Malta College of Arts, Science & Technology

INSTITUTE OF INFORMATION AND COMMUNICATION TECHNOLOGY

Test Driven Development

BSc. (Hons) Software Development

Assignment 2 (Home)

TDD Project Part 1

Instructions to Students

Read the following instructions carefully before you start the assignment. If you do not understand any of them, ask your lecturer for further explanation.

- This assignment has a total weight of **34%**.
- Copying is strictly prohibited and will be penalised through a referral and other disciplinary procedures as per MCAST Policies, Procedures and Regulations.
- Make sure to write your **name, ID card number and group** on the cover sheet of the assignment.
- Your submission should include a:
  - Printed version of this assignment brief and your work (except the code)
  - CD containing the code and a softcopy of all the printed files.
  - Submission to TurnItIn as instructed by your lecturer.
- Each task shows the marks associated with it. A summary can be found on the front coversheet.
- An individual interview will be held upon assignment submission. You are expected to be able to explain your choices and code in specific tasks in order to achieve the marks associated with each task.

Scenario description

Your task is to create a Toolbox that contains the following structure (you need to add method return values, method bodies and any other code required).

```
public class ToolBox {
        Command cmd;
        ToolBox(List<String> params)
        execCommand()
        getCommandOutcome()
}
```

The **ToolBox** constructor takes a list of strings. The first string (index 0) is the name of the command. The other (optional) strings are to be used by the command if needed.

The toolbox is used to access Commands, with the snippet below showing typical usage of a Command called `SendWelcomeEmail` that takes two parameters; an email address and a subject.

```
List<String> params = Arrays.asList("SendWelcomeEmail", "user@mail.ru", "Subject");
Toolbox tb = new Toolbox(params);
tb.execCommand();
tb.getCommandOutcome();
```

When invalid input (null list, empty list or unknown command) is given to the constructor, a placeholder command should be created that always fails on execution.

Command is an abstraction (abstract class or interface), and you are to use the Factory Design Pattern by implementing the class below and using it in the ToolBox constructor.

```
public class CommandFactory {
        public static Command parseCommand(List<String> params)
}
```

You need to implement three commands of your choice and add enough to the skeleton provided above to be able to run the ToolBox. The requirements for the three commands are found below:

1. A command that results in the creation or modification of a file
   e.g. appending a line to a text file (creating it if it does not exist)
2. Another command that connects to the internet
   e.g. download a file, check if a server is responding, etc.
3. A final command that performs some notification
   e.g. send an SMS or email.

Notes

- If you need to hardcode email credentials or API keys, create dummy accounts.
- It is recommended that you discuss your ideas for the commands with your lecturer before starting the development.

Part 1: Development, unit & integration testing (34 marks)

4.1 Scenario implementation:

- ToolBox implementation according to specification. (1 mark)
- Proper implementation of abstraction (Command) and factory design pattern. (1.5 marks)
- Implementation of the three commands. (4.5 marks)

4.2 Your scenario should include unit tests that achieve the following code coverage:

- 100% method coverage on all of the production code. (2 marks)
- 90+% statement coverage on all of the production code. (2 marks)
- 80+% branch coverage on your three commands. (6 marks, 2 for each command)

3.3 Integration testing:

- Correctly use test stubs for the 3 dummy commands. (6 marks, 2 for each command)
- Correctly use a driver to test all your implemented commands. (4 marks)

1.2 Test driven development:

- In not more than one page including diagrams, describe how you used the test driven development approach in implementing the scenario. (3 marks)
- Answer interview questions about your test driven development project. (4 marks)

***