WeatherApp-projektin dokumentaatio

Ohjelman rakenne:

UML-kaavio omassa pdf:ssä

Tärkeimpien luokkien vastuujako:

WeatherApp on JavaFX:n Application luokasta periytetty ohjelman pääluokka, josta ohjelman suoritus alkaa. Se pitää sisällään sovelluksen tilaan yleisesti liittyviä parametreja, joita ei ole luontevaa laittaa muualle ja joita tarvitaan monissa eri luokissa. WeatherApp-luokkaan tallennetaan sen hetkinen ladattu säädata WeatherState-tyyppisenä oliona. Se käsittelee (lataa ja tallentaa) käyttäjän haluamat yksiköt, hakuhistorian ja suosikkipaikat. Luokka aloittaa UI:n latauksen lataamalla WeatherApp.fxml:n juureksi.

WeatherState-luokka pitää sisällään yksittäisen API-kutsun tarjoaman säädatan sopivissa tietorakenteissa. Sen rakentajalla haetaan tieto API:sta ja tallennetaan se itse olioon. WeatherState:n kautta voidaan hakea sen hetkinen, yksittäisen päivän tai tunnin säädata, jolla se saadaan vietyä käyttöliittymälle näytettäväksi. WeatherState:ssa säädata on tallennettuna yksittäisinä olioina WeatherData-tyyppisinä olioina.

WeatherDataCurrent, **WeatherDataDay** ja **WeatherDataHour** ovat luokkia yksittäisten säätietueiden tallentamiseksi. Tällainen luokkajako on tehty, koska API antaa tämänhetkiselle säälle sekä tunti- ja päiväkohtaisille ennusteille eri parametrit, jolloin vältytään luokilta, johon tallennetaan ylimääräisiä null-arvoisia attribuutteja. Nämä luokat on periytetty luokasta WeatherData, koska parametreillä on paljon yhteistä. WeatherData-tyyppiset oliot luodaan rakentajalla ja niissä ovat vain getterit. Jotkin tietotyypit ovat tallennettu Types-luokassa määritetyillä tietotyypeillä.

Types-luokka määrittelee ohjelmassa käytetyt itse luodut tietotyypit. Sielläkin on tyypit erikseen esim. lämpötilalle päivä- ja tuntikohtaisille ennusteille.

JsonParser on luokka, joka tekee varsinaisen API-kutsun, ja tallentaa säädatan WeatherData-olioina WeatherStaten tietorakenteita vastaaviin tietorakenteisiin, jotka voidaan hakea JsonParser-oliosta gettereillä.

UIController on käyttöliittymän toiminnallisuuden pääasiallisesti toteuttava luokka. Se on asetettu WeatherApp.fxml:n controlleriksi ja se toteuttaa sen vaatimat event handlerit. UIController hakee WeatherApp:n kautta ohjelman tilasta dataa, kuten WeatherState-olion ja hakuhistorian. Käyttöliittymän päivä- ja tuntikohtaisten ennusteiden ”laatikot” ladataan UIControllerin kautta erikseen ohjelmallisesti. Yksittäinen laatikko kummallekin tyypille on määritelty erillisissä FXML-tiedostoissa ja niille on tehty omat alaluokat FXML:n lataamista ja controllerin hakemista varten.

DayBoxController ja **HourBoxController** ovat edellä mainittujen controllereiden määrittelevät luokat. Ne periytetään **WeatherBoxController**-luokasta, koska ne ovat samankaltaisia. Näiden luokkien instanssit ovat ohjaavat siis kunkin FXML:ssä määritellyn laatikon toimintaa. Käytännössä

controllereilla muutetaan vain laatikon näyttämää säädataa. WeatherApp:n sisältämän staten mukaisesti UIControllerin kautta.

Ohjelman toiminta ja vapaaehtoiset lisäominaisuudet:

Ohjelma käynnistyessään aukeaa siihen tilaan, kun se on suljettaessa jätetty. Se hakee muistista viimeisimmän paikan ja näyttää sille säätilan. Ohjelma hakee datan openweathermapin APIsta.

Käyttäjä voi hakea uutta paikkaa joko kirjoittamalla paikan hakuboksiin tai valitsemalla pudotusvalikosta jonkun suosikeista tai viimeisimmistä hauista. Uusi haku toteutetaan, kun käyttäjä painaa hakukentän vieressä olevaa hae nappulaa. Jos paikkaa ei löydy, ohjelma muuttaa hakukentän reunat punaiseksi, merkitäkseen väärää paikan nimeä.

Ohjelma näyttää valitulle paikalle tämän hetkisen sanallisen ja kuvallisen sääselitteen, lämpötilan, koetun lämpotilan, kosteuden, tuulen ja suunnan, sekä paineen. Valitulle paikalle, näytetään myös ennuste tuleville päiville, joka sisältää sanallisen ja kuvallisen sääselitteen, lämpötilan, lämpötilavälin, koetun lämpotilan, kosteuden, sateen todennäköisyyden, sademäärän, tuulen ja suunnan, sekä paineen. Lisäksi näytetään tuntiennuste, jossa on kuvallinen sääselite, lämpötila, koettu lämpötila, tuuli sekä tuuleen suunta, ja sademäärä.

Painamalla jotain päiväennustetta, voidaan siirtyä klikatun päivän tuntiennusteeseen. Tunti ja päivä ennusteita voi myös selata vieritysrullasta.

Hakuhistoria: Ohjelma tallentaa viisi viimeisimmäksi haettua paikkakuntaa hakuhistoriaan. Hakuhistoria näkyy hakupalkin pudotusvalikossa suosikkien kanssa. Jos hakuhistoriaan kuuluva paikkakunta ei kuulu suosikkeihin, paikkakunnan nimen edessä ei ole sydänkuvaketta. Paikkakunnan nimi näkyy pudotusvalikossa vain kerran, vaikka se kuuluisi sekä hakuhistoriaan että suosikkeihin. Hakuhistoria tallentuu tiedostoon, kun ohjelma sulkeutuu ja ladataan tiedostosta ohjelman käynnistyessä.

Tuki useammalle mittayksikköjärjestelmälle: Ohjelman asetuksissa (aukeaa rataskuvakkeella varustetusta napista) voidaan valita pudotusvalikosta haluttu mittayksikköjärjestelmä (metric, imperial, standard). Kun painetaan ”Save and close” -nappia, säätiedot haetaan uudella API-kutsulla uuden mittayksikköjärjestelmän mukaisesti ja päivitetään UI:hin. Ohjelman sulkemishetkellä valittuna ollut mittayksikköjärjestelmä tallentuu tiedostoon ja ladataan tiedostosta ohjelman käynnistyessä uudestaan.

Sääkartat: Ohjelmassa on sääkarttaominaisuus, joka pystyy näyttämään sateen, lämpötilan, pilvet ja tuulen kartalla. Kartan keskikohta on aina ohjelman käynnistyksen tai uuden paikan latauksen jälkeen paikassa itsessään ja se zoomaantuu latauksen yhteydessä siihen kauempaa. Oikeasta yläkulmasta ”+”-merkistä voi vaihtaa kartan näyttämiä säädatakerroksia. Karttaa voi zoomata joko vasemmasta yläkulmasta tai normaaleilla nettiselaimen tukemilla hiiren komennoilla, koska kartta näytetään WebView-elementissä. Vasemmasta alakulmasta voi kartan resetoida alkutilaan myös

tilanteessa, jossa käyttäjä on ajautunut OpenStreetMapin tekijänoikeussivulle. Kartan WebView-elementtiin ladataan ”resources” kansiosta löytyvästä tiedostosta map.html. Tässä yksinkertaisessa nettisivussa käytetään vanhempaa Weather maps 1.0 API:a säädatan hakemiseen, joka saadaan kartalle asettuvien ”laattojen” muodossa sekä OpenLayers API:a, joka tarjoaa rajapinnan, jolla laattojen asettelu onnistuu vaivattomasti sekä kartan käyttöliittymän. OpenLayers:n kautta saadaan myös pohjakartta, joka on OpenStreetMap. Itse kartan toiminnallisuus on toteutettu tiedostossa map.js.

Säädatan ilmaisussa on muutama lisäominaisuus. Tuulen kuvakkeen kääntyminen vastaa säädatasta saatua tuulen suuntaa asteina. Ilmanpaineen kuvake kääntyy kolmeen eri asentoon, jos tietyt raja-arvot ylitetään tai alitetaan. ”Tuntuu kuin” kuvake muuttaa väriä todellisen lämpötilan ja ”tuntuu kuin” lämpötilan erotuksen mukaan. Sininen, jos kylmempi, harmaa, jos sama ja punainen, jos kuumempi.

Luokat, joissa on dokumentoituna esi- ja jälkiehtoja yms:

Controller-luokat, JsonParser ja WeatherState tarvitsevat toimiakseen WeatherApp-olion. Lähes kaikki luokat ovat riippuvaisia joistain Types-luokassa määritellyistä tyypeistä.

Sovittu ja toteutunut työnjako:

Siri Sytelä: Current weather UI, yläpalkin UI, hakupalkin toiminnallisuus, hakunapin toiminnallisuus, suosikit, asetukset, hakuhistoria, UML-kaavio, omien osien dokumentointi ja yksikkötestit. Toteutunut työnjako vastasi sovittua.

Oskari Heinonen: WeatherData -luokat, päivä- sekä tuntikohtaisten sääennusteiden UI ja toiminnallisuus, osittain CurrentWeather, sääkartta, Types -luokka ja näiden yksikkötestit sekä dokumentointi. Toteutunut työnjako vastasi sovittua.

Joel Holappa: Tehtävät: datan haku APIsta, tallennus data luokkiin, ohjelman tilan tallennus ja tilan hakeminen käynnistyessä.
Toteuma: JsonParser, iAPI, StartupState, StartupData, Settings, iReadAndWriteToFile, StartupStateTest, LocationException. Toteuma vastaa työnjakoa.

Käyttöohje:

Ohjelmassa pystyy valitsemaan haettavan paikkakunnan joko kirjoittamalla sen syötekenttään tai valitsemalla pudotusvalikossa olevista vaihtoehdoista (suosikit ja hakuhistoria). Haku suoritetaan painamalla suurennuslasikuvakkeella varustettua nappia. Mikäli haettua paikkaa ei ole olemassa, muuttuu hakukentän reunat punaisiksi. Paikan voi lisätä ja poistaa suosikeista painamalla sydänkuvakkeella varustettua nappia. Rataskuvakkeella varustetusta napista aukeaa asetukset uuteen ikkunaan. Asetuksissa voi muuttaa yksikköjärjestelmää valitsemalla pudotusvalikon vaihtoehdoista. Uudet asetukset tulevat voimaan vain ”Save and close” -nappia painamalla. Neljän vuorokauden tuntikohtaisiin ennustuksiin pystyy siirtymään joko klikkaamalla päiväkohtaista

ennustetta tai skrollaamalla. Karttanäkymää voi zoomata ja loitontaa karttakuvakkeen plus- ja miinuspainikkeilla sekä palauttaa alkutilaan vasemman alakulman painikkeella. Karttanäkymän oikean reunan plussapainikkeesta aukeaa valikko, josta voi valita kartassa näytettävät säätiedot.

Ongelmat ja puutteet

UIControllerista olisi loogista erotella vielä ainakin CurrentBoxController ja MapController, koska tunti- ja päiväkohtaiset on toteutettu myös niin. Tilanteen, jossa ohjelma käynnistetään ilman nettiyhteyttä tai API:sta ei jostain muusta syystä saada haettua dataa, voisi käsitellä paremmin kuin kaatamalla ohjelma kokonaan ja ilmoittamalla siitä komentorivillä.

Tekoälyn käyttö:

ChatGPT:tä käytettiin avuksi hakukentän pudotusvalikon toiminnallisuuksien toteuttamiseen, asetusikkunan luomiseen sekä fxml-tiedostojen muokkaamiseen. Lisäksi chatGPT:tä hyödynnettiin virheiden paikantamiseen ja korjaamiseen. Koodia ei kopioitu suoraan tekoälyltä vaan tekoälyä käytettiin nopeuttamaan tiedon etsintää sekä antamaan neuvoja, kuinka hankala kohta koodissa kannattaisi toteuttaa.

GitHub Copilot:ia on käytetty kommenttien, yksikkötestien, gettereiden ja muun triviaalin ”boilerplaten” generoimiseen, Javan ominaisuuksien ja kirjastojen toiminnan selittämiseen ja niistä sopivien ominaisuuksien löytämiseen sekä järkyttävän pitkien virheilmoitusten tulkitsemiseen.