Group 1

Aaron Hirvi

Miikka Venäläinen

Onni Merilä

Samundra Dhakal

Design document

Changes:

- The whole application have seen some radical changes in its structure and implementation.
- There are major redesigns: Some of the data is saved in a object and those objects are then saved in a data structure which is then used by graph/element making classes/functions.
- Original documentation had some good aspects which were then proven lacklustre in a coding process. These are now replaced with better solutions but some changes may still happen.

What has been implemented:

- Backend side of the project is somewhat ready. Data can be fetched from the APIs and it can used by the program.
- Chart creation process, but usage of the API data is not yet implemented.

What will be implemented:

- Saving possibility of the data and its usage in code.
- Error checking.
- Parameter value checking.
- Using actual API data to create different and accurate Charts
- Combining the two separate UI classes
- Better communication between classes (interfaces etc...)

Components:

- MenuWindow
    - A view for selections

    - In this window user can select the data to be shown

Packages and files in it:

Deeper explanations of each function can be found from the code.

- API
    - APICall
        - Contains GetRequest function which fetches all required data from Digitraffic and FMI API.
        - Class and its method is used everytime user searches for new data.
    - RoadDataProvider
        - This class is used to get road related data.
        - Contains all the functions which are used to get data from Digitraffic API.
        - The application uses its functions to get maintenance data, road condition data and traffic messages. This data is used to make graphs and other elements.
        - Class also contains all necessary URL addresses saved to get data from.
    - WeatherDataProvider
        - This class is used to get weather related data.
        - Contains all the functions which are used to get data from FMI API.
        - The application uses its functions to create necessary URLs for getRequest function based on user parameters, to go through the data and to calculate average and min-max values.
        - Class also contains all necessary URL addresses saved to get data from.
    - MaintenanceTask
        - RoadDataProvider uses this class to make objects out of it for maintenance tasks which are then used to make other elements and visualization for the user.
    - RoadCondition

        - RoadDataProvider uses this class to make objects out of it which contains road condition data. Data is then used to make elements and visualization for the user.
    - TrafficMessage
        - RoadDataProvider uses this class to make objects out of it which contains traffic message data. Data is then used to make elements and visualization for the user.
    - Utility
        - Class which contains some helpful functions.
        - Other classes uses its functions to get trimmed date or time data or to create correct timeformat with date and time.
        - Some other functions might be added.
- Components
- Graph
    - GraphProvider
        - This class is the interface for the ui to get JavaFX UI component.
        - ChartViewer class was decided to be used, as it seemed to fit the wanted requirements of our program. Other classes were tried, but they were not ideal for our use case.
            - Org.jfree.chart.fx.ChartViewer
        - At the moment contains the creation of a hardcoded example chart, that can be tested by UI.
    - GraphBuilder
        - At the moment doesn't contain or do anything yet, but the plan is to separate the Provider class from the actual JFreeChart creation, for simplicities and dependencies sake.
        - Using a Factory design pattern was an idea, but further discuss about this in the TA meeting.
- UIView

Flow of the program:

- Main
    - Starts the program
    - Opens menu view

- MenuWindowController
  - Creates a menu view
  - Calls the appropriate class based on the task selection
- RoadDataScene
  - Needs location and time period information – gets it from the menu view. For the forecast there is a seperate time selection
  - Asks for specification of what kind of data user wants to see and what are the search parameters. These are asked via dropdown menu selections where user can select one option (maintenance or condition forecast) and several options (visibility, friction, precipitation, winter slipperiness, overall road condition)
  - Creates a scene based on the data and parameters and injects it into menu view
  - User can make alterations to the data shown
  - When the graph is drawn, save button appears
- WeatherDataScene
  - Needs location and time period information – gets it from the menu view
  - Asks for specification of what kind of data user wants to see and what are the search parameters. These are asked via dropdown menu selections where user can select several options (temperature ,observed wind, observed cloudiness, predicted wind, predicted temperature)
  - Creates a scene based on the data and parameters and injects it into menu view
  - User can make alterations to the data
  - When the graph is drawn, save button appears
- CombinedDataScene
  - Needs location and time period information – gets it from the menu view
  - Gets weather information from WeatherDataScene and road data from RoadDataScene and combines it
  - Data can be altered after it has been created
  - Save button appears when graph is drawn
- APICall
  - Provides data from digitraffic and FMI to other classes
  - Has a getRequest() function which handles the API calls and returns a JSONObject. Function uses org.json to handle JSON data and other function to convert XML to JSON

Dependencies:

- org.json

    - We decided to use org.json as our JSON and XML parser class, as it had both JSON and XML support.
- JFreeChart
    - Used for drawing beautiful plots used in the program. We decided to use it based on the look of the example images, and the fact that it has existed for a long time and still getting regular updates.
- JFrame
- APIs
    - https://www.digitraffic.fi/en/road-traffic/
    - https://tie.digitraffic.fi/swagger/
    - https://en.ilmatieteenlaitos.fi/open-data-manual
    - Program needs these APIs to get all the needed and wanted data.

Traffic:

- User click on the traffic menu, they will be redirected towards the selection view of the traffic.
- User can select the traffic using either by selecting the point or via selecting the road.

Case One: Selection Of the Road

- When user click on the select a road, the window appear with the input field for providing or selecting the road number.
- Once road number is provided, the data will be fetched using the API in the window which included the message of traffic updates.
- These updates will be displayed in the right side of the screen.
- If the user wants to have others information, there is the drop-down selection in the left side of the screen which includes:
    - Temperature, Observed wind, Observed cloudiness, Predicted wind, Predicted Temperature and so on.
- Once user selected the data from the drop-down, the info will be visible in the graph plot.

Case Two: Selection Of Point

- User choose the select the point, they will view the UI with input fields for providing the coordinates of road as well as the search button to get the info from the server.
- Once user click on the search button, the info message should be visible in the right side of the UI.

- In the left side of the UI, there’s present the drop-down selection from where user can request for the additional information just like in the case one and it should be visible in the UI.