Programmierpraktikum- Group Project

Protocol 3 - Final

1. Basic information

Program is functioning as a compiler. It allows the user to write program and required test. Program starts from RED Phase, user is forced to write a failing test first, accordingly to the TDD rules, next user writes program's code and after that he is able to refactor it. It is described as RED, GREEN and REFACTOR phases. Phase changes from RED to GREEN when the code/test is not compiling or exactly one test fails.
In the REFACTOR phase, user can choose whether he wants to edit program's code or tests'. He can switch to RED phase when program as well as tests, compile correctly. Phases in exercises are limited with babysteps turned on.

2. Structure

This application is "separated" into 4 significant parts: "main part", "XML part", "logic part" and "GUI part".
This division is just symbolic, to simplify code's interpretation.
Project includes also tests.

- "main part" classes
  - Main.java
    - This class extends Application which is an entry point of JavaFX which allows us to create a user friendly interface to interact with the user. It implements a method *start()* which is responsible for defining basic settings for the xml scene.
      In the *main()* method is used in case the application cannot be launched.

- "XML part"
  - Config.java
    - This class saves babysteps time and other configurations.

- Exercise.java
  - Class Exercise is responsible for storing all the needed information, holds constructors for those variables.

- ExerciseList.java
  - ExerciseList has a similar role as Exercise, it holds the values using ArrayList.

- SNClass.java, SNClassList.java, SNTest.java, SNTestList.java
  - These are inner classes, are responsible for storing content, by declared constructors and *get(), set()* functions return needed values.

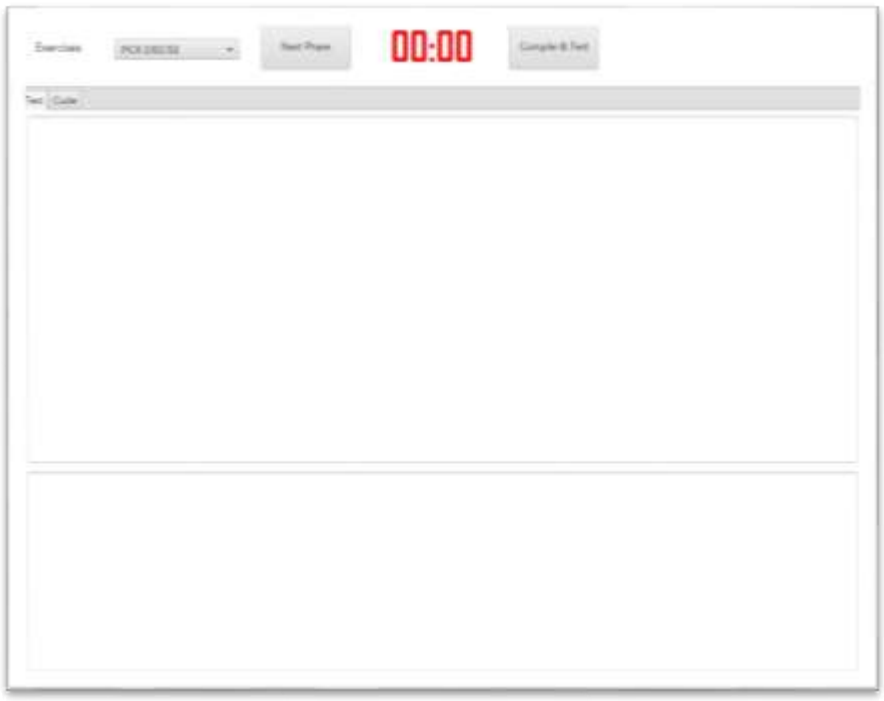
- XmlParser.java
  - This is most important class of this part because it is a trigger point.
    A document in which user will be able to write the code is being created, provides us with exercise list, takes part in handling time tracing time necessary for baby steps.

- "logic part"
  - Babysteps.java
    - This class extends AnimationTimer, class that works by specify a method that will be called in every frame (*handle()*). AnimationTimer implements 3 methods *handle(), start() and stop()* that allows to create a timer, handle it start and stop it. Babysteps class creates the timer that counts time while user is writing code and ends the exercise when the time is up.

  - Logic.java, Phase.java
    - Logic class starts an exercise and changes the phase of it. Program starts from the RED phase, because user must first write a failing test. Application is running depending on babysteps (time allowed). It changes to GREEN when the code/test is not compiling or exactly one test fails.
      After that same happens with program's code and when GREEN – refactoring is possible.

Phase class is just responsible for storing a special data type which is enum which consists of named values which can be assigned as any of the enumerators as a value. It is basically a handy way to access and refer to some constancies. In this case RED, GREED, REFACTOR.

- "GUI part"
    - GUIController.java, Model.java
        - These classes control Graphical user interface. GUIController implements Initializable, as a consequence a method *initialize()* will be used to initialize the controller. It also includes basic JavaFX components like button or label. In the class Model, helps parsing XML, creating XmlParser to access data in the document and to add data to the exercise.



3. Open Points

1. Can an exercise consist of multiple classes and test files?

   No it can’t.

2. It time limited?

   No, the phases in exercises are limited with babysteps turned on.

3. Are there any libraries necessary?

   Yes, gradle-wrapper.jar.

4. Which languages are allowed?

   Java.

5. Are there any additional requirements?

   Java 8 is necessary for the correct compilation.