

Assessment Cover Sheet- Institute of ICT

Course Title	Advanced Diploma in IT	Unit Number & Title	ITSFT-406-1502 Programming Concepts
Lecturer	Elaine Farrugia	Assignment Title	Advanced Programming Concepts and Application
Verified by	Carlo Mamo	Date	
Date Set	19/12/2016	Deadline Date	17/01/2017
Class/group	Please mark as appropriate: IT-SWD-4.2A IT-SWD-4.2B IT-SWD-4.2C IT-SWD-4.2D IT-MSD-4.2A IT-MSD-4.2B IT-MSD-4.2C IT-MSD-4.2D	Academic Year	2016-2017
Student Name		ID Number	

Student's declaration prior to hand-in
☐ I certify that the work submitted for this assignment is my own; and that I have read and understood MCAST/the College's copying and plagiarism policy.

Student's declaration on assessment special arrangements: LEAVE BLANK IF NOT APPLICABLE
☐ I certify that adequate support was given to me during the assignment through the Institute and/or the Inclusive Education Unit.
☐ I declare that I refused the special support offered by the Institute.

Student Signature:		Date :	
---------------------------	--	---------------	--

Assessment Parameters	Max Mark	Marks Achieved
KU3.3 Describe the importance of adding comments to the code	5	
KU4.1 Describe how to debug an application	5	
AA3.2 Demonstrate how to add various controls to the form	7	
AA3.3 Demonstrate how to add controls to the form programmatically	7	
AA4.1 Demonstrate the basic concepts of what exceptions are	7	
SE3.1 Develop a simple windows application	10	
SE4.1 Develop an error handling routine within the code	10	
Total Assignment Mark	51	

Feedback			
Learner signature		Date	
Assessor signature		Date	
Internal Verifier		Date	

Verifier Comments			
Verifier Name		Date	
Verifier Signature			

ITSFT-406-1502: Programming Concepts

Assignment 2

Advanced Programming Concepts and Application

December 2016

Marking Scheme:

AA3.2, AA3.3, SE3.1, SE4.1 – Windows Application	34 marks
KU3.3, KU4.1 – Comments and Debugging	10 marks
AA4.1 – Exception Handling	7 marks
Total:	51 marks

Guidelines (please read):

- You will be given at least **3 weeks** to complete this assignment. Late assignments will not be accepted.
- If you use any resources (including the internet) in order to complete your assignment, these must be referenced. Copying from such sources or from other students will result in the enforcement of the current disciplinary procedures.
- If the sources are correctly referenced but the material is used without being understood and adapted, such disciplinary procedures may still be enforced.
- Although the lecturer will guide you, you are expected to do your best, and it is your responsibility to complete the assignment and solve the difficulties that you encounter.
- When the deadline is due you should submit;
 - i. A printed document, containing the answers for Tasks 2 and 3. Make sure your answers are clear and well-explained. Bind your document or submit it in a flat file.
 - ii. A CD containing the NetBeans project covering Tasks 1 and 2, as well as a digital copy of the same documentation as explained in i).
- Note that assignments handed in without the assignment cover sheet will be considered as NOT submitted.
- You also have to submit your documentation on Turnitin using details provided by your lecturer. Failure to do this can result in your assignment being considered as not submitted.

Scenario

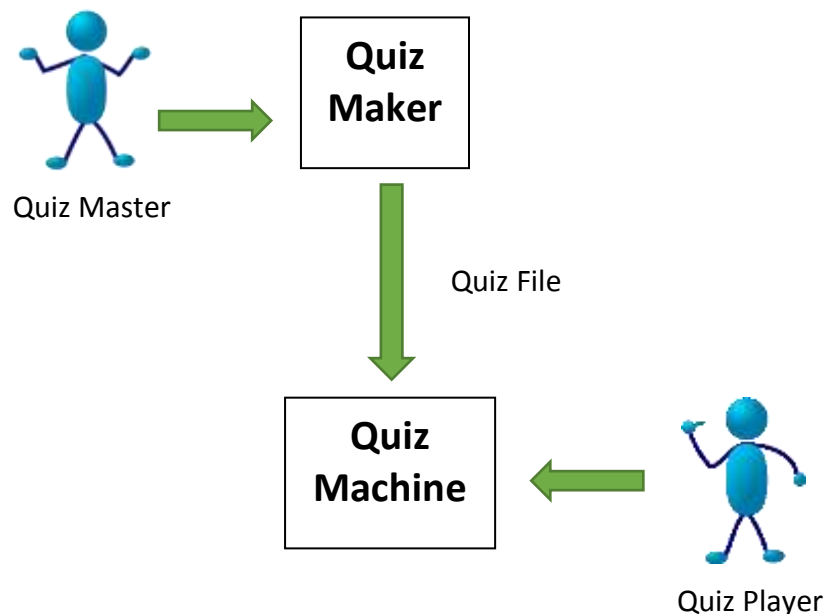
In this assignment you are going to create an application using which the users can create and play multiple-choice quizzes. The system will consist of two programs;

- Quiz Maker

This program will be used by the Quiz Master to create quizzes. The Quiz Master will be allowed to input a set of questions about a subject. For each question the Quiz Master will also input 4 answers, and mark which of them are correct (for each question, 1 or more answers are correct). The Quiz Master can then export the quiz in a text file. The quiz can be opened later by the Quiz Maker for editing.

- Quiz Machine

This program will be used by the Quiz Player to play quizzes created by the Quiz Maker. Quiz files created from the Quiz Maker can be opened using this program. The questions are displayed one by one, and the user is allowed to play the quiz. At the end, the user is shown the score obtained.



TASK 1 - Develop the application

AA3.2 - Demonstrate how to add various controls to the form

AA3.3 - Demonstrate how to add controls to the form programmatically

SE3.1 - Develop a simple windows application

SE4.1 - Develop an error handling routine within the code

34 marks

a) Common Classes for the Quiz Maker and Quiz Machine

In your programs there will be some code which both the Quiz Maker and the Quiz Machine will need to use. In this section you will create this code. Create a new Project for your Assignment.

- i) Create a class called *FileIO*. Inside it, code the following methods. (Note that minor modifications to these method signatures are allowed if approved by the lecturer).

METHOD 1

```
/* This method should display the Save Dialog and return a String
containing the path of the chosen file. If the user does not
choose a file, it should return an empty String. By default, the
filename should be set to fileToSave.txt, and this should be shown
in the File Save dialog. This can then, optionally, be edited by
the user. */
```

```
public static String showSaveDialog()
```

```
{//code goes here}
```

METHOD 2

```
/* This method should display the Open Dialog and return a String
containing the path of the chosen file. If the user does not
choose a file, it should return an empty String. The Open Dialog
should make use of a file filter to display only text files. */
```

```
public static String showOpenDialog()
```

```
{//code goes here}
```

METHOD 3

```
/* This method should take as input the path of a file, and return
a String with the contents of that file. After every line it
reads, it should add "\n" to the accumulated String. In case of
an exception, it should show a user friendly message in a dialog
box. */
```

```
public static String readTextFile(String filePath)
```

```
{//code goes here}
```

METHOD 4

```
/* This method should take as input the path of a file, and a
String to be written to the file. The file should be created if
it does not exist, and overwritten if it exists. This method does
not return anything. In case of an exception, it should show a
user friendly message in a dialog box. */

public static void writeToTextFile(String filePath, String
toWrite)

{ //code goes here }
```

- ii) Create a JFrame Form called *About*. This is a simple JFrame with just a label containing information about the program (A greeting, your name and year).

b) Quiz Maker

In this section you will develop the Quiz Maker. Where possible, make use of the *FileIO* class in order to avoid code duplication. Create a new JFrame Form called *QuizMaker*.

- i) You need to make use of 3 *ArrayLists* in your program, as follows:

- An *ArrayList* called **questionList** to store all the questions in the quiz (Strings)

E.g. {*"Which of these foods contain a high level of carbohydrates?"*,
"What is the function of proteins?",
"Which of these are vitamins?"}

- An *ArrayList* called **answerList** to store all the answers (Strings)

E.g. {*"Chocolate"*, *"Meat"*, *"Pasta"*, *"Cheese"*,
"Regulates temperature", *"Provides acids"*, *"Promotes Growth"*, *"For eyesight"*,
"B12", *"C"*, *"Z"*, *"A"*}

Note that the list contains 4 answers for each question in the question list (in the same sequence). For example the answers of question 2 would be in locations 4-7.

- An *ArrayList* called **correctAnswers** to store all the boolean values specifying whether each answer in the answer list is correct or not.

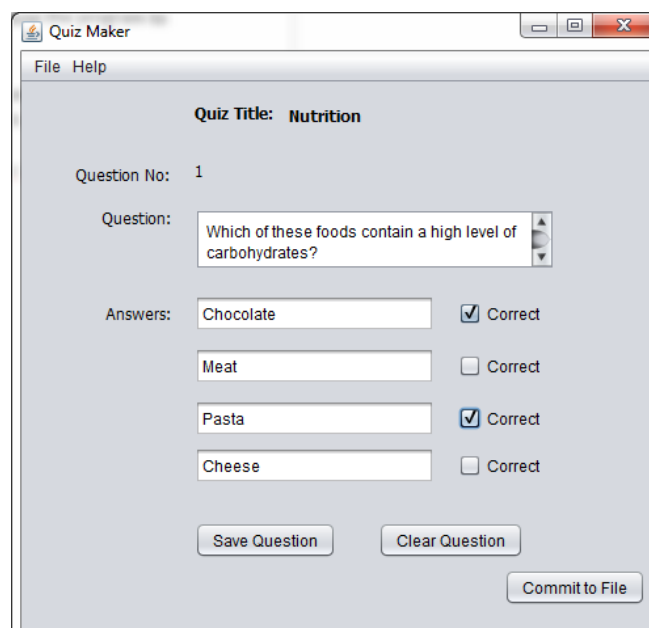
E.g. {*true*, *false*, *true*, *false*,
false, *false*, *true*, *false*,
true, *true*, *false*, *true*}

- ii) When the program is executed it should only display the menu bar, containing the options to *Start a New Quiz*, *Open a Quiz*, *Quit*, and view information *About* the program.



- iii) Clicking the *Quit* menu item should cause the program to terminate.
- iv) Clicking the *About* menu item should open another *JFrame* (already created) showing information about the program. Closing the *About* window should **not** cause the program to terminate.
- v) Clicking the *New Quiz* item should cause the *File Save* dialog to open (use *FileIO* class). The user should specify the name and path of a text file where to save the quiz.

After the Save dialog, the following GUI should be displayed in the JFrame:



Quiz Maker

File Help

Quiz Title: Nutrition

Question No: 1

Question: Which of these foods contain a high level of carbohydrates?

Answers:

Chocolate	<input checked="" type="checkbox"/> Correct
Meat	<input type="checkbox"/> Correct
Pasta	<input checked="" type="checkbox"/> Correct
Cheese	<input type="checkbox"/> Correct

Save Question Clear Question Commit to File

- The Quiz Title should automatically be set to the filename which the user would have entered, without the extension. To complete this step, create and use a method with the following signature:

```
/* This method should take as input a path, and return only  
the filename, without the extension, by using string  
processing techniques */  
  
public String getQuizNameFromPath(String path)  
  
{//code goes here}
```

- The Question number should be displayed and be non-editable.
- The user can input the question in a text area, and 4 different answers in text boxes. For each text box, the user should set whether that particular answer is correct or not, by ticking (or not ticking) the radio buttons.
- The button “*Save Question*” should first validate user input (ensure that the question is not empty, each of the answers is not empty, and that at least 1 answer is marked as correct). After validation is complete, the program should save the user input in the appropriate *ArrayLists*. After that, the question number should be incremented, and the form should be cleared for entry of another question.

To complete this step, create and use methods with the following signature:

```
/* This method should validate the user input to ensure that  
the question is not empty, each of the answers is not empty,  
and that at least 1 answer is marked as correct. It returns  
an empty String if there are no problems, and a String with  
an appropriate error message in case of a problem. */  
  
public String validateEntries()  
  
{//code goes here}  
  
  
/* This method should add the current question, answers, and  
boolean values to the appropriate lists */  
  
public void addQuestionToLists()  
  
{//code goes here}  
  
  
/* This method should clear all the text areas, text boxes,  
and radio buttons in the form. */  
  
public void clearAll()  
  
{//code goes here}
```


- The button “*Clear Question*” should clear the values inside the text areas, text boxes and radio buttons. If the question has already been saved into the lists, it should be removed from the lists as well.

To complete this step, create and use a method with the following signature:

```
/* This method should remove the current question, answers,
and boolean values from the appropriate lists, if present.
*/

public void removeQuestionFromLists()

{ //code goes here }
```

- The button “*Commit to File*” should save the quiz (the contents of the *ArrayLists*) into the text file that the user chose before writing the quiz.

To complete this step, create and use a method with the following signature:

```
/* This method should go through the lists and build up a
String containing the quiz in the format specified below,
and then use the FileIO class to write the quiz to the text
file chosen by the user. */

public void saveAllQuestions()

{ //code goes here }
```

The text file that is written should have the following format:

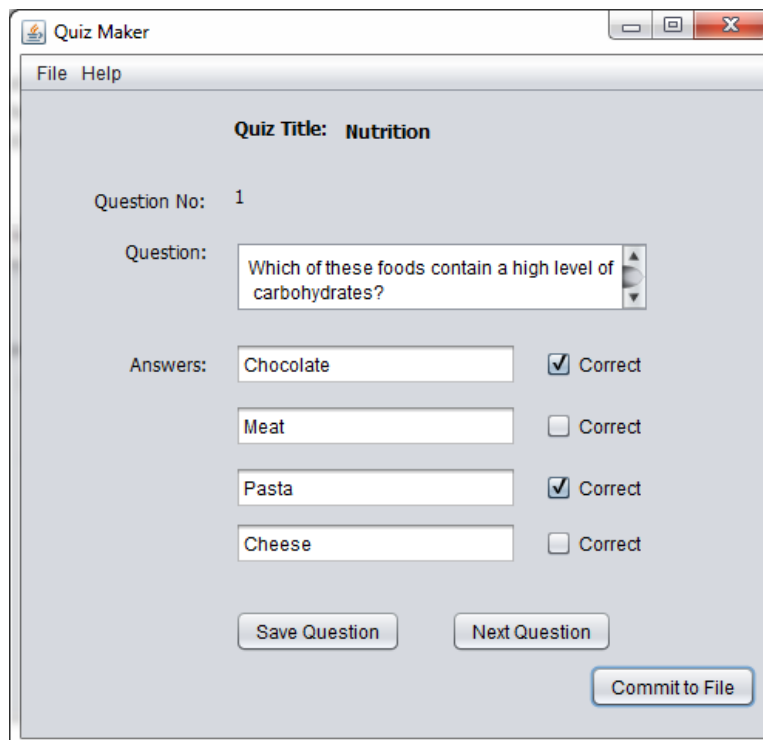
```
$<question1>
<true-or-false>,<answer-1>
<true-or-false>,<answer-2>
<true-or-false>,<answer-3>
<true-or-false>,<answer-4>
$<question2>
<true-or-false>,<answer-1>
<true-or-false>,<answer-2>
<true-or-false>,<answer-3>
<true-or-false>,<answer-4>
...
...
```

For example:

```
$Which of these foods contain a high level of carbohydrates?
true,Chocolate
false,Meat
true,Pasta
false,Cheese
$What is the function of proteins?
false,Regulates temperature
false,Provides acids
true,Promotes Growth
false,For eyesight
$Which of these are vitamins?
true,B12
true,C
false,Z
true,A
```

N.B. Note the \$ (dollar sign) at the start of each question and the , (comma) separating each answer from its boolean value. These are delimiters which will help when reading the file.

- vi) Clicking the *Open Quiz* item should cause the *File Open* dialog to open (use *FileIO* class). The user should choose the quiz file to open. After reading the quiz from the text file, the *ArrayLists* of the program should be populated, and the first question is displayed for editing.



The screenshot shows a window titled "Quiz Maker" with a menu bar containing "File" and "Help". The main area displays the following information:

- Quiz Title:** Nutrition
- Question No:** 1
- Question:** Which of these foods contain a high level of carbohydrates?
- Answers:**
 - Chocolate ☒ Correct
 - Meat ☐ Correct
 - Pasta ☒ Correct
 - Cheese ☐ Correct

At the bottom, there are three buttons: "Save Question", "Next Question", and "Commit to File".

To complete this step create and use methods with the following signatures:

```
/* This method should use the input string (containing the
contents of the text file) to populate the 3 ArrayLists. */
public void fillUpArrays(String fileContents)
{
    //code goes here
}

/* This method clears the current text in the form, gets the
appropriate entries from the lists depending on the question
number passed as parameter, and displays the question and
answers in the form, for editing. */
public void showQuestion(int i)
{
    //code goes here
}
```

- The user should be able to edit the question, the answers and the radio button selections.
- After editing, the user clicks on the “*Save Question*” button. The program should update the *ArrayLists* to reflect the change, and afterwards show the next question (or goes back to the 1st question if the current question is the last one).

To complete this step, create and use a method with the following signature:

```
/* This method should update the 3 ArrayLists to reflect the
editing done by the user in the form. */
public void updateListsWithCurrentQuestion()
{
    //code goes here
}
```

- Note that while editing a quiz, the “*Clear*” button, becomes “*Next Question*”. When the user clicks on this button, the next question is displayed (or the 1st question if the current question is the last one).
- Finally the button “*Commit to File*” should save the quiz (the contents of the *ArrayLists*) into the text file (overwrite the file that the user opened).

c) Quiz Machine

In this section you will develop the Quiz Machine. Where possible, make use of the *FileIO* class in order to avoid code duplication. You can also copy methods from the QuizPlayer. Create a new JFrame Form called *QuizMachine*.

- i) You need to declare and initialise 3 *ArrayLists* in your program, exactly the same as those used in the Quiz Maker. You also need a variable to hold the *score* (a whole number).
- ii) When the program is executed it should only display the menu bar, containing the options to *Play a Quiz*, *Quit*, and view information *About* the program.



- iii) Clicking the *Quit* menu item should cause the program to terminate.
- iv) Clicking the *About* menu item should open another *JFrame* showing information about the program. Closing the *About* window should **not** cause the program to terminate (the same instance of *JFrame* as the one used in Quiz Maker should be used).
- v) Clicking the *Play Quiz* item should cause the *File Open* dialog to open (use *FileIO* class). The user should choose the text file containing the quiz he wants to play.


The quiz is then loaded into the *ArrayLists*. To do this, copy and use the method called **fillUpArrays** (and modify it if required) in the Quiz Maker.

The GUI shown below should then be displayed, showing the 1st question and question number. To complete this step create and use a method with the following signature:

```
/* This method clears the current question (copy and use the
clearAll method created in Quiz Maker), gets the appropriate
entries from the lists depending on the question number passed
as parameter, and displays the question and answers in the form,
leaving all radio buttons unticked. This method also calculates
and displays the number of correct answers for the question, so
that the user is told how many answers to tick. */

public void showQuestion(int i)

{ //code goes here }
```



- The Quiz Title should automatically be set to the filename which the user would have chosen, without the extension.
- The user should try to answer the question by ticking one or more radio buttons.
- The button “*Next Question*” should check the user’s answer for the current question, and afterwards show the next question (if there are more questions left).

To complete this step, create and use a method with the following signature:

```
/* This method should check the radio button selection and
match it with the ArrayList containing the boolean values,
to check whether the user answered correctly. If the answer
is correct, the score is incremented. */

public void checkAnswer(int i) //i is the current question no.
{
    //code goes here
}
```

- When the user arrives to the last question, the text of the button should be changed to “*Score Now*” instead of “*Next Question*”.
- When the user finally clicks on the “*Score Now*” button, the score should be calculated as a floating-point percentage value (out of 100). The score is displayed in a message box, together with a comment, as follows:

Score Range	Comment
0-29	“You should read an idiot’s guide to “+<quiz name>
30-59	<quiz name>+” is not a new subject to you, but you can improve!”
60-79	“You’re on the way to becoming a “+<quiz name>+” pro!”
80-100	“Wow! You really are an expert in “+<quiz name>

To complete this step, create and use a method with the following signature:

```
/* This method should calculate the score as a percentage
and output the result, together with one of the comments as
shown in the table above, depending on the score. */

public void showScore()

{ //code goes here }
```

Marking divided as follows (34 marks):

Correctly adding GUI components to create Quiz Maker GUI	3 marks
Correctly adding GUI components to create Quiz Machine GUI	3 marks
Correctly creating the About JFrame	1 mark
Making GUI components appear programmatically	2 mark
Changing the contents of buttons programmatically	1 mark
Programmatically creating message dialogs	1 mark
Programmatically calling and showing the About form	1 mark
Programmatically creating and showing functional Save and Open File Dialogs	2 mark
Correctly declaring the required variables using appropriate data types	1 mark
Correctly saving the quiz in the specified format	1 mark
Correctly loading the quiz and populating ArrayLists as specified	1 mark
Correctly showing the quiz name according to file name	1 mark
Feature to delete/clear a question works correctly	1 mark
Feature to show, edit and re-save questions works correctly	1 mark
Answer is correctly checked to see whether it is correct or not, and score is correct	1 mark
Mapping of score to comment using if statement is correct	1 mark
Program works perfectly, meeting all requirements, with good modularity	2 mark
The method which writes to a text file contains correct (specific type) exception handling	2 marks
The method which reads from a text file contains correct (specific type) exception handling	2 marks
The catch blocks contain friendly messages displayed in message boxes	1 mark
Correctly validating a new question while creating a quiz (with specific error messages)	2 marks
In the Quiz Maker, clicking Next while on the last question takes you back to 1 st question	2 mark
The File Open dialog includes a functional text file filter	1 mark

TASK 2 – Commenting and Debugging

KU3.3 - Describe the importance of adding comments to the code

KU4.1 - Describe how to debug an application

10 marks

- a) Comment the programs that you completed in Task 1. (3 marks)
- Comments should be consistently written throughout the classes and methods to explain what is happening. Use both in-line comments and descriptive blocks.
 - Most of the lines, which is not obviously trivial (such as a *println()* statement) should be commented.
 - Before each method signature, write a comment to explain what the method is doing, and what inputs and outputs it has.
- b) Mention and explain 2 advantages of commenting your code. (2 marks)
- c) *“NetBeans IDE provides a rich environment for troubleshooting and optimizing your applications. Built-in debugging support allows you to step through your code incrementally and monitor aspects of the running application, such as values of variables, the current sequence of method calls, etc...” (NetBeans User’s Guide)*
- i) Mention and explain 2 disadvantages of using *System.out.println* statements to debug your code. (2 marks)
- ii) Explain how you can do the following debugging activities in NetBeans. For each, provide at least 1 screenshot of how you applied it to debug the program you completed in Task 1.
- Using Breakpoints (1 mark)
 - Stepping through the Code (1 mark)
 - Monitoring Variables (1 mark)

TASK 3 – Exception Handling*AA4.1 - Demonstrate the basic concepts of what exceptions are***7 marks**

- a) Consider the Quiz Maker program created in Task 2. Assume that the following sequence of actions happen, and then answer the questions below.

Step 1 - The user runs the Quiz Maker

Step 2 – The user clicks File...New Quiz

Step 3 - The Open File Dialog opens, but the user clicks Cancel

Step 4 - The user-defined method showSaveDialog returns an empty String

Step 5 - The method getQuizNameFromPath finds the index of the . (dot) character in the file path

Step 6 - The method getQuizNameFromPath gets a substring starting from the index of the . (dot) character, until the last character of the file path

- i) Which step would cause an exception to be thrown? (1 mark)
 - ii) What is the name of the exception that is thrown? (1 mark)
 - iii) Is this type of exception checked or unchecked, and how do you know? (2 marks)
 - iv) How can the problem be solved? (1 mark)
- b) Once again consider the Quiz Maker program created in Task 2. Look at the code below, and assume that the *questions* is an *ArrayList* of size 2.

```
for (int i = 0; i < questions.size(); i++)    //line 1
{
    System.out.println(questions.get(i));    //line 2
}
```

- i) Which line would cause an exception to be thrown? (1 mark)
- ii) What is the name of the exception that is thrown? (1 mark)

-----End of Assignment-----