



## **SENG 275 ASSIGNMENT 2(10%)-O2**

Instructor: Dr. Navneet Kaur Popli

Date: 28 Feb 2023

Submit by: 07 Mar 2023

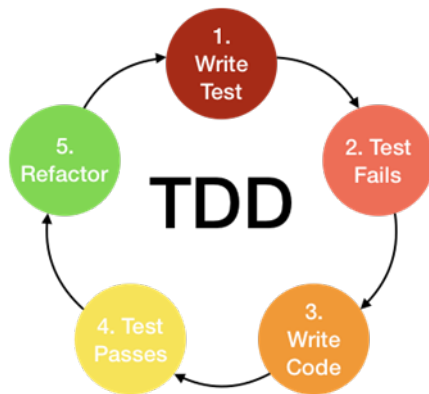
Total Marks: 85

Syllabus: TDD, Mutation testing

This is an individual activity. Copying, discussions with peers or cheating of any kind is not allowed. Plagiarised content is prohibited. Provide suitable references.

Q1) Work with a TDD kata called “AddMyAlphas”. (Total=25M)

This Kata is designed to help practice how a testing is done using the TDD approach. It is intentionally designed to start with a very easy, non-branching base case which slowly becomes added with complexity as additional requirements are added that will require significant pressure to compose additional units.



Before you start remember the TDD approach:

- Add a test
- Run all tests and see if the new one fails
- Write some code
- Run tests
- Refactor code
- Repeat




University  
of Victoria

## Kata Steps

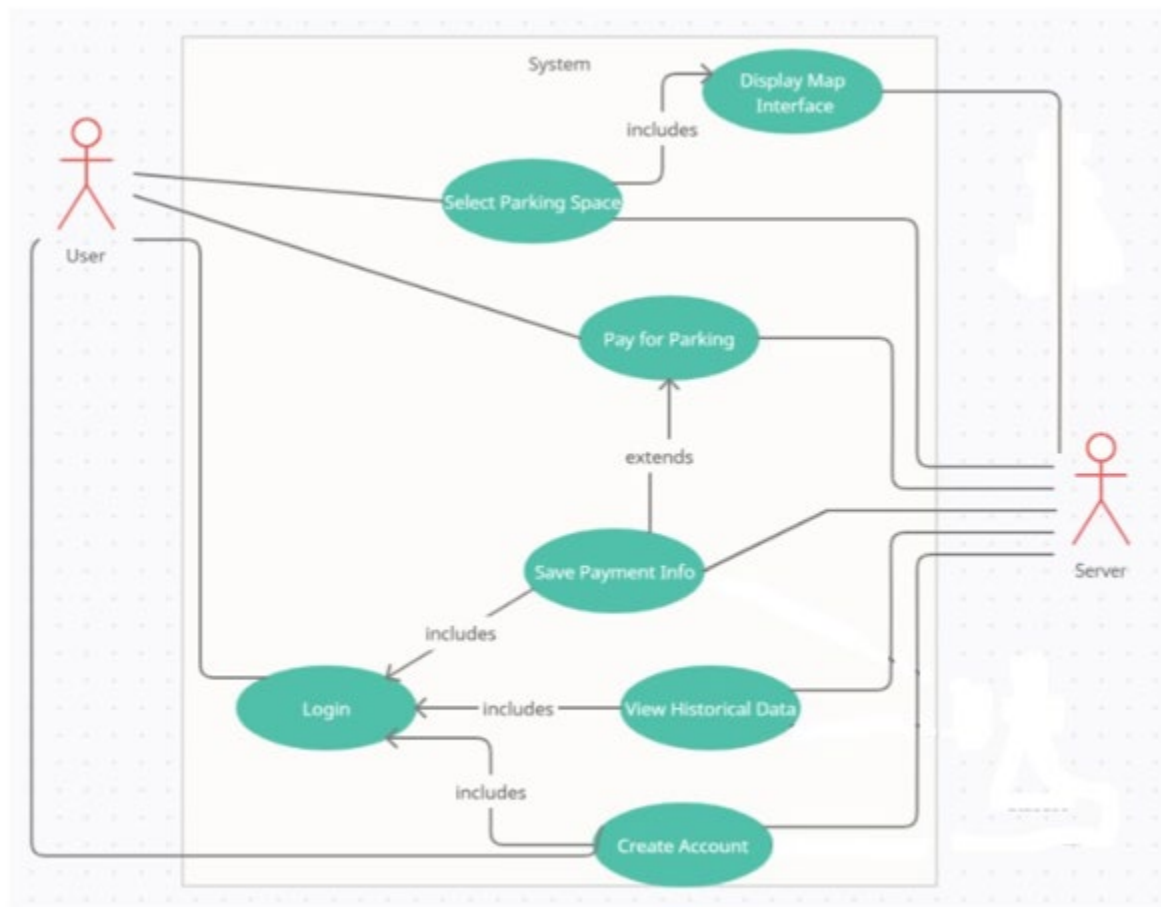
1. In a test-first manner, create a simple class called “AddMyAlphas” with a method `public int Add (String numbers)` (5M)
  - i) The method can take 0, 1, or 2 numbers and will return their sum.
  - ii) For an empty string it will return 0.
  - iii) For example: “” == 0 , “1” == 1 , “1,2” == 3
  - iv) Start with the simplest test case of an empty string then move to one and two numbers.
  - v) Remember to solve things as simply as possible, forcing yourself to write tests for things you did not think about.
  - vi) Remember to refactor after each passing test. Show the refactoring steps. (5M)
2. Allow the Add method to handle an unknown number of arguments/numbers. (2M)
3. Allow the Add method to handle new lines between numbers (instead of commas). (2M)
  - i) The following input is ok: “1\n2, 3” should return 6.
  - ii) The following is an invalid input: “1, \n” is invalid, but you don’t need a test for this case.
  - iii) Only test correct inputs. There is no need to deal with invalid inputs for this kata.
4. Calling Add with a negative number will throw an exception “Negatives not allowed: “ listing all negative numbers that were in the list of numbers. (2M)
  - i) Example “-1, 2” throws “Negatives not allowed: -1”
  - ii) Example “2, -4, 3, -5” throws “Negatives not allowed: -4,-5”
5. Numbers bigger than 1000 should be ignored. Example: “1001, 2” returns 2. (2M)
6. Allow the Add method to support different delimiters: (2M)
  - i) To change the delimiter, the beginning of the string will contain a separate line that looks like this: “//[delimiter]\n[numbers]”. For example: “//;\n1;2” should return 3 (the delimiter is ;)
  - ii) All existing scenarios and tests (using , or \n) should still be supported and should work as before. (5M)

Submit detailed tests, code and complete execution and all test results with appropriate explanations for each revision while following the TDD approach as a series of steps as pdfs.

Q2) An optimized parking application called 'Park Right' is being developed as an extension to the already existing parking system in the area where users find an empty place to park in the street and then pay for the parking spot. The perspective for the solution is listed below:

- The product will assist in the current method of parking by helping users in identifying available parking spaces before entering the intended lot.
  - The product will communicate to the user the lot status of their desired location.
  - The product will work in tandem with the current parking management system, ie. street meters, ticker managed parkades, and ticket managed lots.
  - The product will expedite the process of locating available spaces in parking lots by notifying users of the number and location of available spots through a mobile application.
- 

Find below the use case diagram for the entire application.



One of the use case of the application is the 'Pay for Parking System'. This payment system in the application will allow users to pay for parking spaces. This system will provide an all-in-one parking experience for the users. Thus allowing them to pay for their parking spot without having to step outside their car.

Find below the use case specification for this use case.

| UC-2-3: Pay for a Parking Space                                                                                                                                                                                                                                                                                                                                                                       |
|-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| ID: UC-2-3                                                                                                                                                                                                                                                                                                                                                                                            |
| Description: A user selects and pays for a parking space.                                                                                                                                                                                                                                                                                                                                             |
| Actors: User, Server                                                                                                                                                                                                                                                                                                                                                                                  |
| Secondary Use Case:<br>1. UC-2-4: Save Payment Information                                                                                                                                                                                                                                                                                                                                            |
| Preconditions:<br>1. The user has logged in.                                                                                                                                                                                                                                                                                                                                                          |
| Main Flow:<br>1. The user is prompted to enter a parking space ID.<br>2. The user selects a method of payment.<br>3a. If the method is valid:<br>i. User is prompted to save payment information if desired.<br>ii. User selects to save his payment Information.<br>iii. Payment information is saved.<br>4. The user completes the payment.<br>5. Server sends payment receipt to the user's email. |
| Postconditions:<br>1. The user is parked and registered at the relevant parking space ID.                                                                                                                                                                                                                                                                                                             |
| Alternative Flow(s):<br>3b. If the payment method is invalid:<br>i. Application returns and displays an invalid payment message.<br>ii. The user is prompted to select a different method of payment (Continue from 2).                                                                                                                                                                               |

- Create an activity diagram for this use case. (10M)
- Create a graph from this activity diagram. (10M)
- Generate test cases from this activity diagram which exercise the entire use case. Submit as pdf. (10M)

Q3) You now have an assignment2.git repository at [gitlab.csc.uvic.ca](https://gitlab.csc.uvic.ca). You may use it as the basis for your solution to this problem. Do not submit your work for this question through gitlab; your solution is to be provided in your pdf with the answers to the other questions as shown below.

In `lib/src/main/java/a2`, you will find a function named `a2/permute()`. This function is reproduced below:

```
public String permute(final int a, final int b, final int c, final int d) {  
    if (a > 10) {  
        int x = 1;  
        if ( b < a && c >= d) {  
            return "One";  
        } else {  
            int y = 2;  
            if (c <= 20) {  
                return "Two";  
            }  
            return "Three";  
        }  
    }  
    return "Four";  
}
```

Your task is to write tests against this function such that you achieve both 100% branch coverage and a clean mutation testing report through PiTest. You must achieve 100% branch coverage with your tests before you will be able to run the PiTest mutation Tester.

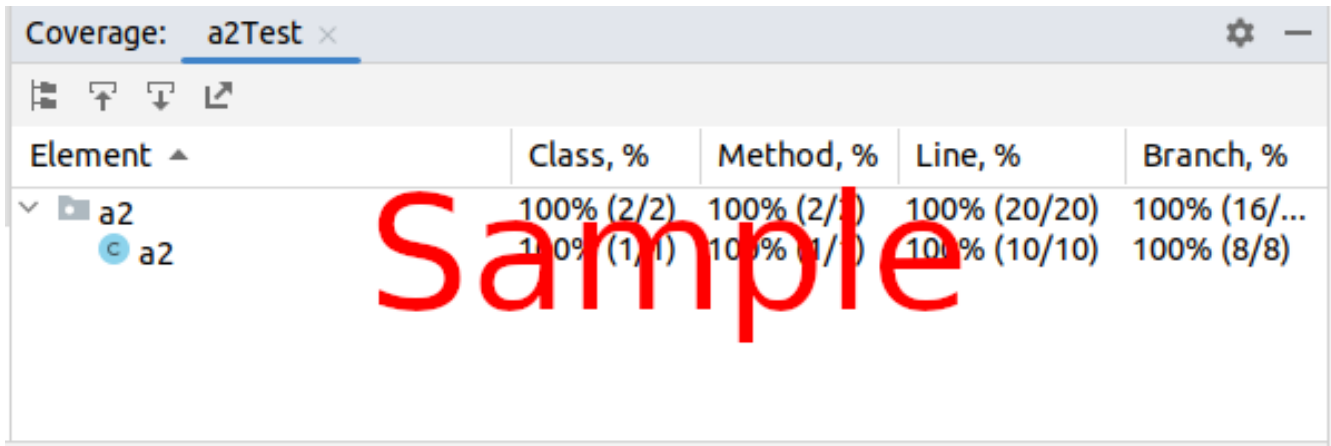
You have been provided a starting file (in `lib/src/test/java/a2/a2Test.java`) which you may use as a starting point for your tests. The provided test function will call a parameter source method named `generate()`, which you must write.

Refer to Lab04 for more information on obtaining coverage statistics and mutation testing, and for help locating the mutation testing report.

For full marks, you must submit the following items:

1. A snapshot of your test code. (10M)
2. A snapshot showing your coverage statistics, including branch coverage (example shown below)(10M)

3. A snapshot showing your mutation testing report (example shown below)(10M)



| Element ▲ | Class, %   | Method, %  | Line, %      | Branch, %    |
|-----------|------------|------------|--------------|--------------|
| ▼ a2      | 100% (2/2) | 100% (2/2) | 100% (20/20) | 100% (16/... |
| a2        | 100% (1/1) | 100% (1/1) | 100% (10/10) | 100% (8/8)   |

## Mutations

|                    |                                                                                                                                                            |
|--------------------|------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <a href="#">23</a> | 1. changed conditional boundary → KILLED<br>2. negated conditional → KILLED                                                                                |
| <a href="#">25</a> | 1. changed conditional boundary → KILLED<br>2. changed conditional boundary → KILLED<br>3. negated conditional → KILLED<br>4. negated conditional → KILLED |
| <a href="#">26</a> | 1. replaced return value with "" for a2/a2::permute → KILLED                                                                                               |
| <a href="#">29</a> | 1. changed conditional boundary → KILLED<br>2. negated conditional → KILLED                                                                                |
| <a href="#">30</a> | 1. replaced return value with "" for a2/a2::permute → KILLED                                                                                               |
| <a href="#">32</a> | 1. replaced return value with "" for a2/a2::permute → KILLED                                                                                               |
| <a href="#">35</a> | 1. replaced return value with "" for a2/a2::permute → KILLED                                                                                               |

Sample