Ohjelmointi 3: harjoitustyön dokumentaatio, ryhmä 3652

Tekijät: Patrik Salmensaari, Sofia Rautakoski & Tommi Paavola

Sisällys

1. Yleistä

Harjoitustyön tekeminen aloitettiin 28.3. aloitustapaamisessa. Tapaamisessa keskusteltiin yhdessä työn tehtävänannosta, arvosanatavoitteesta sekä alustavasta työnjaosta.

Aloitustapaamisen jälkeen ryhmä tapasi vähintään viikoittain, useimmiten kampuksella. Tapaamisissa käytiin läpi, mitä oli tapahtunut sitten viime tapaamisen, keskusteltiin uusista toiminnallisuuksista, ratkottiin yhdessä ongelmia, ja tehtiin suunnitelmaa seuraavalle viikolle. Takarajan lähestyessä tapaamisia pidettiin hieman useammin, ja niissä keskityttiin enemmän ohjelman toiminnallisuuksien testaamiseen.

Ryhmän arvosanatavoitteeksi asetettiin aloitustapaamisessa 3 tai parempi. Tässä tavoitteessa on pyritty pysymään pitkin projektia. Ohjelmassa olevista ominaisuuksista arvosanatavoitteen näkökulmasta kerrotaan tarkemmin luvun 4 alaluvussa 4.1.

2. Ohjelman rakenne

Ohjelman rakenne on kuvattu UML-luokkakaaviossa, joka löytyy gitistä nimellä rakenne.png. Tämä siksi, että pdf-dokumentissa ei saa pientä printtiä näkymään tarpeeksi hyvin.

Ohjelman pääluokka on WeatherApp. Kukin Controller-luokka (Primary, Secondary, History, Favorites) vastaa yhdestä ohjelman näkymästä. PrimaryController huolehtii säädatan näyttämisestä. WeatherDataMap ja WeatherData ovat tietorakenneluokkia, johon tallennetaan käyttöä varten säätietoja. iAPI sekä OpenWeatherMapAPI huolehtivat API-kutsuista, ja niihin liittyen on luotu GSonin käyttöä varten muutama pieni luokkatiedosto. Controller-luokat käyttävät luokkaa iReadAndWriteToFile json-tiedostojen lukuun ja kirjoittamiseen. WeatherApp-luokan yhteys kyseiseen luokkaan johtuu siitä, että ohjelman käynnistyessä haetaan tiedostosta viimeisen avoinna ollut näkymä. Tarkempi ohjelman toiminnan kuvaus selviää luvusta 4.

3. Työnjako

Työnjako on projektin aikana elänyt tilanteen mukaan. Aloitustapaamisessa päädyttiin siihen, että toiminnan aloittamiseksi on tärkeintä saada kuntoon API-kutsujen käyttö, joten tähän allokoitiin heti alkuun sekä Patrikin että Tommin aika. Sofia sen sijaan aloitti käyttöliittymäpuoleen tutustumisen.

Kun API-puoli saatiin jotakuinkin toimimaan, niin työvoimaa allokoitiin uudestaan. Patrik alkoi keskittymään ns. backend-puoleen eli tarkemmin siihen, että API-kutsuista saatu data saatiin sellaiseen muotoon, että sitä pystyi hakemaan kaupungin nimen ja ajan perusteella. Tommi sen sijaan alkoi selvittämään backendin eli Patrikin hoitaman puolen ja käyttöliittymän eli Sofian hoitaman puolen yhdistämistä toisiinsa.

Projektin loppuvaiheessa yhteenliittymä oli jotakuinkin olemassa. Tässä kohtaa Sofia jatkoi graafisen käyttöliittymän viimeistelyä, Patrik keskittyi historian ja suosikkien tallentamiseen json-tiedostoon, ja Tommi alkoi työstämään dokumentaatiota sekä hiomaan lisäominaisuuksia. Luonnollisesti kukin oli myös vastuussa oman koodinsa siivoamisesta, kommentoinnista ja parantelusta.

Pitkin projektia osapuolet tukivat toisiaan tarpeen mukaan, lähinnä keskustelemalla ryhmän TG-keskusteluryhmässä ja viikoittaisissa tapaamisissa. Tapaamisissa myös keskusteltiin ohjelman rakenteesta yleisesti, ja tehtiin linjauksia eri toiminnallisuuksista.

Taulukkoon 1 on karkeasti eritelty projektien tärkeinten osien parissa työskennelleet. Nimenomaan karkeasti, sillä useassa tapauksessa useampi henkilö työskenteli saman luokan parissa.

Taulukko 1: Tärkeimpien luokkatiedoston karkea vastuujako

Kokonaisuus	**Päävastuu**
WeatherApp	Sofia
PrimaryController	Tommi
Muut Controllerit kuin Primary + FXML-puoli	Sofia
WeatherDataMap & WeatherData	Patrik
OpenWeatherMapAPI	Patrik & Tommi
iReadAndWriteToFile	Patrik
Dokumentaatio	Tommi

4. Ohjelman toiminta

Ohjelman käynnistyessä pääluokka WeatherApp kutsuu luokkaa PrimaryController, josta alustetaan ohjelman säänäkymä. Ensimmäistä kertaa ohjelmaa avatessa alustetaan tyhjä kaupunkinäkymä, sillä hakuhistoriaa ei vielä ole. Jos ohjelmaa on käytetty laitteella aiemmin, niin käynnistyessä esille tulee viimeisen näytetyn paikkakunnan säätiedot.

Paikkakuntia, joiden säätiedot haluaa nähdä, pääsee hakemaan ”search city” -napin kautta. Tämä nappi aktivoi luokan SecondaryController. Ohjelma vertaa hakua vaihtoehtoihin hyödyntäen luokan iAPI metodia lookUpLocationCoordinates. Metodi hyödyntää tietovarastona tiedostoa municiPalityCoordinates.json, josta löytyy kaikki suomen nykyiset ja entiset kunnat ja kaupungit koordinaatteineen.

Palatessa haun jälkeen pääikkunaan on PrimaryController alustanut näkymän uudestaan löytyneen paikkakunnan tiedoilla. Sään nykytietojen ja ennusteen haku tapahtuu siten, että paluunapin painallus aktivoi PrimaryControllerin luokat setForecast ja setCurrentWeather. Nämä metodit kutsuvat luokan iAPI kautta OpenWeatherin APIa, ja tallentavat paluutiedot json-tiedostoihin. Tämän lisäksi

samalla kertaa tiedot tallennetaan tietorakenteeseen hyödyntäen tietorakenneluokkia WeatherData ja WeatherDataMap.

Kun tiedot on tallennettu tiedostoihin ja tietorakenteeseen, niin ne haetaan vielä käyttöliittymään. Tämä toimii siten, että kullakin käyttöliittymässä olevalla Label-muotoisella oliolla on id, joka kuvaa sitä, mitä labelin pitäisi näyttää (esim. windDir_current). Labelit on edelleen talletettu ArrayList-varastoon, jotta niitä voidaan käsitellä for-loopissa. PrimaryControllerin for-loopit käyvät läpi kunkin Labelin, ja hakevat oikean tiedon WeatherDataMap-luokan metodeja hyödyntäen. Klikatessa ikkunan keskivaiheella olevia päiväkoosteita haetaan tietorakenteesta toisen päivän tuntikohtaiset tiedot samalle paikkakunnalle.

Jos hakutoimintoa käyttää uudestaan, niin ylemmän kolmen kappaleen mukainen tehtävälista tapahtuu uudelleen.

Lisäksi palatessa onnistuneen haun jälkeen pääikkunaan ohjelma tallettaa näkyvissä olevan paikkakunnan nimen erilliseen tiedostoon. Näitä tallennettuja näkymiä pääsee katsomaan historiasivulta, jossa luokka HistoryController hakee historiatiedot json-tiedostosta.

Vastaavasti toimiva toiminto on suosikit-välilehti. Suosikkeihin lisätään vain ne paikkakunnat, jotka on hakuikkunassa päätetty sinne lisätä. Hakuikkuna, historiaikkuna sekä suosikki-ikkuna hyödyntävät kaikki luokkaa iReadAndWriteToFile tiedostojen lukuun ja päivittämiseen. Valitessa historia- tai suosikkipuolelta näytettävän paikkakunnan tallentuu myös se paikkakunta historiaan.

Ohjelman sulkeutuessa tiedostoista siivotaan pois kaikki iAPI-luokan tuottamat json-tiedostot.

Tarkempi luokkien kuvaus ja luokkien eri metodien toiminta selviää projektissa mukana olevista Javadoc-tiedostoista.

4.1 Minimitoteutuksen päälle lisätyt toiminnallisuudet

Kuten luvussa 1 todettiin, on tehtävässä pyritty mukailemaan perustoteutuksen kriteereitä. Perustoteutuksen kriteerit ovat listattuna alla, ja jokaisen kohdalla eritelty toteutuksen tila ja mahdolliset lisätiedot.

- Omatoimisesti toteutettu graafinen käyttöliittymä.
    - **Toteutettu.** Käyttöliittymässä käytetiin inspiraation lähteenä tehtävänannon esimerkkikuvaa, mutta sitä muokattiin omien mieltymyksien mukaan.
- Voidaan tallentaa paikkakuntia suosikeiksi.
    - **Toteutettu**. Tälle on oma näkymänsä. Suosikit myös pysyvät tallella, vaikka ohjelman sulkee välissä.
- Ohjelman tila (nykyinen paikkakunta ja suosikit) tallennetaan levylle ohjelma sulkeutuessa.
    - **Toteutettu**. Säästetään myös hakuhistoria.
- Yksityiskohtaisempi ennuste.
    - **Toteutettu**. Päiväkohtaisessa koonnissa alin ja ylin lämpötila sekä koko vuorokauden sademäärä. Tuntiennusteessa tunnin keskilämpötila, tuulen suunta ja nopeus, sademäärä ja sään kuvaus ikonimuodossa.
- Ohjelma käyttää omia ikoneita.
    - **Toteuttamatta**.
- Ohjelma selviää hallitusti tiedostojen käsittelyssä tapahtuvista virheistä.
    - **Toteutettu**. Kaikissa tiedostoja lukevissa luokissa on toteutettu tarkastelu, joka haetun arvon puuttuessa palauttaa joko tyhjän(null/NaN) tai sitten epäloogisen arvon (esim.

negatiivinen tuulennopeus). Näiden paluuarvojen kohdalle on suunniteltu toiminta, joka jättää hakua vastaavan näkymän tyhjäksi.

- Ohjelmalle on toteutettu yksikkötestejä.
    - **Toteutettu**. Yksikkötesteillä on testattu OpenWeatherMapAPI-luokan toimintaa, WeatherDataMap-luokan toimintaa sekä iReadAndWriteToFile-luokan toimintaa. Testiluokat löytyvät gitistä kansiosta src/test.

Perustoteutuksen kriteeristöstä puuttuu siis yksi toiminnallisuus. Näiden lisäksi on toteutettu seuraavat lisäominaisuudet:

- Paikkakuntien hakuhistoria
    - Tämä on toteutettu hieman ehdotetusta muunnettuna. Hakuhistorian sijasta tallennetaan ns. sivuhistoria. Tämä tarkoittaa sitä, että aina kun näytetään tietyn paikkakunnan sää, niin tämä paikkakunta tallennetaan historiaan. Toiminnallisuus koettiin ryhmän mielestä järkevämmäksi, sillä suurella todennäköisyydellä käyttäjää kiinnostaa juuri ne paikkakunnat, joiden säätä hän on aiemmin tutkaillut. Historialle on niin ikään oma välilehtensä, josta voi palata vanhoihin paikkakuntiin. Historiaan tallentuu 20 viimeisintä näytettyä paikkakuntaa, ja se säilyy myös json-muotoisena, vaikka ohjelman sulkisi välissä.
- Yksikköjen muuntaminen välillä Imperial – Metric
    - Tämä onnistuu painamalla napista ”*Change units*”. Ohjelma muuntaa lämpötilaa yksiköiden Celsius ja Fahrenheit välillä, tuulennopeutta yksiköiden m/s ja mph välillä sekä sademäärää yksiköiden mm ja tuuma välillä.
- **Tiedostojen poistaminen**
    - Ohjelmaa käyttiessä jokaisen paikkakuntahaun kohdalla generoituu uudet json-tiedostot kertomaan säätilaa ja -ennustetta. Ohjelmaan on implementoitu toiminto, jossa ohjelman sulkuhetkellä poistetaan nämä syntyneet tiedostot, jotta tiedostorakenne pysyy puhtaana.

Ryhmän toiveena on, että toteutetut lisäominaisuudet kompensoisivat yhtä puuttuvaa perustoteutuksen kriteeriä arvosanaa määritettäessä.

5. Käyttöohje

Avatessa ohjelman ensimmäistä kertaa ohjelma avautuu tilaan, jossa paikkakuntaa ja sen säätietoja ei ole. Voit hakea paikkakunnan säätietoja hakuvälilehdeltä. Haku tapahtuu kirjoittamalla hakukenttään paikkakunnan nimen ja painamalla hakunappia. Jos ohjelma ilmoittaa paikkakunnan löytyneen, voit napilla ”*see the weather*” nähdä kyseisen paikkakunnan sään. Painamalla ennen säätietojen näyttämistä kohdasta ”*add to favorites*” voit myös lisätä paikkakunnan suosikkeihin. Jos et halua nähdä uuden paikkakunnan säätä vaan palata edeltävään näyttöön, voit painaa ”*see the weather*” -napin sijasta nappia ”*go back*”.

Säätiedoissa näet ylhäällä nykytilanteen, sekä keskivaiheilla päiväkohtaiset koosteet. Klikkaamalla eri päiviä voit tutkia kyseisen päivän tuntikohtaista ennustetta. Halutessasi voit vaihtaa yksikkö SI-järjestelmän ja Imperial-järjestelmän välillä napilla ”*Change units”*.

Sivulla *favorites* voit katsella suosikkipaikkakuntia. Niitä voi olla maksimissaan kahdeksan kappaletta. Jos yrität lisätä täyteen suosikkikirjoon uusia paikkakuntia, se ei onnistu, vaan joudut ensin poistamaan vanhoja suosikkeja klikkaamalla paikkakuntaa hiiren oikealla näppäilemällä ja valitsemalla ”delete”. Klikkaamalla jotakin suosikkipaikkakuntaa pääset tutkimaan sen sääennustetta.

History-välilehti toimii likimain vastaavasti kuin *favorites*-välilehti, mutta siitä et voi poistaa paikkakuntia. Vastaavasti voit klikata mitä tahansa paikkakuntaa välilehdellä nähdäksesi sen sään. Historian tapauksessa talletettava maksimimäärä on 20 paikkakuntaa.

Jos nyt ohjelman käyttämisen jälkeen päätät sammuttaa sen ja palata ohjelman pariin myöhemmin, niin sekä historia että suosikit ovat tallella viime istunnolta. Lisäksi viimeksi näyttämäsi paikkakunnan sään aukeaa ohjelmaan suoraan näkyviin.

6. Tiedossa olevat ongelmat, puutteet ja bugit

- Ohjelman yhtenä toiminnallisuutena on api-kutsujen pohjalta luotujen json-säätietotiedostojen poistaminen istunnon loppuessa. Tämän toteuttaa WeatherApp-luokan metodi deleteJsonWeatherDataFiles(). Olemme huomanneet, että silloin tällöin metodi ei onnistu poistamaan jokaista tiedostoa.
- Paikkakuntien sijaintien selvityksessä on hyödynnetty tietokantaa, jossa on sekä nykyisiä että vanhoja paikkakuntia. Koska näiden kanssa voi olla pientä päällekkäisyyttä (esim. entinen paikkakunta Toijala on nykyisen paikkakunnan Akaa keskusta), niin ohjelmassa on pieni riski sille, että hakiessa yhden paikkakunnan säätietoja tiedot tallentuvat tiedostoon, jossa on toisen paikkakunnan nimi. Tämä edelleen johtaa siihen, että tietorakenteeseen tallentaminen ei onnistu toivotusti, ja tietoja ei löydy, kun käyttäjä haluaa katsoa säätietoja.
    - Tätä on pyritty estämään sillä, että koordinaattien pitää olla hyvinkin tarkasti samat, että haku täsmää, jolloin tiedoston nimeämisen *pitäisi* onnistua suunnitellusti. Ryhmällä ei kuitenkaan ollut aikaa käydä läpi kaikkia useaa sataa paikkakuntaa, joten bugin esiintymisen estymisestä ei olla satavarmoja.
- Toinen huomio paikkakuntatietovarastosta, että ohjelmalla ei voi hakea muiden maailman valtioiden säätietoja. Tämän lisääminen vaatisi paikkakuntatietovaraston laajentamista.
- Kuten luvussa 4.1 jo todettiin, niin perustoteutuksesta puuttuu omien sääsymbolien käyttö, sillä ohjelmassa on käytetty OpenWeatherin tarjoamia symboleja sellaisenaan.
- Koska kuluvan päivän sääennustetta hakiessa saadaan vain tulevien tuntien sää, niin hakuajankohdasta riippuen kuluvan päivän tuntiennuste-Labelit ovat tyhjiä.
- Tällä hetkellä ohjelman auetessa vakioyksikköjärjestelmä on Metric. Tällöin jos sulkee ohjelman sen ollessa Imperialin puolella, niin uudelleenkäynnistyksessä on palattu SI-järjestelmään.

7. Tekoälyn käyttö työssä

Tekoälyä on hyödynnetty työssä jonkin verran, ja voidaankin sanoa, että sitä on käytetty samalla tavalla jokaisessa luokassa.

Itse koodi on luotu ryhmäläisten toimesta, mutta tekoälyä on käytetty apuna bugien etsimisessä ja korjaamisessa, sekä koodin refaktoroinnissa*.

Käytetty tekoälysovellus on C-GPT 3.5.

**Kun koodin logiikka on saatu toimimaan, on tekoälyltä saatettu kysyä kysymys mallia ”miten tekisit tämän paremmin/loogisemmin/tehokkaammin?”. Tämä on koettu tehokkaana korvikkeena sille, että foorumeilta ja dokumentaatiosta etsii tietoa paremmista koodauskäytännöistä.*