PPA Assignment 4

Wonjoon Seol, Computer Science with Intelligent Systems, K1631098

November 6, 2016

1 Introduction

The assignment creates an exam marker program which compares solution, awards marks and classification. The marker takes Exam objects as parameters which consist of 3 questions (Numerical question, Boolean Question and Multiple choice question). I need to demonstrate my ability to use conditional statements, constructors and be able to interact with multiple classes (Topic 1,. . . ,5).

2 Pseudocode

2.1 Class NumericalQuestion

Pseudocode 1: This class models numerical question.

```
1  Initialise private integer answer
2  Initialise private integer mark
3  Define NumericalQuestion
4      Set answer
5      Set mark
6  Define getAnswer
7      Return answer
8  Define getMark
9      Return mark
10 Define setMark
11     Set answer
```

2.2 Class BooleanQuestion

Pseudocode 2: This class models Yes or No type question.

```
1  Initialise private Boolean answer
2  Initialise private integer mark
3  Define BooleanQuestion
4      Set answer
5      Set mark
6  Define getAnswer
7      Return answer
8  Define getMark
9      Return mark
10 Define setMark
11     Set answer
```

2.3 Class MultipleChoiceQuestion

Pseudocode 3: This class models multiple choice type question.

```
1  Initialise private Boolean option1
2  Initialise private Boolean option2
3  Initialise private Boolean option3
4  Initialise private integer mark
5  Define MultipleChoiceQuestion
6      Set option1
7      Set option2
8      Set option3
9      Set mark
10 Define getOption1
11     Return option1
12 Define getOption2
13     Return option2
14 Define getOption3
15     Return option3
16 Define getMark
17     Return mark
18 Define setMark
19     Set answer
```

2.4 Class Exam

Pseudocode 4: This class holds questions and mark, serve as a question paper or mark scheme.

```
1  Initialise private NumericalQuestion question1
2  Initialise private BooleanQuestion question2
3  Initialise private MultipleChoiceQuestion question3
4  Initialise private integer totalMark
5  Define Exam
6      Set question1
7      Set question2
8      Set question3
9      Set totalMark
10 Define getQuestion1
11     Return getQuestion1
12 Define getQuestion2
13     Return getQuestion2
14 Define setTotalMark
15     Set setTotalMark
```

2.5 Class Marker

Pseudocode 5: This class models test marker process.

```
1  Define markAttempt
2  if Answer of numerical question of examAttempt equals markscheme's then
3  |  Award full mark of question 1 from markscheme to question1's mark of examAttempt
4  else if Difference between answers of question 1 is either +1 OR −1 then
5  |  Award 1 mark off from full mark of markscheme
6  else if Difference between question 1 from markscheme is less than or equal to 5 OR
   Difference between question 1 from examAttempt is less than or equal to 5 then
7  |  Award 1 mark
8  else
9  |  Award no mark
10 end
11
12 Print Mark of question 1 out of full mark
13
14 if Answer of Boolean Question are equal then
15 |  Award full mark of question 2 from markscheme
16 else
17 |  Award 0 mark
18 end
19
20 Print Mark of question 2 out of full mark
21
22 if option 1 from question 3 are equal then
23 |  Award 1 mark
24 if option 2 from question 3 are equal then
25 |  Award 1 mark
26 if option 3 from question 3 are equal then
27 |  Award 1 mark
28 end
29
30 Print Mark of question 3 out of full mark
31
32 Define convertMarksToClassification
33    Initialise local float classificiation
34    if supplied mark is greater than or equal to supplied firstBoundary then
35 |     Set classification to 1.1
36    else if supplied mark is greater than or equal to supplied upperSecondBoundary then
37 |     Set classification to 2.1
38    else if supplied mark is greater than or equal to supplied lowerSecondBoundary then
39 |     Set classification to 2.2
40    else
41 |     Set classification to 0.0
42    end
43    Return classification
```

2.6 MarkExams

Pseudocode 6: This class is going to drive our program

Initialise NumericalQuestion $nqMarkScheme$ with $120369, 11$
Initialise BooleanQuestion $bqMarkScheme$ with $true, 1$
Initialise MultipleChoiceQuestion $mcpMarkScheme$ with $true, true, false, 3$
Initialise Exam $markScheme$ holding these three questions and total mark 15

Initialise NumericalQuestion $nqAttempt$ with $120368, 0$
Initialise BooleanQuestion $bqAttempt$ with $false, 0$
Initialise MultipleChoiceQuestion $mcpAttempt$ with $false, true, false, 0$
Initialise Exam $examAttempt$ holding these three questions and total mark 0

Initialise Marker $marker$
Mark $examAttempt$ using answers from $markscheme$ from $markAttempt$ method

Print total mark for this attempt
Print $classification$

3 Class Diagram

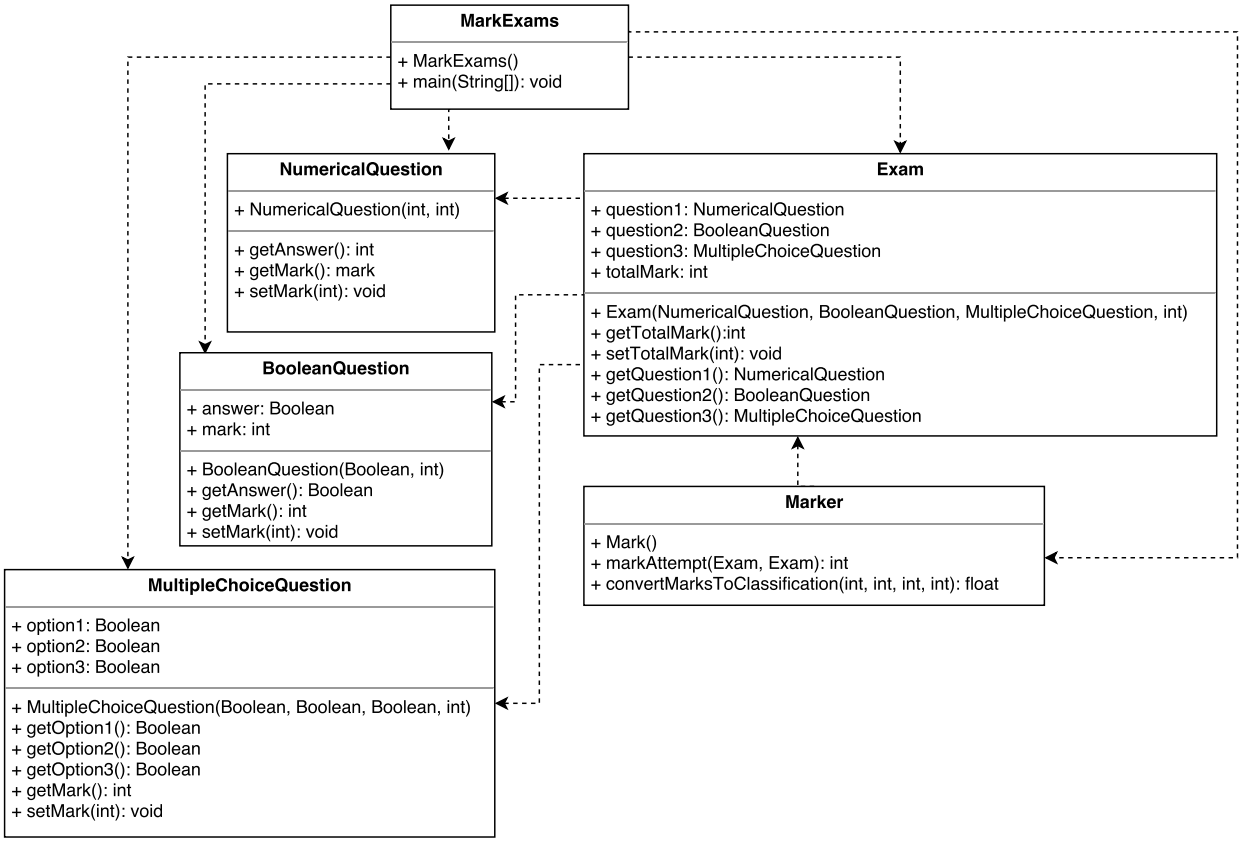

Figure 1: MarkExams Class Diagram.

4 Description

As you have noticed, I have started using Latex to generate the document for this assignment. There have been gradual improvements in my past documents. I have used basic paint tool avail-

able in the Strand computer lab to draw class diagram until assignment 2. After that, I have started to use draw.io instead and also stopped copying entire codes for a line by line description. I hope this trend of gradual improvement will continue in my future assignment too.

I would also like to let you know that due to the change in the word processor, there has been some inevitable change in my pseudocode. i.e, I have used *integer* instead of *int* and have *end* at the end of if statement, as *end* syntax is default one available for if syntax in algorithm2e package. I thought it looked nicer too. For the rest of the syntax, I have reconfigured algorithm2e package syntax to stay consistent with my previous style of pseudocode as much as possible.

1. Class NumericalQuestion, BooleanQuestion, MultipleChoiceQuestion
   These classes are constructed with a number of provided solutions using appropriate data type and its mark. BooleanQuestion and MultipleChoiceQuestion have yes and no solution so Boolean data type is acceptable. These also have getters to call answers and setters to set its mark.

2. Class Exams
   This class accepts each of above classes as an object and create an Exam object. In real life, this constitutes to question paper or solution paper. Initially, I expected to create two separate extra classes: QuestionPaper and Markscheme. As this program scales, we will have many questions and solution paper. (i.e. Core mathematics 1 question paper and its solution paper from the year 2010 to 2017). Having separate classes for question paper and mark scheme (constitute to cabinets in real life) would be better to organise larger data set. However, I expect there will be better data management solution instead of using objects. I look forward to the coming lectures.

3. Class Marker
   The most interesting class in this assignment. There was an alternative way to implement markAttempt. In multipleChoiceQuestion, Instead of making three separate methods to return individual options I could have used a conditional getter: getOption(int i) instead. The benefit of using this is being able to use for-loop inside markAttempt to cycle each getOption(i) in examAttempt and compare to getOption(i) in markscheme. This would have substituted a large number of code lines here. But the immediate problem of using conditional getter is a default case, especially when others are using my code. If this program scales to accept a further number of options but users do not update this new range of index i, the compiler will no longer inform the users that they have made a mistake and will start to return unexpected values. This is quite a significant issue and therefore not implemented here.

   On the other hand, the method $ConvertMarksToClassification$ is straightforward. One thing of an interest is initially I thought it would be better to pass Exam *examAttempt* directly to the method, but the brief mentioned *suppliedmark* so I have changed its parameter. Furthermore, I could have made a constructor so that marker object could initialise with private object attemptExam and markscheme. But I did not see a benefit of having a specific marker for each question paper and its mark scheme.

4. Class MarkExams
   This is our driver class, where the user input all the necessary data and run the program. one thing that could be improved is to use a local variable instead of manually calculating the total mark for my markscheme object.

   If the program scales and accepts many more questions than three, the users are likely to make arithmetic mistakes. Therefore it would be better to set a variable dedicated to this:
   int totalMark = nqMarkScheme.getMark(i_1) + ... + mcpMarkScheme.getMark(i_n);
   Exam markScheme = new Exam(nqMarkScheme, bqMarkScheme, mcpMarkScheme, totalMark);