Programming Project Group 1

Project Brief:

Construct an application to explore data from a social news website.

## Front end:

We took the design of our front end to mimic a news agitator sites such as YCombinator HackerNews (the site where our data comes from), Reddit and others. We took a minimal design aesthetic to not distract the user by presenting the information readily and clearly. Our UI can be broken down into three different parts or ‘screens’. The three screens are the home screen, story screen, and the user screen. Each screen has a banner with a button on the logo which takes you back a page. This is implemented using a Stack, where whenever the screen is changed, the current screen is pushed onto a Stack of screens. Whenever the function to go back a Screen is called, the previous Screen is popped off the Stack and set as the current screen.

The home screen displays a preview of each story. This includes the title, points, who submitted it, when they submitted it, and the number of comments. Hovering over this preview will create a drop shadow, giving the user feedback that it is a clickable button. Clicking on the preview will open the story itself. On the home screen there is also a search function which updates the stories as you type your search. A dropdown allows you to select whether to search by Author or Story Title. There is also a sort button which, when clicked, is a drop down that allows you to sort the information in a series of ways. There is also a reset button which resets the database in the backend and reloads from the .json file.

Clicking into a story brings up the full post information and all the comments. The story screen contains the title, author, score, time, amount of comments, and the thumbnail of the website (fetched from the URL). It also displays all comments for the story, with all levels of depth, and scrolling down you can see the other comments. If you hover over the story title or author tags (for both the story and the comments), you will see a blue underline, showing the button to be clickable. Clicking on the title of the article will open the article in your web browser. Clicking on a username will open the user page for that user. All the comments that are in the database will be loaded up. With the smaller data sets, the comments aren’t necessarily accessible and if that is the case they are not shown.

Going to the user page allows you to see all the information on that user. You can see the total number of points that they have, the number of posts, and the number of comments. There are two buttons under this information which switch between showing the comments and showing the stories that the user has posted. You can scroll through them all using the arrow keys in the bottom right corner. Clicking on a story or comment will open the story page with that comment or story.

**Event system:**

The frontend also uses Java 8 lambdas to handle events (such as button click and hover). This allows for much greater code reusability and abstracts event handling from the frontend code.

**Dependency Handling:**

The project also uses Maven to handle dependencies (Processing and the H2 driver library). This allows everyone to have the same version of the libraries automatically downloaded into the IntelliJ project. Because the version of Processing on the Maven Central repository was outdated, we manually installed the newest version into a local repo in the SVN code.

## Back end:

The back of our application takes in a file in a JSON format and returns a set of methods to easily display said data in the front end. It is split into 4 main parts, the file loader, the database manager, the queries and the three main types of data in the JSON (stories, comments, users).

We use an H2 embedded database to story three tables: Stories, Comments and Users. The Users table collects data for all the users from the stories and comments tables and stores them for easier and faster querying.

The database also makes use of carefully-selected indices to speed up the querying of the database.

The file loader:

* + Parses the JSON line by line into an SQL database
  + Ignores wrongly formatted items
  + Ignores deleted items

The database manager:

* Inserts the parsed data into tables using the H2 DBMS for JDBC
* Runs the SQL code that it receives against the tables

The queries:

* Runs basic SQL commands against the tables to sort the data into quickly loadable chunks
* Front end calls these methods to get an array list of desired objects (stories/comments)

Story:

* Loads comments on the story
* Returns the story comment IDs, URL, title, author, score, time and ID

Comment:

* Loads comments on the parent comment
* Returns comment IDs, parent ID, text, time, author and ID

User:

* Loads the story and comments from said user
* Returns the story IDs and comment IDs
* Loads total story points