Tampereen yliopisto

PROJEKTI: SÄÄSOVELLUS

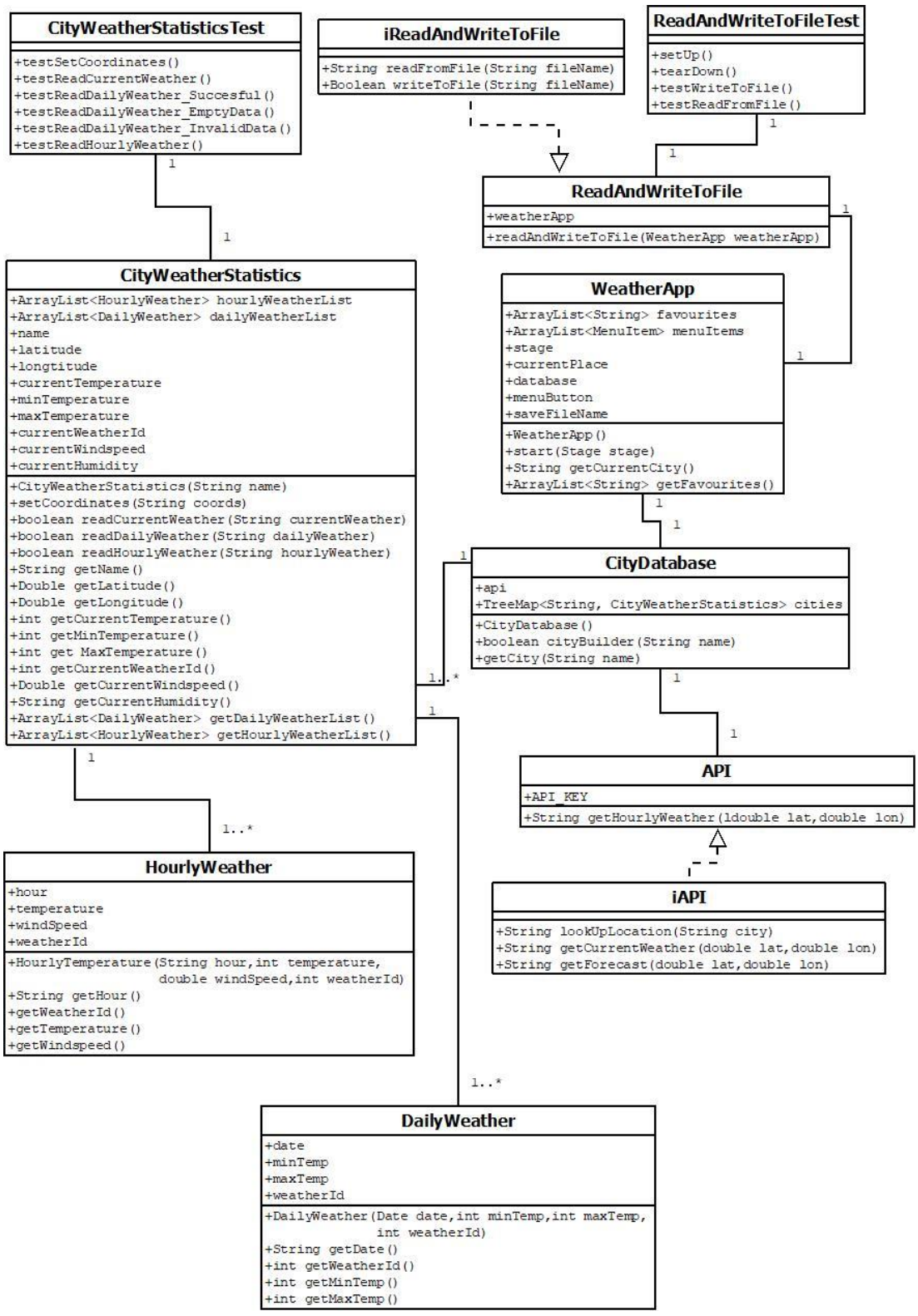
COMP.CS.140 Ohjelmointi 3: Rajapinnat ja tekniikat

Tekijät:
Jonna Hartikka
Aada Härmä
Jaakko Sysimetsä

SISÄLLYSLUETTELO

.

1. UML-LUOKKAKAAVIO JA VASTUUJAKO



Yllä oleva UML-luokkakaavio kuvaa kokonaisuudessaan Sääsovelluksen luokkajaon, luokkien sisäiset attribuutit, julkiset metodit ja luokkien väliset suhteet. Käyttöliittymän ja toiminnallisuuden toteuttaminen on pidetty ohjelmassa erillään. WeatherApp –luokka on vastuussa käyttöliittymän toteuttamisesta. ReadAndWriteToFile –luokka toteuttaa rajapinnan iReadAndWriteToFile ja on vastuussa tiedostosta lukemisesta ja tiedostoon tallentamisesta. ReadAndWriteToFileTest –luokka on vastuussa ReadAndWriteToFile –luokan testauksesta. CityDatabase –luokka on vastuussa CityWeatherStatistic olioiden luomisesta ja varastoimisesta. API-luokka toteuttaa rajapinnan iAPI ja on vastuussa säätietojen hakemisesta OpenWeatherMap rajapinnan kautta. CityWeatherStatistics on luokka kaupungin säätiedoista, jossa parsitaan ja tallennetaan API –luokasta saadut säätiedot. CityWeatherStatistics -luokan attribuutteihin tallennetaan nykyinen säätila omina attribuutteinaan. HourlyWeather –luokka kuvaa yhden tunnin säätiedot yhdelle kaupungille. DailyWeather –luokka kuvaa tulevan päivän ennustetta yhdelle kaupungille. HourlyWeather- ja DailyWeather -oliot tallennetaan CityWeatherStatistics –luokkaan listana ja luokassa on metodit kaikkien säätietojen hakemiseen. CityWeatherStatisticsTest –luokka on vastuussa CityWeatherStatistics –luokan testaamisesta.

2. OHJELMAN TOIMINTA

2.1 Ohjelman toiminta

Ohjelma on sääsovellus, josta voi hakea säätietoja eri kaupungeille ja maille. Ohjelma avautuu ensimmäisen kerran tervetuloikkunaan, ja seuraavalla käyttökerralla viimeisimmän paikan säätiedot avautuvat suoraan. Säätietoja haetaan OpenWeatherMap –palvelun API:sta. Mikäli haku ei onnistu, ohjelma näyttää käyttäjälle ponnahdusikkunan, jossa virhe kuvataan. Säätiedoista näytetään nykyinen lämpötila sekä säätä kuvaava kuvake. Lisäksi näytetään tuulen nopeus, kosteus sekä minimi- ja maksimilämpötila. Lisäksi näytetään seuraavan kymmenen tunnin ennuste, jossa näkyy kuvake, lämpötila sekä tuulen nopeus. Ohjelma näyttää myös neljän seuraavan päivän kuvakkeet sekä minimi- ja maksimilämpötilat. Paikkoja pystyy lisäämään suosikeiksi, jolloin ne saa pudotusvalikosta helposti uudelleen. Ohjelman suljettaessa quit-napista, viimeisin paikka ja suosikit tallentuvat levylle.

2.2 Ohjelman ominaisuudet

Ohjelma toteuttaa perustoteutuksen kaikki vaatimukset.

Lisäominaisuudet:

1) Yksikkötestit on toteutettu myös etätietovaraston putkeen (CI/CD).

2) Oma lisäominaisuus: Päivän ennusteen ja tuntikohtaisen ennusteen lisäksi käyttäjä saa seuraavien neljän päivän ennusteet, joissa on kuvake, sekä minimi- ja maksimilämpötilat.

3) Oma lisäominaisuus: Hakukentän suorittamisessa toimii myös enterin painaminen.

4) Oma lisäominaisuus: Haku antaa error-ponnahdusikkunan käyttäjälle, mikäli haettua paikkakuntaa ei löydy.

3. SOVITTU JA TOTEUTUNUT TYÖNJAKO

Projektin aloitustapaamisessa sovittiin karkeaksi työnjaoksi seuraavaa: Jonna tekee API-rajapinnan käyttöön tarvittavan ohjelmoinnin, Aada tekee käyttöliittymään tarvittavan ohjelmoinnin ja Jaakko JSON-tiedoston lukemiseen ja kirjoittamiseen liittyvän ohjelmoinnin. Tässä vaiheessa ajatuksena oli, että pärjättäisiin näillä luokilla, mutta työn edetessä kävi ilmi, että luokkia tarvittaisiin lisää. Luokkia lisättiin lopulta seuraavasti: CityDatabase (Aada/Jonna), CityWeatherStatistics (Jonna), HourlyWeather (Jonna) ja DailyWeather (Jonna). Lisäksi projektin edetessä sovittiin, että Jaakko toteuttaa testiluokat ReadAndWriteToFileTest ja CityWeatherStatisticsTest. Jaakon vastuulle sovittiin myös testien integrointi GitLabin CI/CD-testiputkeen.

Myös raportin kirjoittamiseen sovittiin vastuualueet. Jonna piirsi UML-luokkakaavion ja kirjoitti luokkien vastuujaosta, Aada kirjoitti ohjelman toiminnasta ja Jaakko kirjoitti projektin työnjako- ja käyttöohjeosuuden. Lisäksi yhteisiä raportointialueita tuli kullekin tiedossa olevista puutteista, tekoälyn käytöstä sekä lisäominaisuuksista.

4. LYHYT KÄYTTÖOHJE

Ohjelman käynnistyessä käyttäjän tulee ensimmäisenä kirjoittaa hakukenttään haluamansa paikkakunnan nimi hakeakseen kyseisen paikkakunnan säätiedot painamalla Search-painiketta. Ohjelman käyttöliittymä näyttää isoimmalla fontilla paikkakunnan tämänhetkiset säätiedot. Tämän lisäksi käyttöliittymästä näkee tulevien vuorokausien päivittäiset huippukohdat (daily highlights, minimi- ja maksimilämpötilat) sekä lähituntien säätiedot.

Kulloinkin haettuna oleva paikkakunta voidaan lisätä suosikkilistalle painamalla Add to favourites -painiketta. Suosikkilistauksen etuna on, että paikkakuntia ei tarvitse erikseen kirjoittaen hakea. Ohjelman saa suljettua painamalla Quit-painiketta. Käynnistäessä sovelluksen uudestaan, edellinen haettu paikkakunta haetaan muistista ja näytetään automaattisesti ensimmäisenä käyttöliittymässä. Lisäksi aiemmin tallennetut suosikit löytyvät edelleen suosikeista ohjelman käynnistyessä uudelleen.

5. TIEDOSSA OLEVAT PUUTTEET

Tiedossa on seuraavia puutteita:

1) Suosikkilistalta ei pysty poistamaan paikkakuntia (ei tosin ollut vaatimuksissa).
2) Yksikkötestit eivät mene läpi yhdellä ryhmän jäsenellä. Testit toimivat kahdella jäsenellä sekä GitLabin CI/CD-putkessa. Ainoaksi eroksi pääteltiin käyttöjärjestelmä, sillä testit menevät läpi Linux-jäsenillä ja Windows-jäsenellä ei. Toiseksi vaihtoehdoksi selvisi googlettelulla, että Maven- ja Java-versiot olisivat poikkeavia.

6. TEKOÄLYN KÄYTTÖ

Jonnan kooste tekoälyn käytöstä:

Luokissa DailyWeather, HourlyWeather, API ja CityWeatherStatistics on käytetty tekoälyä (ChatGPT, 3.5) apuna javadoc kommenttien luomisessa, sillä tekoälyn käyttö nopeutti huomattavasti kommenttien tekemistä. Lisäksi näitä luokkia on annettu tekoälyn debugattavaksi ja kysytty, olisiko luokissa vielä jotakin parannettavaa. DailyWeather –luokassa tekoälyn apua on käytetty siinä, kuinka java.util.Date luokan Date oliosta saa parsittua pelkän viikonpäivän, kuukauden ja päivän. API –luokassa tekoälyltä kysyttiin apua HTTP-pyynnön tekemiseen sekä virhetarkastuksien tekemiseen. CityWeatherStatistics –luokassa tekoälyn apua pyydettiin json datan parsimiseen. Lisäksi tekoälyn apua käytin joissain tilanteissa siinä, että jos ohjelmassa tuli virhe, kysyin tekoälyltä tarkempia tietoja virheen korjaamiseksi. Mielestäni tekoälyn (ChatGPT, 3.5) käyttäminen auttoi paljon projektityön eteenpäin saamisessa. Monet ongelmat ratkesivat nopeasti tekoälyn avulla. Joissain tilanteissa tekoäly ei aina antanut hyvää vastausta, mutta kysymyksiä tarkentamalla ja omaa harkintaa käyttämällä sai hyviä lopputuloksia aikaan.

Jaakon kooste tekoälyn käytöstä:

Koodissa on käytetty ChatGPT 3.5 tekoälytyökalua ReadAndWriteToFile-luokan ja testiluokkien CityWeatherStatisticsTest ja ReadAndWriteToFileTest runkojen sekä em. luokkien javadoc-dokumentointeihin. Tekoälyn käyttö nopeutti huomattavasti työskentelyä harjaantumattomalle koodarille. Epäilen, että ilman tekoälyn käyttöä projektin aikatauludeadlinet olisivat tulleet vastaan. Toisaalta tekoälyn käytöllä on myös käänteinen puoli oppimiselle, kun koodia ei joudu kirjoittamaan rivi riviltä itse. Esimerkiksi java-kielen syntaksin hallitseminen puhtaalta pöydältä ei kehity.

Testiluokan ReadAndWriteToFileTest kehittämisessä oli isoimmat ongelmat. ChatGPT ohjasi käyttämään WeatherApp-luokasta mock-oliota, mutta tämän suhteen oli suuria ongelmia saada oikeanlaiset mock-ohjelmoinnit, käytetyt kirjastot ja riippuvuudet toimimaan yhteen. Tekoälyn kanssa joutui jatkuvasti umpikujaan eikä sen avulla asiaa saanut ratkaistua. Asia ratkesi lopulta konsultoimalla kokenutta koodaria. Edellä mainittujen asioiden lisäksi ChatGPT:tä käytettiin GitLabin CI/CD-yksikkötestiputken käyttöönoton opastuksessa.

Muodostaisin tästä sellaisen johtopäätöksen, että tekoäly on nykyisin hyvä tekemään triviaaleja asioita. Ongelmallisimmissa tilanteissa tulisi kuitenkin oma osaaminen olla riittävällä tasolla, jotta tekoälyä osaa johdatella tarpeeksi tarkasti oikeaan suuntaan.

Aada kooste tekoälyn käytöstä:

WeatherApp-luokassa tekoälyä (ChatGPT, 3.5) on käytetty oppimaan käyttämään MenuButton-komponenttia, jota en ensin saanut toimimaan. Myös CSS-tiedoston konfiguroinnissa on käytetty tekoälyä. Yritin tekoälyn avulla saada muualta ladatut kuvakkeet toimimaan, mutta se ei onnistunut, ja sain ne lopulta toimimaan eri lähteen mukaan. Neljäs asia, johon käytettiin teköälyä, oli rinnakkaisuuden toteuttamiseen, jota tapahtuu ohjelman avautuessa ja sulkiessa. En ollut aikaisemmin tehnyt tällaista, joten tekoälyltä sain rungon taskin luomiseen, ja sen ajamiseen applikaatio-threadin ulkopuolella. Rungon sai sitten muokattua itse sopivaksi omiin tarkoituksiin. Viimeiseksi käytin tekoälyä Gson-syntaksin debuggaamiseen, sillä olin ensin tehnyt sen virheellisesti. Tekoäly on hyvä renki, mutta huono isäntä. Sillä sai hyvin ratkaistua syntaksiin liittyviä virheitä, mutta ohjelman riippuvuudet ja logiikka täytyy ymmärtää itse, että saa tekoälystä eniten irti. Oman kokemuksen mukaan tekoäly on hyvä työkalu, kunhan muistaa arvioida sen tuottamia vastauksia kriittisesti ja muokata niitä itselle sopiviksi.