CPS3230 – Assignment 1 Chakotay Incorvaia 358199(M)

Fundamentals of Software Testing

Semester 1

2019/2020

B.Sc.IT(Hons) Software Development



Contents

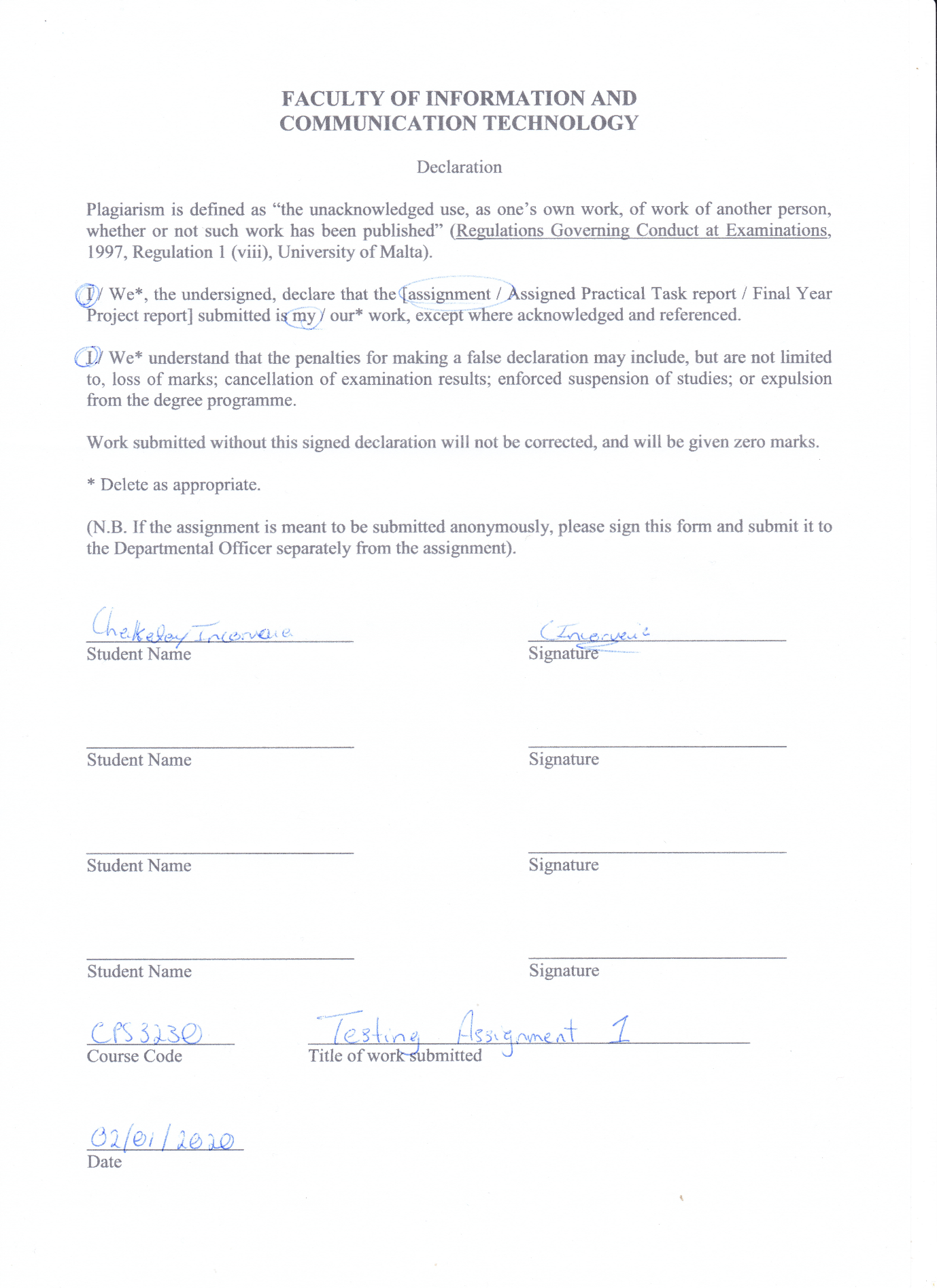
[Plagiarism Form: 3](#_Toc28880594)

[Assignment Github Page: 4](#_Toc28880595)

[Task 2: 4](#_Toc28880596)

[Task 3: 5](#_Toc28880597)

# Plagiarism Form:



# Assignment Github Page:

The examiner should be already added: <https://github.com/CHAKOTAY99/CPS3230_Assignment_1>

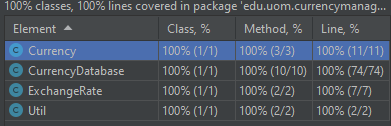
The project contains two main branches **master** and **Task3**. The branch **master** contains all the code for Task 1 and Task 2. The branch **Task3** contains code amended from Task 2 for Task 3.

Tags have been created for both tasks by **v2.0** and **v3.0**. **These tags may be updated so please download the latest version.**

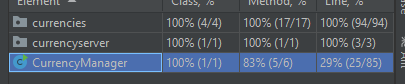
# Task 2:

The following is the coverage for the entire project of Task 2:









The reason that 100% coverage is not possible is because the main method cannot be covered by unit tests. At most the main method could be reduced in size to achieve greater code line coverage. Otherwise the entire class has been covered.

# Task 3:

**This task has not been finished due to the time constrains and the required time investment. This assignment exceeded the amount of estimated hours of 20 to 25 hours. It was no longer feasible to continue the assignment after 50 hours were exceeded including research and development.**

The plan to change the SUT was as follows:

* Increase the separation of concerns in order to increase testability. Classes such as **ExchangeRate** would be tasked with handling all operations with regards to rate. This required to move functionality from **CurrencyDatabase** and **CurrencyManager**.
* Interfaces were added to each feasible class to increase the inversion of control. This made the classes more testable and allowed the ability to mock other classes and their results.
* Place read and writing operations to file in a separate class to be called when they were required.
* Operations done in the main class would have been moved into separate classes or placed inside different existing classes. This would greatly have reduced the lines of code in the main class.

Together with traditional methods of applying dependency injection, it was hoped that the following would have greatly increased testability although code coverage would have remained the same.

The following test Patterns were planned:

* **Factory Pattern** would have been implemented for each currency. The list of currencies would have been tied to only one database.
* The database would have been a **singleton pattern** as it would be assumed that only one database would have been allowed to exist at a time.

Sources on dependency lookup are few and far between. A lot of online resources indicated that their use are mutually exclusive – only dependency injection or dependency lookup could have been used.

## Actual Milestones Achieved

Very little of the code is operational of Task 3 is operational unfortunately. Whilst the user could access all functions, functionality related to the **ExchangeRate** could not be fixed. Their respective test cases were not developed for reasons outline above. **The development of Task 3 was greatly flawed due to no use of TTD.** I am certain that should time have been permitting, task completion would have been attained.