Final Project: Collaborative Editing
Objective: Programming in the Large
Assigned: 4/7/15
Due: 5/6/15

Manager: Ben Whitely (whitely@email.arizona.edu)

Overview

What-you-see-is-what-you-get, or WYSIWYG (pronounced "wussywig"), programs are editors that allow users to create documents. These editors allow users to create and publish text-based documents in which the text contains metadata: information about the text. This information includes, but is not limited to, text color, font, and weight. As the acronym suggests, text created in a WYSIWYG editor will appear the same every time it is opened in that program. WYSIWYG editors include Microsoft Word and Libre/Open Office.

Real-time collaborative editing programs, such as Google Docs, allow multiple users to simultaneously edit a single document. Changes to the document are tracked and attributed to the user that made them. The same feature set that appears in single-user editing mode appears in multi-user editing mode, although multi-user editing mode includes additional features pertaining to user management.

Your Task

You will be creating your own version of a collaborative online editing program. The program will be created in Java and will use the standard Java GUI facilities we have used in class. The program must have the ability to work over the network. Your program will consist of a server portion and a client portion. The server can handle multiple client connections over the network.

Requirements:

Server

- Facilitates the creation of users
- Allows users to create documents
- Allows multiple users to connect simultaneously over a network connection using ServerSocket objects
- Maintains a list of users
- Allows users to grant editing permissions or ownership to other users

User

- Has a unique user name (String) and unique numerical identifier (id)
- Has a password which is stored only in hashed form
- Has the ability to change a password
- Has the ability to reset a forgotten password by providing their user name (*for the sake of simplicity, we will assume this is all they need to do*)
- Has a list of documents owned by the user
- Has a list of documents editable by the user

Document

- Contains rich text information which can be stored in any format: html, rtf, plain text with separate metadata – the storage format is up to you
- Contains a revision history, where each revision contains the text modified, the time of revision, and the revising user. Sample revisions include:
  - user 'a' adding one line of text at 09:12;00.00
  - user 'b' adding a line of text at 09:12;05.01 (note that this revision depends on the revision of user 'a' – by itself, it will not produce the document that they have produced together)
  - user 'a' formatting some text at a given time
  - user 'a' deleting some text
- The document should, at every moment, have revisions that represent the final product. This means that if a document were to be created from a revision list, it should match exactly the document with that revision list.
- The document must support rich text formatting, and must include these formats:
  - bold text
  - italicized text
  - indented text
  - colored text
  - text in different fonts
  - text in different sizes
  - lists
- The documents must also include more formatting elements as specified by you
- Formatting is not mutually exclusive; multiple formats can be combined. For example:
  - bold, indented text
  - a list of text, where each list element has a different color and font size
  - a sentence where each word has a different font and font size

GUI

- Prompts users to log in
- Allows users to perform functionality listed under *User* (change password, reset password, etc.)
- Displays a list of documents owned or editable by the logged-in user
- Allows for creation of a new documents
- Allows for changes to the permissions or ownership of a document
- Editor
  - Has buttons for each of the supported formats.
    - If a button is pressed while text is highlighted, apply that formatting to the text
    - If a button is pressed while no text is highlighted, apply that formatting to all text entered until the button is pressed again. Make it obvious to the user when this is the case.
  - Has a large text editing area (such as a `JtextPane`)
  - Displays the revision history of the document. **Clicking on a revision will display the document as if that revision was the most recent and no other revisions had been made**. The editors can opt to "rollback" the document to a specified revision.
  - Displays a list of currently connected editors, as well as all editors, and provides a chat panel for them to chat to each other without modifying the document.

Possible Extra Credit

As mentioned, you must include additional formatting options. Extra credit could include more formatting, but you would have to demonstrate that the problem is more novel than re-implementing a required format. Some additional ideas include import of documents to your format from html (*a separate project specification will be given to you specifying the input/output format for html, or you can agree on one with the leader*) or supporting export of documents to html. Remember that all extra credit must be agreed upon between you and the project leader.