8th homework assignment; JAVA, Academic year 2011/2012; FER

As usual, please see the last page. I mean it! You are back? OK. Here we have five problems for you to solve.

Problem 1.

We will continue our work on JNotepad application we started during lectures. I have prepared basic example (without localization) and placed it in repository on Ferko (RadniPrimjer.zip) – you should already have everything that is in there and more, so you can continue using your code from lectures.

For problem 1 your job is to complete localization as shown in slides. Write implementations of LocalizationProviderBridge and FormLocalizationProvider (check diagrams in lecture-slides). LocalizationProviderBridge derives from AbstractLocalizationProvider; it has a reference to its parent and it has inner ILocalizationListener which can be registered to its parent (when connected=true); commands connect / disconnect are there to allow derived classes to manage this status. FormLocalizationProvider is a class derived from LocalizationProviderBridge; in its constructor it registeres itself as a WindowListener to its JFrame; when frame is opened, it calls connect and when frame is closed, it calls disconnect. All JFrame components that need localization will register itself to this FormLocalizationProvider and not to global LocalizationProvider.getInstance() object.

Add method:

```
public String getLanguage();
```

into ILocalizationProvider interface and make appropriate changes where ever necessary. Make sure that your LocalizationProviderBridge remembers what was the currently set language at the moment that users called its disconnect() methods an if upon the connect() method is called the parent's current language changed, you should fire notifications to all of your registered listeners.

Problem 2.

From this point on (in this homework) you should provide adequate localization for English and Croatian for all menu items, dialog messages, dialog titles etc. To support this, you are required to implement LJMenu (localized JMenu) and LJLabel (localized JLabel) and other components if necessary for completion of this homework. Put them all into package hr.fer.zemris.java.tecaj_10.local.swing. Constructors for all of these components should accept a reference to ILocalizationProvider and a *key* that is used for their localization. They should register themselves as listeners and change displayed text when language changes.

Go through your code and make sure you have localized everything (even JFileChooser dialog title and JOptionPane messages you display to user).

Problem 3.

Add menu *Language* with options: *Hrvatski* and *English*. Choosing each item should change the current application localization.

Problem 4.

Convert your application to multi-document application by using tabs (`JTabbedPane` component).

At this point, you should provide following menu structure with appropriate action implementation:

- *File* with options *New document*, *Open document*, *Save document*, *Save as document*, *Close document*, *Exit application*. For example, *New document* will open new tab for document.

Make appropriate changes to support this. You are advised to implement additional class that will encapsulate all the data about a single document (document file path, reference to `JTextArea` responsible for its display/editing, etc). Make sure you do not allow user to open the same document in multiple tabs! Instead, in that case just activate the tab that contains that particular document.

Problem 5.

Make sure that you have menu *Edit* with following items:

- *Copy*
- *Cut*
- *Paste*
- *Clear all*

Add submenu *Edit* → *Change* case with options:

- *Uppercase*
- *Lowercase*
- *Toggle case*
- *Word case*

Activating *Word case* should make each word to have first letter uppercased and the rest ones lowercased.

Add submenu *Edit* → *Advanced* with options:

- *Sort lines ascending*
- *Sort lines descending*
- *Remove empty lines*

All of these work on selected text only. They treat the selected text as multiple lines and do transformations on lines.

Problem 5.

Add menu *Info* with item *Statistics*. Activation of *Statistics* item should bring up new modal dialog (implement new class derived from `JDialog`). Read:

http://docs.oracle.com/javase/tutorial/uiswing/components/dialog.html
http://docs.oracle.com/javase/7/docs/api/javax/swing/JDialog.html

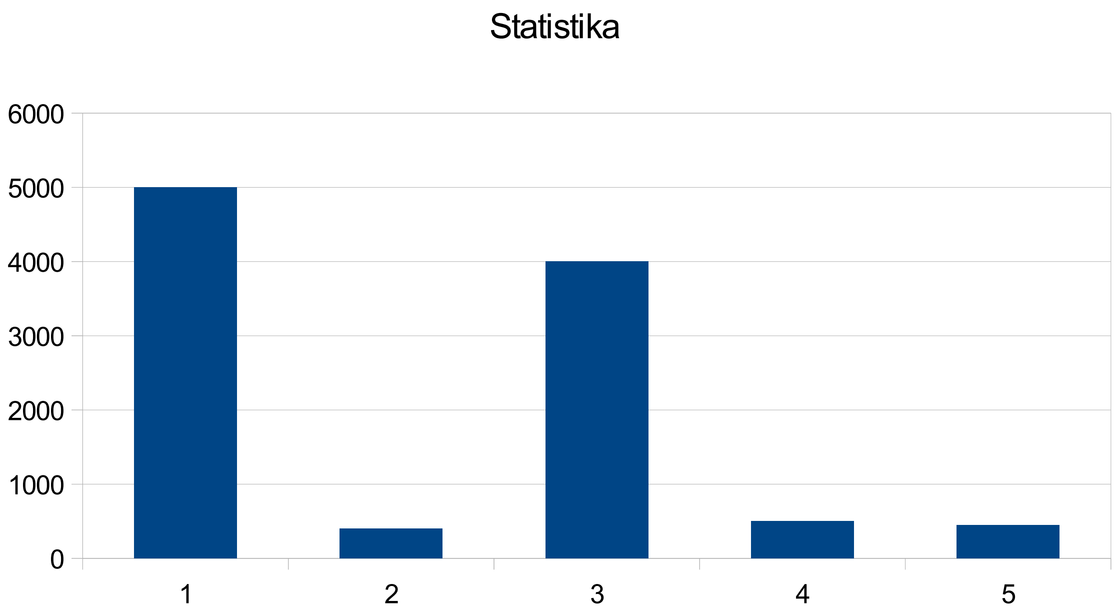
In this dialog you must display following information:

- (1) document length
- (2) number of uppercase letter

- (3) number of lowercase letters
- (4) number of spaces
- (5) number of words

Use `BorderLayout` and add these information in a table (use `JTable`) that is placed on `BorderLayout.PAGE_START`.

Create a new inner class `StatisticsVisual` that derives from `JComponent` and that creates a visual rendering of these information. You should try to mimic following rendering:



Put this component as `BorderLayout.CENTER` in your dialog.

To learn how to create components that paint itself, read end of lecture slides and then:

http://docs.oracle.com/javase/tutorial/uiswing/painting/index.html

as well as:

http://www.oracle.com/technetwork/java/painting-140037.html

Please note. You are not allowed to use any foreign library or code prepared by someone else for chart drawing. This is a simple exercise and you are expected to write it by yourself for this specific case.

Have fun!

Please note. You can consult with your peers and exchange ideas about this homework *before* you start actual coding. Once you open you IDE and start coding, consultations with others (except with me) will be regarded as cheating. You can not use any of preexisting code or libraries for this homework (whether it is yours old code or someones else). Document your code!

In order to solve this homework, create a blank Eclipse Java Project and write your code inside. Once you are done, export project as a ZIP archive and upload this archive on Ferko before the deadline. Do not forget to lock your upload or upload will not be accepted.