Exercise 1: Graphical User Interface with Java

The objective of this exercise is to experiment with graphical user interfaces and to collect or refresh the experience with Java and Eclipse.

1.1 Triangle1st GUI Application(3 points)

Develop a Java-Application named *Triangle1st* containing a graphical user interface, which calculates the *perimeter* and the *area* of a triangle. Figure 1 shows one way the user interface may look like; however you are free to implement your ideas.

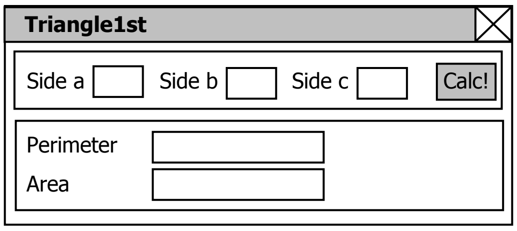

Figure 1: Proposal for designing the user interface layout.

Please keep in mind:

- Develop the GUI, for example, by writing it manually using Swing UI components.
- Create a useful layout that keeps intact if the application is resized.
- The application should contain a button to close the window.
- Put your application code in the package *at.fhhagenberg.sqe.exercise1*.
- Name the project to fit following convention: SQE01-*Lastname*_Triangle1st, where *Lastname* is your name. Put all referenced libraries in the project. Use only relative paths!
- Add a comment at the beginning of every source code file containing your full name and your student ID number.

1.2 Input Validation (1 point)

Use exception handling or related approaches to make sure that no exceptions will be raised in case of invalid inputs, e.g., non-numerical characters.

- React on non-numerical Inputs by showing an error message, for example by opening a dialog (see JOptionPane). Hint: It is sufficient to check the input when the button *Calc!* is clicked.
- Catch also other exceptions and react by displaying the exceptions' message as error description.

1.3 Externalization of text messages (1 point)

Java provides mechanisms for externalization/internationalization of applications, i.e., text messages and labels are not stored as constants in the code but in separate property files. In order to support language-independent applications all strings have to be replaced by symbolic names. The connection between the symbolic names and the actual text messages and labels is maintained by an object of the class *ResourceBundle*.

- Eclipse supports internationalization of text messages. Proceed as follows: Choose *Source > Externalize Strings…* in the context menu (right click on a Java class or project). Select the text messages you want to internationalize in the following dialog. Go through the next steps to create a class *Messages* and an associated properties file *messages.properties*.

1.4 Create a JAR file (1 point)

Java libraries are stored in JAR archive files (Java archive). Crate a JAR file from your project containing the executable binaries (class files) and the necessary resources.

- Proceed as follows: Select *Export… > Java > JAR file* in the context menu (right click on the project). Export the binaries and resources to a JAR file named *Triangle1st.jar*.
- Start your application from the JAR file via the command line. Use the command:

```
java -classpath Triangle1st.jar at.fhhagenberg.sqe.exercise1.Triangle1st
```

1.5 Documentation using JavaDoc (1 point)

The tool *JavaDoc* generates documentation in HTML format from source code comments starting with /** (see *How to Write Doc Comments for the Javadoc Tool* at http://www.oracle.com/technetwork/java/javase/documentation/index-137868.html).
Write comments for all the classes and methods you developed and generate HTML documentation.

- Proceed as follows: Select *Export… > Java > Javadoc* in the context menu (right click on project). Export the documentation in a separate directory. Pack the directory as Zip archive for submitting the exercise.
- If necessary you may need to configure the path to the Javadoc command. In that case, specify the path to the executable *bin\javadoc.exe* in your JDK directory.

1.6 Test Cases (2 point)

Consider what input should be used to test the application *Triangle1st*. Create a table containing the columns *a*, *b* and *c* for input parameter and an additional column *result* to document the expected results. Have at least 10 test cases (i.e., rows in the table), which you store in the file *SQE01-TestCases.txt*.

1.6 Questions and Answers (1 point)

Read the article by *Noel Nyman: "Does a Bug Make a Noise When It Falls in the Forest?"* (bug-in-forest.pdf) and answer the following questions. Document your answers in the file *SQE01-Q&A.txt*.

- In the case described in the article, what were the *failure*, *fault* and *error*? (Stick with the definitions from the lecture.)
- In a software project, who should decide whether a problem is a bug or not?

Submission Instructions

- Add a comment at the beginning of every source code file containing your full name and your student ID number.
- Submit the complete Eclipse project including all relevant additional files as specified (e.g., JAR files, Exercise1-TestCases.txt, Exercise1-Q&A.txt) via packing all in a Zip archive.
- The Eclipse project has to be self-contained using only relative paths to included files and libraries. Avoid absolute paths to external libraries!
- Adhere to the naming convention for projects: SQE*##-Lastname_Title* (*##* = exercise number, *Lastname* = your name, *Title* = exercise title).
- Upload the archive at the specified submission link on the elearning platform http://elearning.fh-hagenberg.at/ until the specified date, usually Tuesday 23:55 before the next lecture.
- Only complete submissions adhering to all of the above requirements are considered. Late submissions, submissions via email or submissions failing to meet the specified requirements will not be accepted.