Cairo University Faculty of Computers and Information



**CS352 – Software Engineering II**

**Phase 1 Template**

**2017**

**Project Team**

|  |  |  |  |
| --- | --- | --- | --- |
| **ID** | **Name** | **Email** | **Mobile** |
| 20140289 | Nada Ashraf Ahmed | nadanodyy@ymail.com | 01100591922 |
| 20140213 | Michael Wageuh Ramzy | michaelwageuh12@hotmail.com | 01200634332 |
| 20140281 | Monica Millad Aziz | Monicamillad18@gmail.com | 01271223662 |
| 20140287 | Mina Nabil William | mnabil405@gmail.com | 01142642703 |

Contents

[Design and CODE Check List 3](#_Toc476413281)

Suggestion for improvements……………………………………………………………………………………………………………………….5

[Testing 6](#_Toc476413282)

Changes in the code……………………………………………………………………………………………………………………………………..9

[Git repository link 9](#_Toc476413283)

# 

**Design and Code Checklist**

**Notation:**

∏ for Yes.

∏ for No.

**Design Principles**

1. Does the design follow SOLID principles? ∏ What % 40

Related Issues:

* + Single Responsibility Principle:

There is one controller that responsible for most of the main functionalities and mixing between them and the views (setGameButtons , setStudentHomeListenner ,

Register ).

* + Open Close Principle:

It was difficult to test it so what about extensions!!!

* + Liskov’ Substitution Principle:

It was achieved in this code as they extend the class student and teacher from user.

* + Interface Segregation Principle:

The views give each user the functionalities that they need only.

1. Does the design follow OOP rules? ∏ What % 90
2. Is the design simple and easy to modify? ∏ What % 70

Related Issues:

The design do not match the code .

**Coding Standards**

1. Is the code understandable and readable? ∏ What % 50

Related Issues:

It is readable but it’s not understandable.

1. Does the code follow Java Coding Style? ∏ What % 90
2. Is indentation used properly? ∏ What % 90
3. Do variable have good names? ∏ What % 100

**Comments**

1. Is the code commented enough? ∏ What % 100

Related Issues:

There are no comments.

1. Is every class and method commented? ∏ What % 100

Do comments follow Javadoc style? ∏ What % 80

Related Issues:

Eg. (@Override).

1. Is Javadoc generated for all the code? ∏ What % 100
2. Are there useless / wrong comments? ∏ What % 100

Related Issues:

There are no comments.

**Code Structure**

1. Does the code follow the design precisely? ∏ What % 60

Related Issues:

There are classes that do not included in the diagrams and are included in the code and vice versa.

1. Are there very long classes or methods? ∏ What % 80

Related Issues:

Eg. Class Controller is very long class with so much functionalities and StudentController class is empty.

1. Is there repeated code ?(put put in a function) ∏ What % 100

**Error Handling**

1. Does the code handle errors and exceptions? ∏ What % 50

Related Issues:

In function Register if there are missing data the user will be created and if the data that was entered already existed the user will not be created however the message says the opposite; “Account has been created!”.

1. Is defensive programming used to avoid errors? ∏ What % 80

**Logic**

1. Do loops have correct conditions and bounds? ∏ What % 100
2. Do loops always terminate? ∏ What % 100

**Overall**

1. **Are the design and code of good quality?** ∏ What % 40

Related Issues:

About the code quality:

* There is a single user that performing all the functionality of the system, and there is no creation for a new object in any scenario( Users, Games ) .
* Some checking for validation are done in the functions that write on files.

# Suggestion for improvements

* Do not write the main functions in the GUI.
* Separating the Entities from the Models.
* Apply the single responsibility principle on the controllers as there is almost one controller that is responsible for all the main functions.

# Testing

1. **ControllerTest2 class (login)**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Number** | **Testing function** | **Description** | **Result** | |
| **1.** | **setUser(usersTable.selectUser(loginWindow.getUserNameTextField().getText(), loginWindow.getPasswordTextField().getText())** | **Testing function setUser by checking if the frame is visible with function isvisible .** | **Test1** | **Failed** |
| **Test2** | **Failed** |
| **Test3** | **Passed** |
| **Test5** | **Passed** |
| **Test6** | **Passed** |
| **Test7** | **Passed** |
| **Test8** | **Passed** |

1. **ControllerTest3 class (register)**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Number** | **Testing function** | **Description** | **Result** | |
| **1.** | **addUser(User u)** | **Testing registering by checking the value this function returns (Boolean value) and compare it with the expected value.** | **Test1** | **Passed** |
| **Test2** | **Passed** |
| **Test3** | **Passed** |
| **Test4** | **Passed** |
| **Test5** | **Passed** |
| **Test6** | **Failed** |
| **Test7** | **Failed** |
| **Test8** | **Failed** |
| **Test9** | **Failed** |
| **Test10** | **Failed** |

1. **ControllerTest3 (PlayGame)**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Number** | **Testing function** | **Description** | **Result** | |
| **1.** | **Main.tController.sHome.startGame.doClick()** | **Testing this by checking the name of the game the used chose and the name of the object currentGame** | **Test1** | **Passed** |
| **Test2** | **Passed** |
| **Test3** | **Passed** |
| **Test4** | **Passed** |
| **2.** | **Main.tController.gameTF.submitButton.doClick()** | **Testing the functionality of the game by checking the score of the currentGame after playing** | **Test5** | **Passed** |
| **Test6** | **Passed** |
| **Test7** | **Failed** |
| **Test8** | **Failed** |

1. **GamesHandlerTest class**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Number** | **Testing function** | **Description** | **Result** | |
| **1.** | **getGames()** | **Testing by reading all the games with getGames() fun and then adding the new game and checking if the game was added or not and if it was the expected scenario (it was supposed to be added or not).** | **Test1** | **Passed** |
| **Test2** | **Passed** |
| **Test3** | **Passed** |
| **Test4** | **Failed** |
| **Test5** | **Failed** |
| **Test6** | **Failed** |
| **Test7** | **Failed** |
| **Test8** | **Failed** |
| **Test9** | **Failed** |
| **Test10** | **Failed** |
| **Test11** | **Failed** |
| **Test12** | **Failed** |
| **Test13** | **Failed** |
| **Test14** | **Failed** |

# 

# Changes in the code

1. Making the variable tcontroller in the main function attribute in the Min class as it is need in all the functionalities.
2. Adding some getters and setters to be able to access some needed variable.
3. Making trueRadio and falseRadio buttons attributes in TFGameForm class.

# Git repository link

**https://github.com/monicamillad/SoftwareEngineering-2.git**