Java course

summer 2024 - Exercise 3

Ramez Mannaa - 209074491- ramez132@gmail.com
Meir Zoref - 305643231 - meirzo@mta.ac.il

System Overview

The app from the 2nd exercise was enhanced to a client-server app.

The client is a JavaFX spreadsheet application that allows the user to manage and manipulate data in a tabular format (the system refers to it as a sheet).

The server contains the system engine, which manages several sheets from different users. All communication of a client is only with the server. The clients do not communicate “directly" with each other.

Users can do the following:

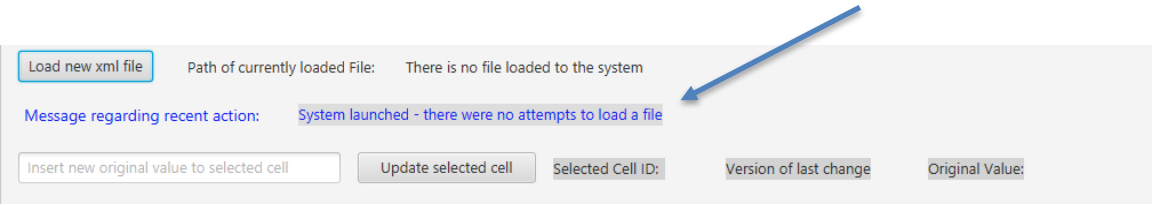
- Upload their own sheets.
- See all the sheets that exist in the system and all access permissions/permission requests that each sheet has.
- Request access to other users' sheets (Access to the sheet can be for reading only or also for editing).
- Approve/deny access requests to their sheets.
- Watch and edit a sheet, depending on the permission level. All capabilities are the same as the 2nd exercise app, with addition of dynamic analysis:
  See the data of a single cell, update cell value, watch previous versions, watch/add/delete a range of cells, filter a range, sort a range.
  The dynamic analysis is the possibility to mark a cell, set min and max values, and a step size, and using a slider to watch the sheet changes dynamically.

If several users have editing capabilities - they can edit the sheet at the same time and see the changes that each of them makes in the system.

<u>Important notes</u>:

1) When moving between the stage that manages all sheets to/from the stage of a single sheet – the window grows/shrinks. If possible, it's recommended to use 2 screens when using 2 clients.
2) After opening the batch file run-batch-file, it takes a few seconds for the app to start. It's recommended to close the CMD window when using the app itself.
3) **When another user changes a sheet while you watch it, it will be indicated by a message and the button "Display recent sheet" will become green, until pressed.**
4) Almost each action or error produce a message that is displayed to the user. The message lets the user know if the operation succeeded or failed, with additional data. If the operation failed, the message informs the user what he/she should do for it to work.

The message is displayed in the second line of the app:



5) The user can press buttons to request permissions only when selecting a sheet he/she doesn't own. The user can press buttons to approve/reject permission request only when selecting a sheet he/she owns.
6) When sorting, filtering, using dynamic analysis or viewing a previous version, all buttons except one are disabled, so the user cannot change data in the sorted/filtered/previous sheet. There are buttons to abort sorting/filtering/dynamic analysis or returning to recent sheet, and when pressed, all buttons are enabled again, and the original recent sheet is displayed.
7) <u>Due to personal problems and delays, we were not able to add these features:</u>

Changing the display of a single cell, changing alignment, row height or column width.

Main classes and modules added since exercise 2

Client Classes

1) Controllers: the client has 3 stages, for login part, managing all sheets and using a single sheet. There is a new controller for the login stage, and 4 new controllers for the management window (main, top, center and right controller). They all communicate throw the main controller.
2) Classes for refreshing data from the server: the client app has dedicated classes that manages extra threads to constantly keep the data in the client updated with changes in the server – there are classes to refresh data of all sheets, all permission requests and status for selected sheet. When using a single sheet, there are threads to get data of newer versions and for changes in the ranges.
3) HttpClientUtil: Class to manage Http requests, using **OkHttp** library.

   DTO module
4) The server, the engine and the client know this module. The module contains all relevant DTO classes. The server and the client uses these classes to transfer data (using **Gson** library).

   New Engine Classes:
5) PermissionRequest class and Enums to manage permissions data.
6) DTO factories classes, to create new DTO instances.
7) SingleSheetManager to manage a single sheet.
8) EngineManagerForServer to manage all SingleSheetManager for each sheet. The server holds an instance of EngineManagerForServerImpl.
9) Server module

   Holds multiple classes that extends HttpServlet to handle http requests. There are servlets for all kinds of actions in the system.

   Uses **Tomcat** library tomcat to manage the communication.

Information of classes from the Readme file of the second exercise

Main classes added since exercise 1

10) Expression: An interface representing an expression in a spreadsheet cell, which can be evaluated in real time to produce a value.
    New classes implement this interface - a class for each new function available in the system: If, Less, Bigger, Equal, And, Or, Percent, Sum, Average. Some of the classes can contain only Boolean inputs (And, Or), some only numeric inputs (Bigger, Average, Percent), and some can contain different input objects (Equal, IF). Sum and Average work with ranges of cells.
    Each class overrides the method 'eval' which evaluates the expression in real-time, depending on the type and value of the actual expression and the actual values and types in cells it might reference.
11) RangeImpl - The RangeImpl class represents a range of cells in a spreadsheet, defined by a start and end coordinate. It includes methods to get the start and end coordinates, check if a coordinate is within the range, and getting all coordinates that belong to the range.
12) RowInArea – class to hold effective value of a specific row, in a certain range (between 2 specific columns).
13) RangeWithRowsInArea – manages a map of all RowInArea in a range. Used for sorting and filtering.
14) RangeSorter – used for sorting a selected range, in a specific order of columns. Uses a recursive algorithm.
15) Controllers to manage the user interface:
    The app has 3 "areas" – top part, left part, sheet part. Each area has its own fxml file and capabilities. Therefore, each part has its own controller. The app also has one main controller that manages everything and the communication between the different parts.

Additional information of classes from the Readme file of the first exercise

Main Classes and interfaces

CellImpl: A class that implements Cell interface and represents a single cell in the sheet. The class manages the cell's original and effective values and holds a data structure of cells it depends on, and a data structure of cells it influences on.

SheetImpl: Represents a spreadsheet, managing its cells, and operations related to cell updates and value retrieval. This class implements Sheet interface, which is an interface that extends 2 interfaces: SheetReadActions, SheetUpdateActions.

CoordinateImpl: A class that implements Coordinate interface and represents a coordinate in a spreadsheet, defined by a row and column.

CoordinateFactory: A class with static methods and a static data structure to handle the creation and retrieving of a coordinate from a string (i.e. B5) or from a row and column numbers. It holds and manages a data structure of all previously created coordinates in the sheet, to avoid creating many CoordinateImpl objects with the same actual coordinate.

Expression: An interface representing an expression in a spreadsheet cell, which can be evaluated in real time to produce a value.
There are several classes implementing this interface - a class for each function available in the system. Each class overrides the method 'eval' which evaluates the expression in real-time, depending on the type and value of the actual expression and the actual values and types in cells it might reference.

SheetFromFilesFactory: Manages the serialization of objects from an XML file using JAXB (Java Architecture for XML Binding).