Software Test Automation

B.Sc. (Hons) in Software Development
B.Sc. (Hons) in Multimedia Software Development

Home Assignment 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 Home Assignment (Part 1) has a total weight of **29%**
- Plagiarism is strictly prohibited and will be penalised through a referral and other disciplinary procedures as per MCAST Policies, Procedures and Regulations.
- Submit through Moodle (documentation and code).
- Each criterion 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 may be held after assignment submission where you would be expected to explain your choices and code in specific tasks to achieve the marks associated with each task.

You will be given a Maven project with an abstract class called **Task** with the methods below:

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

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 (what is valid depends on task's requirements). The **run()** method performs the task and returns a String representing the outcome of the task.

The Eclipse project also includes the implementation of a reference Task called **WeatherTask.** It is configured by passing 1 parameter: a name of a city. Running **Demo** class creates the object, gets the weather forecast from a web service and prints out information about the difference between the maximum and minimum temperatures for today.

Note: you need an API key to use the web service, get one from: https://openweathermap.org/appid

---

Your assignment is 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. Both tasks should involve implementing some logic and not just retrieve data and display it as-is.

Notes:

- It is important to discuss your task ideas with your lecturer and proceed with development only **after** you get the go-ahead.
- Create temporary accounts if you need to hard-code login credentials, tokens, or API keys. *Note: It might be simpler if you avoid OAuth logins.*
- For a list of web services refer to websites like:
  - programmableweb.com/category/all/apis
  - rapidapi.com/marketplace
  - developers.google.com/apis-explorer
  - github.com/n0shake/Public-APIs
- Your implementation of `configure()` should handle all the cases of invalid input e.g. null list, empty list, missing or invalid parameters, etc.

Below is a list of possible file-related tasks:

Name	Parameters	Logic	Output example
Flag	Country name Width (pixels) Height (pixels) e.g. France, 250, 200	Creates a PNG image which is of the given width and height. The image contents will depend on the country name.	france.png created
WordCount	Textfile-path e.g. res/bible.txt	Opens the given text file and counts all the words inside it	783137 words in res/bible.txt
LogFile	Logfile-path e.g. res/events.log	Opens the given log file and checks if the last line contains an INFO, ERROR or WARNING entry	Last entry in res/events.log: Warning
FileSize	File-path e.g. files/docs.zip	Checks the file size and returns a description as follows: < 100KB: Tiny < 1MB: Small < 100MB: Medium < 1GB: Large >= 1GB: Huge	Size of files/docs.zip: Medium
Other	File-Path	Interacts in some way with the file system to read from or write to a file	Result of the logic that took place

Below is a list of possible web-related tasks:

Name	Parameters	Logic	Output example
Hollywood	Film name e.g. Joker	Uses a web service to get rating of a film or TV series and returns a string describing the rating: < 4/10: Bad < 7/10: Decent < 10/10: Good 10/10: Awesome	Joker is a Good film
Forex	Amount From Currency To Currency e.g. 100, GBP, EUR	Uses a web service to retrieve the rate for a given currencies and calculates the amount offline	100GBP = 117.78 EUR
Sport Odds	Team name (or ID) e.g. Barcelona	Use a webservice to get the odds of the team winning the next fixture and returns a descriptive string: < 1.5: Easy Game 1.5…2: Tight Game > 2: Hard Game	Odds: Easy Game
Twitter	Twitter handle Number of tweets e.g. @FranckRibery, 5	Gets a collection of the most recent number of Tweets posted by a Twitter user, and returns a string indicating the average length: < 25 characters: Short tweets >= 25 characters: Long tweet	FranckRibery likes Long tweets
Other (excluding Weather)		Interacts with a web service or scrapes a website to retrieve some information or perform a web-related task	Result of the logic that took place

Part 1: Development, unit testing & integration testing (29 marks)

AA1 Test driven development:

- In a short document, describe how you used the TDD approach when implementing the scenario. Mention the major challenges encountered. (3 marks)
- For both of your tasks document the following (4 marks):
  - Description of the task and logic, including Web service (URL) and/or libraries used
  - Inputs required and validation rules
  - All possible outputs
  - Sample of inputs and relative output.

AA4 Scenario implementation:

- Make proper use of OOP techniques. (1 mark)
- Implement the two Tasks in a Java project with correct Maven dependencies. (6 marks, 3 marks for each task)

SE3 Create unit tests that achieve the following code coverage on all the production code:

- 90+% instruction coverage. (5 marks)
- 80+% branch coverage. (5 marks)

KU5 Integration testing:

- Create test stubs (or mock objects) for the tasks to test all the outcomes (success and different failures) of each task without any external dependencies (Internet and HDD). (5 marks, 2.5 for each task)

Home Assignment Part 1 Rubric

Task	No Marks	Partial Marks	Full Marks	Marks
AA1	No Documentation submitted.	Documentation vague, incomplete, or incorrect in parts (1 – 6 marks)	Documentation complete and correct.	
AA4	No implementation, code does not compile or crashes and cannot be tested.	Partial marks awarded if the Tasks are not fully functional or not according to specifications. (1 – 6 marks)	Tasks are fully functional and according to specifications.	
SE3	No Tests implemented, or test do not compile and run.	Statement coverage: 70% - 89% (2.5 marks) Branch coverage: 60% - 79% (2.5 marks)	Instruction coverage: 90+% Branch coverage: 80+%	
KU5	No Stub/Mocking implemented, or tests using them do not compile and run.	Task stubs incomplete or do not mock all dependencies (1 mark per task)	Task stubs/mocks are complete and achieve their goal.	