Personal information

Otsikko; tekijän nimi, opiskelijanumero, koulutusohjelma, vuosikurssi; päiväys.

Jaakko Hintsala
886790
Tietotekniikan kandi
2020
28.4.2021

General description

I have completed the Data Dashboard project on the moderate difficulty. Also as an extra, I have implemented the ability to have multiple-series charts, and I have also added extra information to the Datapoint tooltips.

User interface

When you run the program, you are first greeted by a very simple Gui. You can add new charts/cards/tables by using the menubar on the top.

When you wish to add a chart/card you are greeted with a pop-up. Please select data from a table and an axis name for the both axises and then a name for the legend.

You can update the data in a chart/card by editing the corresponding table cell. You may also reselect the the data of a series.

You can either save a single table/chart/data object to a file by clicking the corresponding item in the objects contextmenu. You can save the dashboard by going File → Save Dashboard on the upper menubar.

You can load cards/charts/dashboards from file by going File → Open.

Algorithms

I will elaborate on two interesting algorithms in my program.

1. How do I get the data from table cells into a chart/card.

DataValueChooser listens to the focusOwnerProperty of the scene and in the case it is a TableView it adds a change listener to its selectionmodel which updates DVChoosers "positions" observablebuffer with its selected cell positions. When the selection for the values of one axis is done, the current values are saved and future changes in "positions" affect the values of the other axis. In the end the selected positions are given as a paremeter for a new DataObject.

2. How do I do maintain the updatability of charts/cards through saving and loading from a file.

(updatability = chart changes when a table cell is edited to a new value)

Each TableView has a unique id created with java.util.UUID.randomUUID. When a TablePosition is saved into a file. the id of its parent tableview is saved with it alongside with a string value of the cell's contents.

When a chart/card is loaded from file, during each one of its serialized Tablepositions conversion process the program checks if a tableview with the same id already exists and initializes the normal TablePostion with it if it does if not the chart will be initialized with string data from the serialized table cell.

Data structures

I use the same basic principle for all the data structures I use. The class that saves the data (GenericTaulu, DataObject, CardDataObject) takes its initial data as parameters and saves it into a Observablebuffer. And then different methods listen to that buffer and take care of updating various elements.

Files

As for files I took advantage of the Serializable trait that case classes have. When the program saves a an object, first turns all the relevant data of the object into a case class and writes the case class into a file. The file is not human readable.

Worst and the Best

The main weakness of my code is that it is badly organized and hard to understand.
In the DataChoosers package I believe I could reduce the amount of code to half with some refractoring.

Some highlights of my program:

- GenericTaulu class, I spent a lot of time (too much) in the beginning of the project to make the tables work nicely. (adding columns was hard)
- The Mechanisms I used to update data, I spent a lot of effort to make sure everything (EVERYTHING) updates.
- file saving, in my opinion the way I used case classes and Extension filters was pretty clever in my opinion