Cairo University Faculty of Computers and Information

CS352 – Software Engineering II

Phase 1 Template

2017

Staff:

Dr Amr Kamel	a.kamel@fci-cu.edu.eg
Dr Khadiga Mohamed	kelbedweihy@fci-cu.edu.eg

TA: Ragia Mohamed Aboulfadl

TAs:	Eng Mohamed Samir	m.samir@fci-cu.edu.eg
	Eng Omar Khaled Ali Ragab	o.khaled@fci-cu.edu.eg
	Eng Ragia Mohamed	r.mohamed@fci-cu.edu.eg
	Eng Ebtehal yahia	ebtehal.yahia@fci-cu.edu.eg
	Eng Ahmed Emad	ahmed.emad@fci-cu.edu.eg
	Eng Amr Tarek	a.tarek@fci.cu.edu.eg

CS352: Phase 1 – Gamers++ <edugame>

Phase 1 document

Project Team

ID	Name	Email	Mobile
20140066	Alaa Atef Badr	alaa.badr.25@gmail.com	01140911255
20140062	Ismail Ahmed Mahmoud	ismail.ahmed2014@stud.fci-cu.edu.eg	01005887976
20140206	Karim Ehab Ahmed	karimehabahmed@stud.fci-cu.edu.eg	01062064070
20140274	Mostafa Mohamed ElMenshawy	mustafa1elmenshawy@gmail.com	01148504856

Contents

Phase 1 document

Review Check List:
Design and Code Checklist

Design Principles

1- Does the design follow SOLID principles?

Percentage: 0%
Related Issues: As database classes do more than one job, saving the data entered and also validating all of it through different checking functions. Also, there is only one boundary class deals with all the GUI, forms and pages of the system of around 2000 lines.

2- Does the design follow OOP rules?

Percentage: 60%
Related Issues: All classes are used properly with private variables, setters and getters. But abstraction is not used in Account class as its inherited class are nearly not used because it has a Boolean showing the type of the account. Also, class Question should be made and True or False, Match and MCQ Questions could inherit from it, as the all include the same attributes and nearly make similar functions.

3- Is the design simple and easy to modify?

Percentage: 40%
Related Issues: Having classes doing more than one job is making the whole project hard to modify. As in the boundary class if I want to change anything in the GUI or the forms. Also, database class are responsible for storing as well as validating data, which makes it hard to modify later on.

Coding Standards

4- Is the code understandable and readable?

Percentage: 50%
Related Issues: There is only one boundary class deals with all the GUI, forms and pages of the system of around 2000 lines, which makes it hard to read and understand the job of the whole class. Also, not some of the variables' names are ambiguous.

5- Does the code follow Java Coding Style?

Percentage: 50%
Related Issues: Functions' names start with capital letters, a class name "cleverZone" starts with a small letter. Also, some blocks and conditions of one line do not have braces. More than one

statement are written in the same line.

6- Is indentation used properly?

Percentage: 50%
Related Issues: Some variables' names have typos. Some if conditions are all written in one line. Different tabs and spacing are used. More than one statement are written in the same line.

7- Do variable have good names?

Percentage: 30%
Related Issues: Many variables in the GUI (boundary) class have ambiguous names. Also, some have typos, others names are some unknown abbreviations.

Comments

8- Is the code commented enough?

Percentage: 0%
Related Issues: The code is not commented at all. Commenting is only used to omit code lines.

9- Is every class and method commented?

Percentage: 0%
Related Issues: Only methods generated by the IDE have generated comments as well.

10- Do comments follow Javadoc style?

Percentage: 0%
Related Issues: There is no comments in the first place.

11- Is Javadoc generated for all the code?

Percentage: 5%
Related Issues: Only those generated by the IDE, mostly for setters and getters.

12- Are there useless / wrong comments?

Percentage: 0%
Related Issues: Because there is really no comments in the code.

Code Structure

Phase 1 document

13- Does the code follow the design precisely?

Percentage: 60%
Related Issues: Some sequence diagrams do not fully describe the coding sequence. Also, some use cases do not represent exactly what is done throughout the system. And also some design concepts are not yet implemented, as anyone can enter the system even if it is an unregistered user.

14- Are there very long classes or methods?

Percentage: 100%
Related Issues: As CleverZone class is responsible for all the boundary in the system, it is very long with 2000 lines in the initialization function.

15- Is there repeated code? (put in a function)

Percentage: 0%
Related Issues: There is no any repeated code.

Error Handling

16- Does the code handle errors and exceptions?

Percentage: 0%
Related Issues: When creating a game, you can choose correct answer other than the entered choices. Pressing buttons in different order cause system to throw exception and freeze. True or false questions accept answers either than both. Only one email can be used to register more than once.

17- Is defensive programming used to avoid errors?

Percentage: 0%
Related Issues: Due to the same reasons mentioned before.

Logic

18- Do loops have correct conditions and bounds?

Percentage: 100%
Related Issues: Loops are only of one kind “for loops”, they always loop on the size of the datatype used, so it is always have a correct condition and bound with a good coding syntax.

19- Do loops always terminate?

Percentage: 100%
Related Issues: Loops always terminate with the stopping condition written in the beginning of the “for loop”.

Phase 1 document

Overall

20- Are the design and code of good quality?

Percentage: 50%

Related Issues: There is an unused variable in the code, and some unused functions. Also, with only one email, more than one account can be registered. Magic numbers appear in the code. Design is not fully compatible with the code. Code needs to be modified and properly arranged. Comments needed to be inserted. The Classes need to be described precisely. There is also some missing relations, as the one between delete a game and removing it from a classroom. Many includes are missing in the design document. Meaningless exceptions are written in all the 20 use case tables, all about logging out from the system. Good writing style must be used, as capitalization, punctuation, spacing and numbering.

Suggestions for improvement:

- Use MVC properly by dividing the responsibilities across different control classes.
- Apply the 'S' in the SOLID principles: One class for the boundary is not recommended. As well as control. Database is responsible for storing and retrieving but not validation of the data.
- Abstraction can still be applied in a more proper way across classes. Example: Account, Student and Teacher classes. Also, connecting the three classes responsible for questions to one general class.
- Exception handling and logical errors.
- Consistency between documentation, design and coding for easier extending and reusability.

Testing

1. T_FQuestionTest

Number	Testing function	Description	Result
1.	setAnswer(answer)	Testing function for setAnswer function in the True or False game. It tests if the teacher enters different answers than true or false while creating the game, using 7 different test cases.	3: Passed 4: Failed

Phase 1 document

2. DataBaseUserTest

Number	Testing function	Description	Result
1.	SaveAccount_Test (account)	Testing function for SaveAccount function in the Database of the user. It tests the registration sequence, using 4 different test cases. Assumption: unique username for each user account.	4: Passed
2.	checkData(account, result)	Testing function for checkData function in the Database of the user. It tests the availability of the username and the validity of the email entered while registration, using 4 different test cases. Assumption: email must contain '@' and '.' To be valid.	3: Passed 1: Failed

Phase 1 document

3. cleverZoneTest

Number	Testing function	Description	Result
1.	T_FAnswer	Testing function for the button “Submit Answer” action function in the True or False game. It tests the change in the entered answer by the teacher while creating the game to be valid. Unfortunately, answers that do not start with ‘T’ or ‘F’ remain as they are which freezes the system while playing. This is tested using 7 different test cases.	6: Passed 1: Failed

Git repository link

https://github.com/IsmailAhmedIsmail/SWE-Project.git