SE/ComS 319 Homework 3: Swing

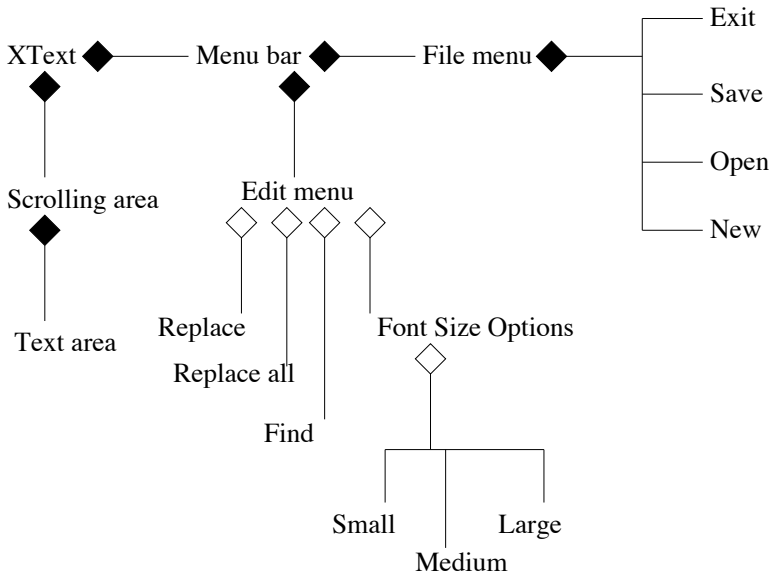
Homework must be individual's original work. Collaborations and discussions of any form with any students or faculty members other than Samik Basu are not allowed. If you have any questions/doubts/concerns, post your questions/doubts/concerns on the blackboard LS and/or contact the TAs or instructor.

The homework is due on October 1, 2012 at 11:59PM. Submission guidelines are as follows:

1. Submit via Blackboard
2. Your top-level Java class must be XText⟨yournetid⟩ (your Java file is, therefore, named XText⟨yournetid⟩.java)
3. You must submit just XText⟨yournetid⟩.java, i.e., all class definitions should be present in one file XText⟨yournetid⟩.java; all classes that you write should be named ⟨someclassname⟩⟨yournetid⟩
4. Your submission XText⟨yournetid⟩.java must compile and execute in Java 6 or 7, without requiring any external libraries or files
5. Write your name at the top of XText⟨yournetid⟩.java and mention the Java version you have used (Java 6 or 7)

Violation of any of the above requirements will result in automatic 10pt deduction from your homework assignment.

Consider a simple text editor where the the static user-interface components are organized as follows (you are not required to assume that these components are classes in your implementation):



In the above, XText represents the editor which contains a scrolling area, which in turn, contains a text area. The editor also contains a menu bar containing a file and an edit menu.

The functionalities available to the end-user are specified as follows:

1. **Use Cases**:
   (a) Text area should be editable. The title of the text editor should be set to `XText:`⟨NameOfFileEditted⟩. Initially it should be set to `XText:New`.

(b) i. New menuitem will empty the text area for the user to enter new text (the title of the text editor should be set to `XText:New`).

ii. If a file was already loaded and it was modified (see below for explanation), the user is provided with the option to save the file (option "to save or not to save"). If user response is to save, then a file dialog should open allowing the user to save the file that was already loaded.

iii. If the user selects "cancel" option in the file dialog, then the file will not be saved and the text area will be emptied as mentioned in b(i).

(c) i. Clicking the Open menuitem will open up a file dialog which will only let the user select and open documents with extensions ".text" or ".txt" (file dialog should allow the user to traverse the directory structure starting from the user's home directory). Once a file is selected it will be loaded into the text area.

ii. If there was a file already loaded and it was modified (see below), the user is provided with the option to save the file (similar to item b(ii)).

(d) Save will let the user save the loaded file. A save file dialog should be used.

(e) Exit will let the user to exit to editor. If there is a file loaded in the text area and it has been modified (see below), then the user is provided with the option to save the file.

A file opened in the XText editor is said to be modified if the user has edited the file and has not saved the file after the edit.

(f) Clicking replace menuitem will create an input dialog asking the user to input the string to be replaced. Lets refer to this string as the from-string. If the from-string is not present in the text starting from the current cursor position in the text area, you are required to create a dialog stating that the from-string is not found. Otherwise, open a new dialog asking the user to provide the string with which from-string will be replaced. Let us refer to this string as to-string. Once the user inputs the to-string you are required to replace the first occurrence of the from-string starting from the current cursor position with the to-string.

(g) Clicking Replace All menuitem will perform the same set of actions as in Replace; the only difference is that all occurrences of the from-string starting from the current cursor position will be replaced with the to-string.

(h) Instead of getting from-string from the user via a dialog, you should also allow the user to select a string from the text and then Replace/Replace All. In this case, the first step of opening a dialog for user input regarding from-string will not be necessary. For Replace, the selected from-string will be replaced. For Replace All, the selected from-string and all the proceeding from-strings present in the text area will be replaced.

(i) Clicking the Find menuitem will open a dialog for the user to input a search-string. If the search-string is found in the text starting from the current cursor position, then the first occurrence of the search-string after the cursor position will be highlighted.

(j) Small menuitem selection converts fonts to size = 10, Medium selection converts fonts to size = 15, and Large selection converts fonts to size = 20.

(k) If user right clicks (or ctrl-clicks for Mac users) on the text area, a pop-up menu should appear with the options Replace, Replace All, Find, Small, Medium and Large.

2. **Your code must be reasonably commented**

   (a) Every class must have a description. If the class is responsible for creating an user-interface, the description should state that. If the class is responsible for realizing some storage and computation, the description should state that.
   Sample comment:

```
/**
 *  ButtonPanel contains three buttons and
 *  implements the action listener that listen
 *  to the button events
 */
```

   (b) Every method that has more than one program statement must be commented.
   Sample comment:

```
/**
 *    checkRowWin method: to check whether a row has a wining combination
 *    @param index: row index of the gridElement
 *    @return 0 if player zero has winning combination in row index
 *             1 if player one has wining combination in row index
 *            -1 otherwise
 */
```

   (c) If you are using Eclipse, you can just type "/**" followed by ENTER to create the comments.

3. **You can use any code-snippets from the public domain** if you find anything useful. Provide a reference for all such code-snippets at relevant places in your source. For example, if you use the following code from this document, you can cite this document in the comment.

```
/**
 *  Obtained from homework SE/ComS 319 ...
 */
// for reading from "file"
try
{
  BufferedReader reader = new BufferedReader(new FileReader(file));
  String line = null;
  while ((line = reader.readLine()) != null)
  {
     //   write your code to put the line in your text area
  }
  reader.close();
} catch (IOException e1) { /* do something here */}

// for writing to object "file"
try
{
   PrintWriter out = new PrintWriter(new FileWriter(file));
   // "contents" hold the text in your text area in the form of String
   out.print(contents);
   if (out.checkError())
     throw new IOException("Error while writing to file.");
   out.close();
} catch (IOException e1) { /* do something here */}
```