin kokonaisuuksiin.

3.1. Luokkakaavio ja koodi

Ohjelmisto on rakennettu siten, että Suoritus-olio on keskiössä. Suoritus-luokka toteuttaa Intensiteetti-rajapinnan ja koostuu Paikka, Aika, Laji ja Käyttäjä -olioluokista. Suoritukset tallentuvat tietokantaan ArrayList-rakenteeseen. Tietokanta sisältää toiminnallisuuden kannalta keskeisimmät metodit, eli suorituksen lisäämisen, etsimisen sekä poistamisen.



Kuva 2. Luokkakaavio

Metodissa lisaaSuoritus, lisätään ArrayList:iin suorituksia luomalla Suoritus-luokan esiintymä ja tallentamalla se heti listaan. Metodin sisällä on myös toiminta, jossa määritellään intensiteettiin liittyvä arvo, käyttäjän valitseman lajin perusteella.

```
public void lisaaSuoritus(Kayttaja k, Aika a, Laji l, Paikka p){
    int i=0;
    if(l.nimi.equals("Jalkapallo") || l.nimi.contains("Koripallo") || l.nimi.contains("Tennis")){
        i=1;
    }
    else if(l.nimi.equals("Juoksu") || l.nimi.equals("Pyöräily") || l.nimi.equals("Uinti") || l.nimi.equals("Sulkapallo")){
    i=2;
    }
    else if(l.nimi.equals("Kävely")){
        i=3;
    }
    else{
        i=4;
    }

    suoritukset.add(new Suoritus(k, a, l, p, i));
}
```

Kuva 3. Suorituksen lisäys

Oleellista koodia ohjelman toiminnan kannalta on myös btnTulostaActionPerformed -metodissa, jonka avulla muut komponentit saadaan toimimaan ja tallennettuja suorituksia tulostetaan listaan. Metodissa sallitaan poisto- ja muokkausnappien käyttäminen, kun ArrayList ei ole tyhjä.

```
private void btnTulostaActionPerformed(java.awt.event.ActionEvent evt) {
    lstPaivakirja.removeAll(); //Tyhjennetään päiväkirjan tulostuskenttä ennen seuraavaa tulostusta

    //Tulostetaan päiväkirjan kaikki suoritukset
    for (Suoritus s : t.suoritukset) {
        lstPaivakirja.add(s.toString());
    }

    //Jos on tulostettu suorituksia, niitä voidaan muokata, poistaa ja hakea
    if (lstPaivakirja.getItemCount() != 0) {
        btnMuokkaa.setEnabled(true);
        btnPoista.setEnabled(true);
        btnHae.setEnabled(true);
        cmbHaku.setEnabled(true);
    }
}
```

Kuva 4. btnTulostaActionPerformed-metodin toiminta

Poista-nappia painettaessa valittu suoritus poistetaan käyttöliittymän List-komponentista ja Tietokanta-luokan ArrayList:stä. Tällöin valittu suoritus poistuu siis kokonaan. Kun käyttöliittymän listassa ei ole suorituksia, estetään nappien Poista, Muokkaa ja Hae käyttö.

```
private void btnPoistaActionPerformed(java.awt.event.ActionEvent evt) {

//Jos suorituksia on tulostettu, voidaan poistaa valittu suoritus listasta ja tulostetuista suorituksista
    try {
        if (lstPaivakirja.getItemCount() != 0) {
            int valittu = lstPaivakirja.getSelectedIndex();
            lstPaivakirja.remove(valittu);
            t.poistaSuoritus(valittu);

            //Jos suorituksia ei ole, ei voida poistaa, muokata tai hakea suorituksia
            if (lstPaivakirja.getItemCount() == 0) {
                btnPoista.setEnabled(false);
                btnMuokkaa.setEnabled(false);
                btnHae.setEnabled(false);
                cmbHaku.setEnabled(false);
            }
        }
```

Kuva 5. Painikkeiden toiminta

Suorituksen tallentamisen kannalta oleellisena muuttujana on boolean ’muokattu’, jonka avulla tarkkaillaan tallentaako käyttäjä uuden suorituksen vai muokatun suorituksen. Jos käyttäjä muokkaa tallennettua suoritusta, poistetaan ArrayList:ssa oleva valittu suoritus ja laitetaan sen paikalle muokattu suoritus. Tällöin ohjelma ei tallenna muokattua tietoa uutena suorituksena ArrayList:n loppuun.

```
t.poistaSuoritus(valittu); //Poistetaan valittu suoritus listasta
t2.lisaaSuoritus(k, a, l, p);
t.suoritukset.add(i, t2.suoritukset.get(0)); //Listan Valittuun paikkaan lisätään muokattu
muokattu = false;
```

Kuva 6. Suorituksen muokkaus

3.2. Käyttöliittymä (GUI)

Käyttäjä syöttää omat tietonsa, paikka- ja aikatiedot JTextField-komponentteihin sekä valitsee JComboBox-komponenttiin määritellyistä vaihtoehdoista lajin. Jos käyttäjä ei täytä kaikkia kohtia tai tietojen syöttö on tapahtunut väärin, käyttöliittymä ilmoittaa virheestä JOptionPane-komponentin avulla. Käyttäjä ei voi tulostaa suorituksia, jos hän ei ole ensin tallentanut suoritusta Tallenna-napilla. Napin painalluksen jälkeen tietojensyöttökentät tyhjenevät ja voidaan syöttää uuden suorituksen tiedot.

Suoritusten tulostus tapahtuu List-komponenttiin. Kun suoritukset ovat tulostettu listalle, JButton-komponentit, Muokkaa, Poista ja Hae ovat käytettävissä. Käyttäjä voi valita listasta klikkaamalla suoritusta, jonka haluaa poistaa tai mitä halutaan muokata. Jos käyttäjä muokkaa suoritusta, suorituksen tiedot siirtyvät JTextField-komponentteihin, tietojen muokkauksen helpottamiseksi. Käyttäjän tulee painaa ensin Tallenna ja sitten Tulosta-nappia, jotta listassa näkyy päivitetyt tiedot. Jos käyttäjä ei valitse poistettavaa tai muokattavaa suoritusta, käyttöliittymä antaa virheilmoituksen JOptionPane-komponentin avulla.

Käyttäjä voi myös hakea listasta suorituksia valitsemalla JComboBox-komponentista lajin nimen ja painamalla Hae-nappia. Jos käyttäjä on hakenut suorituksia lajin nimellä, tulostettuja suorituksia ei voida poistaa tai muokata. Tällöin tulee painaa uudestaan Tulosta-nappia, jotta voidaan tarkastella kaikkia suorituksia, joita voidaan poistaa tai muokata.

Kun listassa ei ole yhtään suoritusta, JButton-komponentit Tulosta, Muokkaa, Poista ja Hae eivät ole käytettävissä. Käyttäjän tulee siis syöttää ja tallentaa uusi suoritus, jotta nämä painikkeet ovat taas käytössä.

Kuva 7. Graafinen käyttöliittymä

4. Johtopäätökset ja jatkokehitysmahdollisuudet

Harjoitustyön tavoitteena oli luoda oliopohjainen ohjelmisto vapaavalintaisella sisällöllä ja käyttötarkoituksella. Loimme urheilijan päiväkirjan, jonka avulla käyttäjä pystyy pitämään kirjaa urheilusuorituksistaan. Ohjelmisto sisältää ArrayList-tietokantarakenteen, johon suoritukset tallentuvat, interface-rajapinnan, viisi olioluokkaa sekä graafisen käyttöliittymän. Seuraavassa vaiheessa ohjelmistoa voisi kehittää eteenpäin lisäämällä kalenterin suoritusten lisäämistä ja tarkastelua helpottamaan, laajentaa lajikohtaista analyysia ja tarkastelua tai lisätä ohjelmistoon komponentin, joka tuottaa erilaisia koosteraportteja syötetyistä suorituksista. Myös graafista käyttöliittymää voisi kehittää eteenpäin muotoilemalla rakennetta ja tyyliä lisäämällä esimerkiksi värejä ja sivurakenteita.