TME 3

Test Plan

Odinakachukwu Nzekwe

Athabasca University

**Computer Science 308:**

Java for Programmers

COMP 308

# # Test Plan for Greenhouse Controls Project

# Introduction

The Greenhouse Controls project is designed to simulate a greenhouse control system using Java programming language. This test plan outlines the approach to verify the functionality, reliability, and robustness of the codebase.

# Objective

The primary objective of this test plan is to ensure that the Greenhouse Controls project functions correctly, handles errors gracefully, and provides accurate control over greenhouse operations.

# Testing Approach

The testing approach for the Greenhouse Controls project will involve the following strategies:

Unit Testing: Verify individual components such as classes, methods, and interfaces to ensure they perform as expected.

Integration Testing: Validate the interaction between different components and modules within the system.

Functional Testing: Assess the system's functionality against specified requirements and use cases.

Error Handling Testing: Evaluate the system's response to unexpected inputs, errors, and exceptions.

Performance Testing: Measure the system's performance in terms of event handling, resource utilization, and response time.

Regression Testing: Ensure that modifications or updates to the codebase do not introduce new bugs or regressions.

Test Scenarios

1. Unit Testing

Verify the functionality of individual classes and methods:

Test the addEvent method in the GreenhouseControls class to ensure events are added correctly.

Test event classes such as ThermostatNight, LightOn, WaterOff, etc., to verify their actions.

Test methods in the PowerOn and FixWindow classes to ensure proper fixing and logging.

2. Integration Testing

Verify the interaction between components:

Test the integration between GreenhouseControls and Event classes to ensure events are executed correctly.

Test the integration between Restore and GreenhouseControls classes to verify system state restoration.

3. Functional Testing

Validate the system against functional requirements and use cases:

Test adding different types of events to the greenhouse control system and verify their execution.

Test system shutdown and restoration to ensure proper handling of errors and system state.

4. Error Handling Testing

Assess the system's response to errors and exceptions:

Test scenarios such as power outages, window malfunctions, and unexpected exceptions to verify error handling mechanisms.

5. Performance Testing

Measure the system's performance under various conditions:

Test the system with a large number of events to assess event handling performance.

Measure resource utilization and response time during event execution.

6. Regression Testing

Ensure that recent changes to the codebase do not introduce new defects:

Re-run previous test cases after making modifications to the codebase to identify any regressions.

Test Environment

The Greenhouse Controls project will be tested in the following environment:

Java Development Kit (JDK) 20 or higher

JUnit testing framework for unit testing

Test environment setup with sample greenhouse configurations and events

Conclusion

By following this comprehensive test plan, we aim to ensure that the Greenhouse Controls project meets its functional requirements, performs reliably, and handles errors effectively. Through rigorous testing, we can ensure the stability and correctness of the codebase, thereby delivering a robust greenhouse control system.