CSCI 2300 – Software Engineering and Object Oriented Software Design

Instructor: Javier Gamo

Project 2: Java XML parsing

Deadline: March 16, 14.00H

What to Submit

You should upload your answer into blackboard on a single ZIP file, including **all the files** you use/produce. Be sure the file you submit have exactly the following name:

FirstnameLastname_P2.zip

Example: JohnSmith_P2.zip *for Project 2 submitted by John Smith*

Since this is a team project, you must upload all the common files for the project, and a short presentation of the tasks you have developed on this project, as described below.

Moreover, you will be presenting in class your contribution to the project, during week 10 (most likely by March 17).

Background

You want to implement Java parsing for XML files using the Document Object Model (DOM). Your program will have search and replace capabilities. As **extra bonus**, you can implement the XML parsing using SAX / StAX, as described below.

There are no constrains at all regarding the design and functionally of your project, apart from the minimum requirements mentioned below.

As a first approach, you should make a command-line version. Once the command-line version is fully working, then you should make a Graphical User Interface. As a test example, you will be parsing the file `hamlet.xml`, an XML version of the play "Hamlet" by William Shakespeare. Other books from Shakespeare in XML can be found at:

http://xml.coverpages.org/bosakShakespeare200.html

This is a team project. All the grading aspects will be shared, except the presentation, which is being graded individually.

Requirements

Your program will be using the DOM to perform the following tasks **(40 points)**

- File choosing: Prompt the user to choose a XML file (by default, `hamlet.xml`)
- Number of characters: Calculate and display the number of PERSONA appearing in the play
- Number of times a character acts: Prompt the user for a name of a SPEAKER (by default, Hamlet). Calculate and display the number of times the selected SPEAKER acts on the play

- Fragment searching: Prompt the user to enter a fragment of the play (by default, "*To be, or not to be*"). If the fragment is not located in the XML file, you must show the following message, along with the time needed to do the search:

  ```
  Sorry, fragment not found. Search performed in XX.XX seconds
  ```

  If the fragment is found, you must display the full sentence where the fragment is present, and the time employed to do the search, as follows:

  ```
  The fragment has been found in the following sentence:
  "To be, or not to be: that is the question:"
  Search performed in XX.XX seconds
  Do you want to replace it? (Y/N)
  ```

  If the fragment is found in more than 1 sentence, your program should display all of them.

- Fragment replacing: If the users says `Y` or `y` to the question above, prompt the user to enter the alternative text (for instance, "*To be, or MAYBE not to be*"), and display the new line, as indicated:

  ```
  The sentence has been replaced as follows:
  "To be, or MAYBE not to be: that is the question:"
  Do you want to save the changes? (Y/N)
  ```

- File saving: If the users says `Y` or `y` to the question above, the user is prompted to enter the name of the file (by default, the original name `hamlet.xml` should be provided). If the user keeps the same name, the program should ask for confirmation.

  ```
  Do you want to overwrite the file hamlet.xml? (Y/N)
  ```

  In any case, the program should inform the user about the file saving

  ```
  File saved successfully.
  ```

Graphic User Interface(GUI): Design a GUI for your program. Your GUI should implement graphically all the functionalities for the command-line version (**20 points**).

UML and Javadoc Report: You must include a report (DOC or PDF file) describing your project. In the report, you must represent your classes, with their *data fields* (e.g. properties) and *behaviour* (e.g. methods) using UML diagrams. **Also, your code should be extensively commented. You must also provide the HTML documentation produced by Javadoc. (20 points)**

Presentation: You must also include a presentation (PPT or PDF file), describing the tasks you have performed in the project. You will be giving this presentation in class. A professional, good-quality presentation of your app will also count! (**20 points**)

EXTRA BONUS: You are encourage to write another project, using either the SAX Parser or the StAX Parser (**10 points**)

Assessment

This assignment contributes 15% of your final course grade. The maximum possible grade on this assignment is 100 points.