Test Driven Development

B.Sc. (Hons) Software Development

Assignment 2 Part 1

Test Driven Development Project (Home)

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 (Part 1) 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.
- Your submission should include a:
    - Printed version of this assignment brief (3 pages)
    - Printed and signed coversheet.
    - Submission to Moodle as instructed by your lecturer.
- Each task shows the marks associated with it. A rubric with a detailed breakdown can be found on the last page of this assignment.
- An individual interview will be held upon assignment submission. You are expected to be able to explain your choices and code in specific tasks to achieve the marks associated with each task and to the interview itself.

Create an abstract class called **Task** with the same methods as below:

```
import java.util.List;

public abstract class Task {
    public abstract boolean configure(List<String> params);
    public abstract String run();
}
```

Task is an abstract class and you are to extend it when developing your two tasks.
The **configure()** method takes a list of parameters used by the particular task as needed. It returns true if the list of parameters is a valid one.
The **run()** method performs the Task and returns a String representing the outcome of the task.

The snippet below shows typical usage of an implementation of Task called **EmailTask.**
It is configured by passing 3 parameters: an email address, a subject and a body.
Running EmailTask sends the email requested.

```
List<String> params = Arrays.asList("john@doefamily.name", "Hello", "Long time...");
Task mailer = new EmailTask();
if (mailer.configure(params)) {
    String emailOutcome = mailer.run();
    System.out.println(emailOutcome);
}
```

You need to implement two tasks of your choice. One of the tasks must use the internet in some way while the other should use a local file. Below is a list of possible Tasks:

- **FlagTask**: Generates a simple flag (e.g. Germany, Italy, Russia) of a given size
- **LogFileTask**: Opens a log file and reads some information from it, e.g. latest entry
- **FileSizeTask**: Checks the size of a given file on the HDD
- **YourFileTask**: Interacts in some way with the file system to read from or write to a file
- **WeatherTask**: Uses a weather forecast web service and gets the weather details for a city
- **ForexTask**: Uses a foreign exchange web service to retrieve the rates for a given currency
- **NotificationTask**: Send an SMS (e.g. Twilio) or email to a recipient
- **TwitterTask**: Return a collection of the most recent Tweets posted by a **Twitter** user.
- **YourWebTask**: Interacts with a web service or scrapes a website to retrieve some information or perform a web-related task

Notes:

- It is important to discuss your task ideas with your lecturer and proceed with development after you get the go-ahead on a design.
- If you need to hardcode credentials, tokens or API keys, create dummy accounts.
- For a list of web services refer to sites like: https://programmableweb.com/
- Your implementation of configure() should handle all the cases of invalid input e.g. null list, empty list, missing parameters, etc.

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

AA4 Scenario implementation:

- Proper use of OOP techniques. (1 mark)
- Implementation of two Tasks. (6 marks, 3 marks for each task)

SE3 Create unit tests that achieve the following code coverage on your tasks:

- 90+% statement coverage on all the production code. (5 marks)
- 80+% branch coverage on your two Tasks. (5 marks)

SE2 Integration testing:

- Create test stubs for the tasks to test all the outcomes (success and different fails) of each task without external dependencies (internet and HDD). (6 marks, 3 for each task)
- Create a driver to test the real implementation (real files and connections to the internet) of your tasks. (4 marks)

AA1 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)

Home Assignment Part 1 Rubric

Task	No Marks	Partial Marks	Full Marks	Marks
AA4	No implementation, code does not compile or crashes and cannot be tested.	Half marks awarded if the Tasks are partially functional or not according to specifications.	Tasks are fully functional and according to specifications.	
SE3	No Tests implemented, or test do not compile and run.	Statement coverage: 70% - 89% (3 Marks) Branch coverage: 60% - 79% (3 Marks)	Statement coverage: 90+% Branch coverage: 80+%	
SE2	No Driver/Stub implemented, or tests using them do not compile and run.	Task stubs attempted but incomplete or do not mock all dependencies (1 Mark per task) Driver attempted but incomplete (2 Marks)	Driver and task stubs are complete and achieve their goal.	
AA1	No Documentation submitted. Interview not attended or failed to answer most questions correctly.	Documentation vague, incomplete, or incorrect in parts (0.5 Mark per Task) Some answers to interview questions not satisfactory. (2 Marks)	Documentation complete and correct. Implementation complete and mostly correct. All answers to interview questions satisfactory.	