|  | Munster Technological University    **Software Testing**  **CA 1 (40%)** |
| --- | --- |

**Due Date: 1pm on 22nd March 2024.**

# **Question 1.**

**INSTRUCTIONS**

**The Employee Class.**

The Employee class accepts an employee’s name, age, and number of years of experience. The following rules apply,

The name must not be more than 40 chars long.

The age must be between 18 and 62 years.

The years of experience must be greater than or equal to 3 years.

The employee’s salary is then calculated using the follow rules.

|  |  |  |
| --- | --- | --- |
| **Age** | **Years experience (YE)** | **Salary** |
| < 40 | <= 6 | 30,000 + (100 \* YE) |
| < 40 | > 6 | 31,000 + (100 \* YE) |
| >= 40 | <= 6 | 33,000 + (150 \* YE) |
| >=40 | > 6 | 34,000 + (150 \* YE) |

**Part A (50%)**

Applying **Specification testing techniques** identify the **test conditions** and the **test cases** required to test the employee class. Show all working clearly.

# **Solution (Part A)**

## **Equivalence Partitioning**

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **Condition** | **Valid Partitioning** | **Tag** | **Invalid**  **Partitioning** | **Tag** | **Valid Boundaries** | **Tag** | **Invalid Boundaries** | **Tag** |
| **Name** | 1 to 40 | V1 | 0 chars | X1 | 1 char | B1 | 0 char | D1 |
| Valid chars | V2 | 41 chars | X2 | 40 chars | B2 | 41 chars | D2 |
| Invalid chars | X3 |
| **Age** | 18 to 62 | V3 | 17 | X4 | 18 | B3 | 17 | D3 |
| Valid digits | V4 | 63 | X5 | 62 | B4 | 63 | D4 |
| Invalid digits | X6 |
| **Experience** | >= 3 | V5 | < 3 | X7 | 3 | B5 | 2 | D5 |
| Valid digits | V6 | Invalid digits | X8 | ? |  | ? |  |

**Valid chars:** unclear from instructions.

**Valid digits:** unclear from instructions.

**Invalid chars/digits:** null, any other.

The instructions do not properly say:

* What "name" refers to.
* What is an "invalid start boundary" and what are "valid characters (such as "space", a-z, A-Z, etc.)" for testing. For the sake of this example, let's assume that for testing, we take a range of 1 to 40 characters, including spaces.
* For "age", it is not clear if 18 and 62 are included or excluded. Let's assume that for testing, we include both 18 and 62.
* For "experience", it is not clear what the "invalid end boundary" is.

Because of these conditions, the testing of the Employee class will not be fully finished.

## **Conditions and Value**

**Conditions:** 3

**Value/Test Cases:** 2\*2\*2 = 8

Name: Valid, Invalid (2)

Age: Valid, Invalid (2)

Experience: Valid, Invalid (2)

|  |  |  |
| --- | --- | --- |
| **Test Tag** | **Input** | **Expected Output** |
| X1, X2  D1, D2 | 1. “” 2. “Abcdefghigklmnopqrstlmnopqrrergrgbhrdbgbgldbbhrjbjhr” | Throw exception message |
| V1, V2  B1, B2 | 1. "Anna" 2. "A A A" 3. "anna" 4. "ANNA" | All valid |
| X4  D3, D4 | 1. 17 2. 0 3. 63 | Throw exception message |
| V3  B3, B4 | 1. 19 2. 44 3. 60 | All Valid |
| X7  D5 | 1. 1 2. 0 3. -2 | Throw exception message |
| V5  B5 | 1. 3 2. 6 3. 22 | All Valid |

Because of unclear instructions, I cannot test all possible inputs that should be tested for the Employee class.

## **Decision Table**

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  | **T1** | **T2** | **T3** | **T4** | **T5** | **T6** | **T7** | **T8** |
| **Name** | V | In | V | In | V | In | V | In |
| **Age** | V | V | In | In | V | V | In | In |
| **Exp.** | V | V | V | V | In | In | In | In |
| **Valid?** | T | F | F | F | F | F | F | F |

**V – valid**

**In - invalid**

According to the decision table, I can make the following conclusions: I can test T1, T3, T5, and T7. These tests should cover all possible scenarios for the Employee class. If the name is invalid, it will become False (invalid), and the salary will not be calculated. However, I will still include T2 in my testing to ensure that the invalid name is handled correctly.

## **State Transition Diagram**

S3

Validate Exp.

S4

Calculate Salary

enter valid exp.

enter valid name

S7

Invalid exp.

S0

Start

S1

Validate name

S2

Validate Age

S5

Invalid name

S6

Invalid age

enter valid name

enter valid age

Entered invalid name

Entered invalid age

Entered invalid exp.

Return

Return

Return

|  |  |  |
| --- | --- | --- |
| **Main Success Scenario**  **U: User**  **S: System** | **Step** | **Description** |
| 1 | U: Enter Name |
| 2 | S: Validate name |
| 3 | U: Enter age |
| 4 | S: Validate age |
| 5 | U: Enter experience |
| 6 | S: Validate experience |
| 7 | S: Calculate Salary |
| Extensions | 2a | Name not valid |
| Display message |
| 4a | Age not valid |
| Display message |
| 6a | Experience not valid |
| Display message |

## **Use Cases**

|  |  |  |
| --- | --- | --- |
| **Use Case** | **Input** | **Expected output** |
| T1  (valid) | Name: “Anna Kovalenko”  Age: 19  Exp.: 4 | Valid  Valid  Valid  Calculate Salary: Yes |
| T3  (invalid) | Name: “Anna Kovalenko””  Age: 15  Exp.: 6 | Valid  Invalid  Valid  Calculate Salary: No |
| T5  (invalid) | Name: “Anna Kovalenko””  Age: 25  Exp.: 2 | Valid  Valid  Invalid  Calculate Salary: No |
| T7  (invalid) | Name: “Anna Kovalenko””  Age: 64  Exp.: 1 | Valid  Invalid  Invalid  Calculate Salary: No |
| T2  (invalid) | Name: “”  Age: 30  Exp.: 5 | Invalid  Valid  Valid  Calculate salary: No |
| T9  (valid) | Name: “Anna Kovalenko”  Age: 19  Exp.: 4 | Salary: 30,000+(100\*4) = 30,400 |
| T10  (valid) | Name: “Anna Kovalenko”  Age: 30  Exp.: 9 | Salary: 31,000+(100\*9) = 31,900 |
| T11  (valid) | Name: “Anna Kovalenko”  Age: 45  Exp.: 5 | Salary: 33,000+(150\*5) = 33,750 |
| T12  (valid) | Name: “Anna Kovalenko”  Age: 62  Exp.: 10 | Salary: 34,000+(150\*10) = 35,500 |

## **Other Tests**

Tests that required by Part B

|  |  |  |
| --- | --- | --- |
| Test | Input | Excepted output |
| Constructor | 1.Not NULL  2.“Anna Kovalenko”  19  4  30400 | Constructor Tested  Valid |
| public GetName() | “Anna Kovalenko” | Valid |
| public GetAge() | 19 | Valid |
| public GetExperience | 4 | Valid |
| Set age 55 | “Dasha”  20  5  Set age: 55 | Age changed to 55. Print message. |
| Print Employee | “Kate”  21  4  30400 | Valid |

Total Tests: 21

For my tests JaCoCo is reporting that 5 methods have 0% coverage, while IntelliJ IDEA is reporting 100% coverage for those same methods. I included both reports in my project.

**Part B (50%)**

Take the code given to you for the employee class and implement the tests created in part A using Junit. Ensure you do the following step.

1. Test the constructor.
2. Run a positive test through each of the public methods.
3. Run your specification-based tests. You can divide these into positive tests and negative tests.
4. Create a test that allows you to invoke the setAge method with an age of 55 (different to the age you give when you create the instance) and verify that the age member has been updated.
5. Make sure that the Constructure test is always run first.
6. Turn on code coverage. Use the JaCoCo tool. Include a copy of the code coverage report in your answer folder.

Are there any parts of the code that require more testing?

If so, add the required tests. Include a comment in your test file to indicate any such tests and **why** you are including them.

1. Error Guessing. Are there any other tests you think should be run?
2. Your solution should include some **parameterised testing**.
3. Did you find any error in the code? If so outline the errors found in your word document.

# **Submission Details**

Create a folder and name it – **your name – Tnumber** eg Claire Horgan T000123

Copy your project into this folder.

Also copy your word document where you completed your specification test techniques.

Include a copy of the code coverage report.

Zip your folder.

Upload your file on Canvas. In the software testing module go to assignments. Select the Software Testing CA 2024, select upload.

If you have any questions please email me at [Claire.horgan@mtu.ie](mailto:Claire.horgan@mtu.ie)