COEN 6761 Software Testing and Validation

White-Box and Black box Testing

QA Report

Student Name	Student ID
Harshilsinh Solanki	40298679
Vraj Shah	40311967
Tanisha Fonseca	40266436
Teja Sayila	40321469

Submitted To:

Dr. Yan Liu

Table of Content:

1. Introduction

This report outlines the regression testing process and code changes implemented by the development team in response to the issues identified by the QA team. The purpose of regression testing is to ensure that newly introduced code changes do not negatively impact the existing functionality and that all previously passing test cases continue to pass after the updates.

The RobotController application was reviewed by the QA team, and their feedback identified two significant bugs. We analyzed these findings, implemented appropriate code modifications, wrote regression test cases, and validated the correctness and stability of the updated application.

1.2. GitHub Source Code Update

The following updates were committed to the respective GitHub repositories:

- **Updated source code** of the RobotController class
- **Expanded test suite** including all regression test cases
- **Updated README** with clear build and run instructions

🔗 **QA Report Repository:**
https://github.com/tanishaf28/Coverage/tree/master

🔗 **Developer Response Repository (Task 4):**
https://github.com/Vraj-2011/COEN-6761/tree/main

1.3. Objective

The objective of this report is to document the regression testing and code modifications made in response to bugs identified by the QA team in the RobotController application. It highlights how the development team resolved critical issues, enhanced test coverage, and ensured the application's stability through validation, improved test cases, and effective collaboration.

2. QA Team Findings

The QA team identified two critical bugs during their review of the RobotController system:

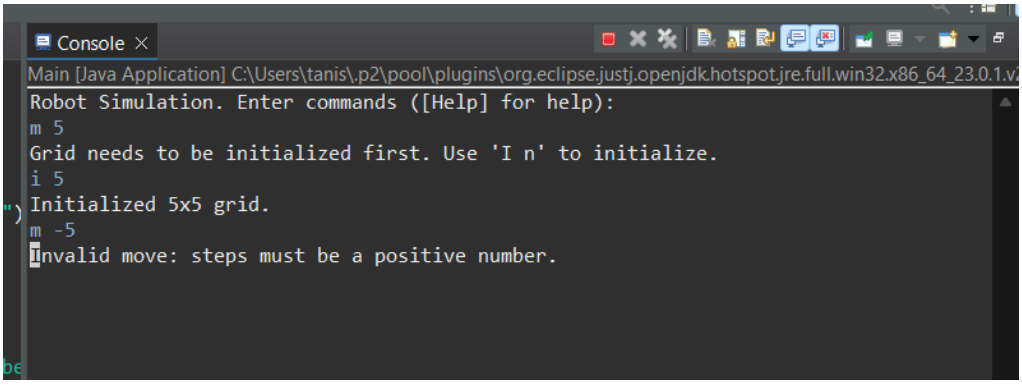
The robot was able to move before the grid was initialized, causing undefined behavior and incorrect history logs. Additionally, the system accepted zero or negative step values, violating logical constraints and risking misuse.

2.1 Move Command Executable Before Initialization

Problem:	The application allowed the robot to execute a move command even before the grid was initialized. This was not anticipated during development, as it was assumed that the user would always initialize the grid first.
Impact:	This led to undefined behavior and inconsistencies in the movement logic, and a misleading entry ("Moved x steps") was being logged to history even without an initialized grid.

2.2 Move Command Accepting Negative Step Values

Problem:	The application accepted zero or negative integers as step values in the move command.
Impact:	The behavior violated expected constraints, as movement should logically be limited to positive integers only. Allowing such input could lead to illogical outcomes or system misuse.



3. Developer Action and Code Changes

The development team reviewed the issues raised and implemented fixes directly within the move method of the RobotController class.

3.1 Prevent Move Before Initialization

A condition was added to the start of the move method to check whether the grid (floor) has been initialized:

```
74      public void move(int steps) {
75          /*Added code based on QA comments*/
76           if (floor == null) {
77                  System.out.println("Grid needs to be initialized first. Use 'I n' to initialize.");
78                  history.add("Attempted to move before initializing the grid.");
79                  return;
80              }
81            if (steps <= 0) {
82                  System.out.println("Invalid move: steps must be a positive number.");
83                  history.add("Attempted to move with invalid steps: " + steps);
84                  return;
85              }
86          for (int i = 0; i < steps; i++) {
87              switch (direction) {
88                  case NORTH -> y = Math.min(y + 1, floor.length - 1);
89                  case SOUTH -> y = Math.max(y - 1, 0);
90                  case EAST -> x = Math.min(x + 1, floor[0].length - 1);
91                  case WEST -> x = Math.max(x - 1, 0);
92              }
93              if (penDown) {
94                  floor[y][x] = 1;
95              }
96          }
97          history.add("Moved " + steps + " steps.");
98          System.out.println("Moved " + steps + " steps.");
99      }
```

3.2 Restrict Non-Positive Step Values

A condition was introduced to restrict movement to only positive step values:

```
if (steps <= 0) {
    System.out.println("Invalid move: steps must be a positive number.").
    history.add("Attempted to move with invalid steps: " + steps).
    return;
}
```

4. Test Case Enhancements

In response to these changes, we added new JUnit test cases to cover the newly handled scenarios and confirm that the system behaves as expected.

4.1 Test: Move Before Initialization

This test checks the behavior of the move method when it's called **before the robot grid has been initialized**.

testMoveBeforeInitialization :

This test checks that the robot **cannot move before the grid is initialized**. It verifies two things:

1. The system displays an appropriate error message:
   "Grid needs to be initialized first. Use 'I n' to initialize."
2. The invalid action is recorded in the robot's history:
   "Attempted to move before initializing the grid."

Why it's important:

This test ensures the application handles incorrect usage gracefully—by showing an appropriate error message and recording the incident—rather than crashing or misbehaving.

Here's a neat and concise explanation of the three tests in words:

4.2 Test: Move with Zero Steps

This test checks how the system handles a move command with zero steps after initializing a 10×10 grid.

1. It verifies that the output shows the correct error message:
   "Invalid move: steps must be a positive number."
2. It also checks that this invalid action is logged in the robot's history as:
   "Attempted to move with invalid steps: 0"

4.3 Test: Move with Negative Steps

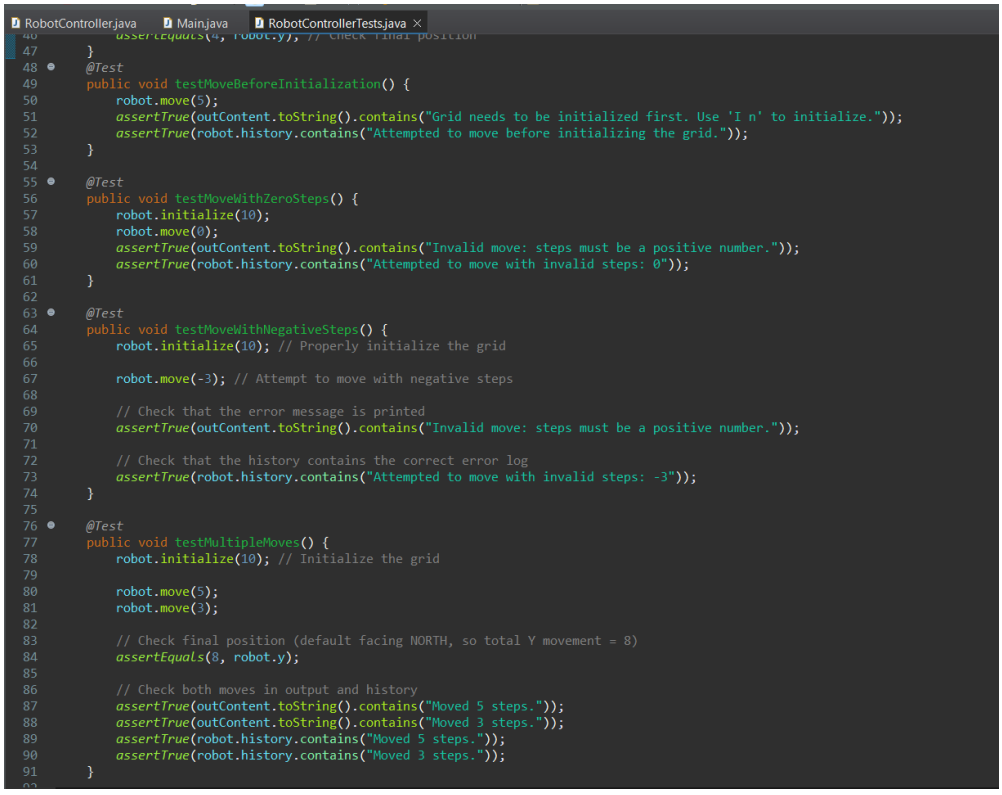
This test ensures the robot does not accept negative steps after initialization.

- It confirms the output shows:
  "Invalid move: steps must be a positive number."
- It also validates the robot's history includes:
  "Attempted to move with invalid steps: -3"

4.4 Test: Multiple Valid Moves

This test checks if the robot correctly performs multiple valid move commands.

- After initializing the grid, the robot moves 5 steps and then 3 more.
- It confirms that the robot's final vertical position (y) is updated to 8.
- The history should record both moves as:
  "Moved 5 steps." and *"Moved 3 steps."*



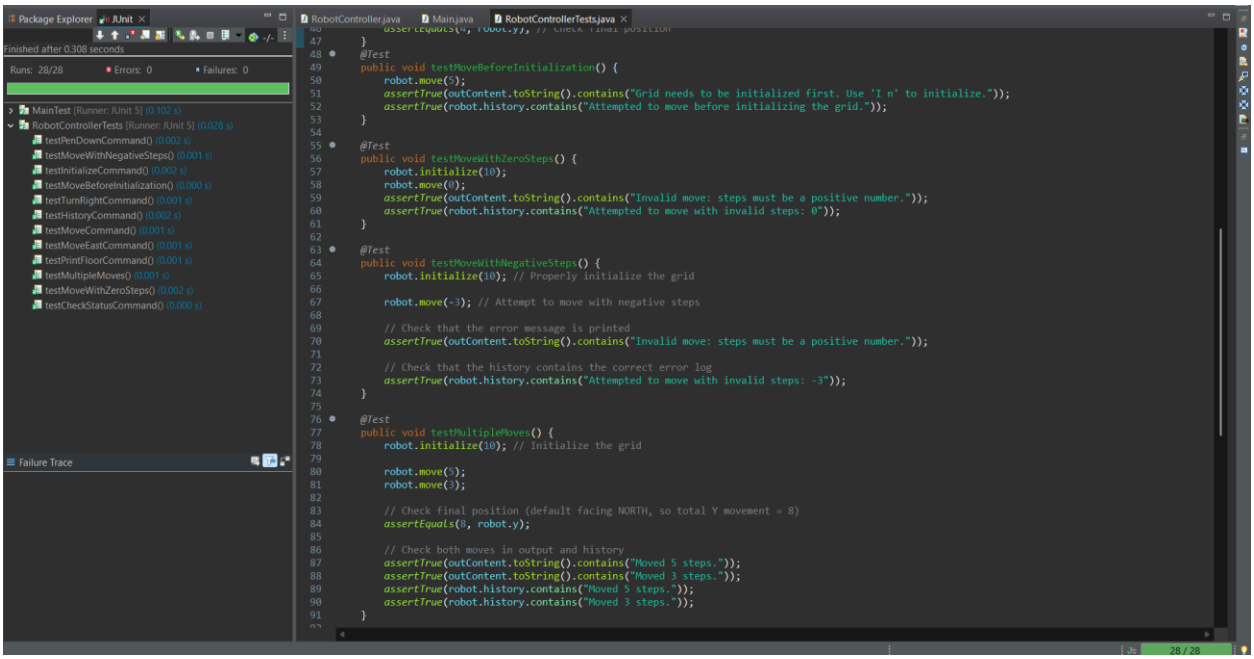
5. Regression Testing and Results

All pre-existing unit tests were re-executed alongside the newly added tests to confirm that the recent changes did not break any part of the system.

Test Coverage and Metrics:

Metric	Value
Total test cases before change	24
New test cases added	4
Total test cases after change	28
Old test cases passed	100%
New test cases passed	100%
Regression bugs introduced	None

The test suite confirmed the correctness of the fixes and verified that the application remains stable.



6. Conclusion

The feedback provided by the QA team helped us uncover critical edge cases that were not initially addressed. By resolving these issues, enhancing our test suite, and performing comprehensive regression testing, we have significantly improved the reliability and robustness of the RobotController application.

These steps have also emphasized the importance of validation, assumptions handling, and cross-functional collaboration in building high-quality software.

7. AI Statement

AI has been minimally utilized in this project to support specific tasks such as automating certain aspects of test case generation and assisting in coverage analysis. While not a central focus, AI has contributed to improving efficiency in testing processes and ensuring more comprehensive coverage. Its role has primarily been in enhancing the accuracy and reliability of the system through targeted applications, with a focus on streamlining operations rather than driving the entire development process.

Report Created by:

Tanisha Fonseca: Test case modifications, changes in dev team code, adding additional test cases wrt code.