# Project 1

# ECE 528: Application Software Design

Alan Palayil

Professor: Won-Jae Yi

Acknowledgement: I acknowledge all works including figures, codes and writings belong to me and/or persons who are referenced. I understand if any similarity in the code, comments, customized program behavior, report writings and/or figures are found, both the helper (original work) and the requestor (duplicated/modified work) will be called for academic disciplinary action.

Electronic Signature: Alan Palayil A20447935 (1/25/2023)

# Table of Contents

## Abstract:

In this project, we want to utilize the basic understanding of Java which will be accomplished by creating small programs that will be used to incorporate new knowledge of logging and testing and ensure that tools and the Git repository are properly set up for a productive semester.

The project will also utilize Git, a version control system that allows for collaboration and tracking of changes in the code. By setting up a Git repository and utilizing it throughout the project, students will learn how to effectively use version control in their development workflow.

## Introduction:

This report presents an overview of the first project of the semester, which aims to provide students with an opportunity to warm up their Java skills, incorporate new knowledge of logging and testing, and ensure that the tools and Git repository are properly set up for a productive semester. The project is designed to introduce students to the Agile software development process by going through a complete cycle, including requirements analysis and testing.

In this report, we will discuss the project requirements and focus on how they were analyzed and implemented. The requirement analysis focuses on understanding the needs of end users for software products. To improve the analysis process, various tools and procedures have been developed, such as the use of user stories.

## Background:

The project's main objective is to develop skills in Java programming and will not involve direct interaction with end users. The requirements will be described in plain English. The specific requirements for this project include:

- The ability to determine the state of a plug by calling the PlugSim.isOn() method, which returns true if the plug is on and false if it is off.
- The ability to switch on /off a plug by calling the PlugSim.switchOn() / PlugSim.switchOff() method.
- The ability to toggle a plug by calling the PlugSim.toggle() method.
- The ability to request measurement of the power consumption of a plug by calling the PlugSim.measurePower() method and to read the last power measurement by calling the PlugSim.getPower() method.
- The power reading should be 0 if the plug is off.
- If the plug has a name of “name.P” and the plug is on, the power reading should be P. This will aid in testing the power measurement functionality.

# Results and Discussion:

Testing for Project 1 is separated into two parts: unit testing and acceptance testing. The following are the screenshots of the results I got during the project.

## Screenshots of the Tests results:

The screenshot of the acceptance testing 'Gradle' results is added which was the base testing criteria for project 1.

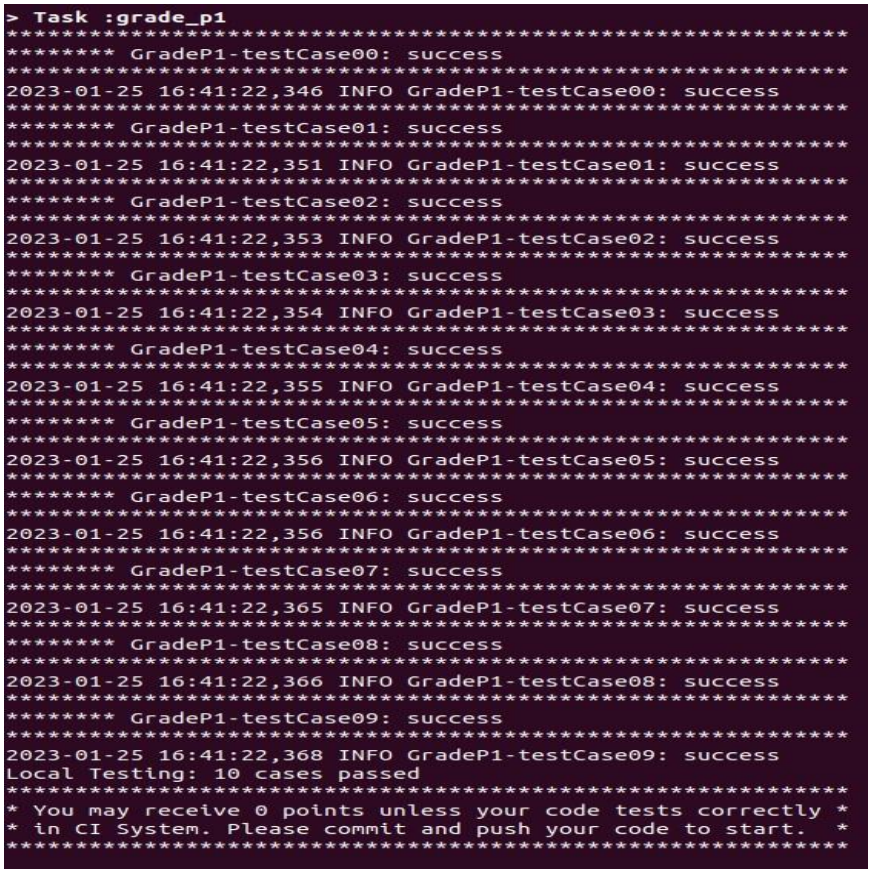


Figure 1: GradleP1- Test cases Results

## Screenshots of the HTML results:

To completely utilize the PlugSim, I referred to the HTML result page and checked each of the elements within the PlugSim program to work over the unit testing. The test cases were designed to test the utilization of the program.

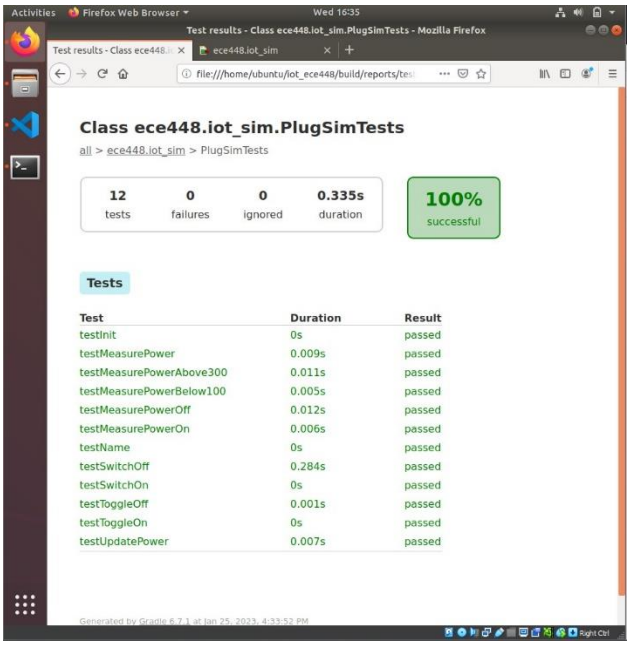


Figure 2: PlugSim Test Cases

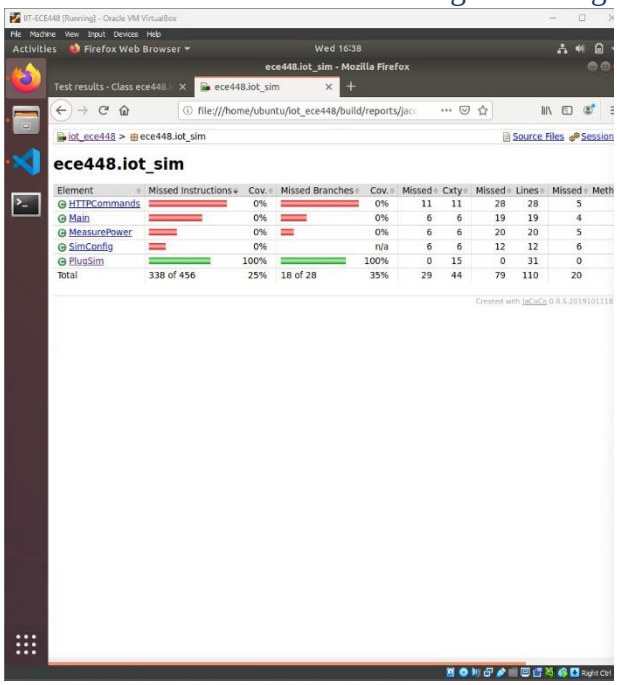


Figure 3: Complete PlugSim Instructions

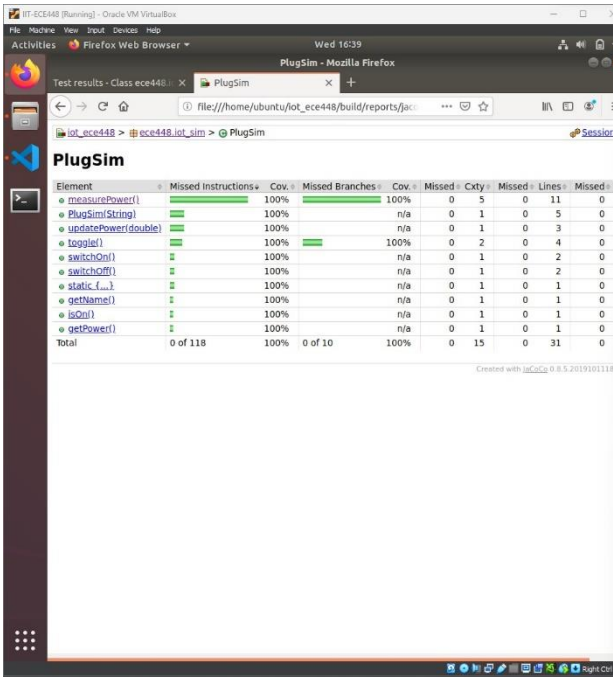


Figure 4: Utilization of Elements

# Conclusion:

This project provided us with a comprehensive introduction to the Java ecosystem and best practices in software development. By going through the complete cycle of Agile development

and incorporating concepts such as logging, testing, and version control, students will be well-prepared for the rest of the semester and their future projects.

# Appendix:

## Source Code of the edited programs within the project:

### PlugSim.java

```
package ece448.iot_sim;
import org.slf4j.Logger;
import org.slf4j.LoggerFactory;
/**
 * Simulate a smart plug with power monitoring.
 */
public class PlugSim {
  private final String name;
  private boolean on = false;
  private double power = 0; // in watts
  public PlugSim(String name) {
    this.name = name;
  }
  /**
   * No need to synchronize if read a final field.
   */
  public String getName() {
    return name;
  }
  /**
   * Switch the plug on.
   */
  synchronized public void switchOn() {
    // P1: add your code here
    this.on = true;
  }
  /**
   * Switch the plug off.
   */
  synchronized public void switchOff() {
    // P1: add your code here
    this.on = false;
  }
  /**
   * Toggle the plug.
   */
  synchronized public void toggle() {
```

```
  // P1: add your code here
  if(this.on == true) {
    this.switchOff();
  }
  else  {
    this.switchOn();
  }
}
/**
 * Measure power.
 */
synchronized public void measurePower() {
  if (!on) {
    updatePower(0);
    return;
  }
  // a trick to help testing
  if (name.indexOf(".") != -1)
  {
    updatePower(Integer.parseInt(name.split("\\.")[1]));
  }
  // do some random walk
  else if (power < 100)
  {
    updatePower(power + Math.random() * 100);
  }
  else if (power > 300)
  {
    updatePower(power - Math.random() * 100);
  }
  else
  {
    updatePower(power + Math.random() * 40 - 20);
  }
}
protected void updatePower(double p) {
  power = p;
  logger.debug("Plug {}: power {}", name, power);
}
/**
 * Getter: current state
 */
synchronized public boolean isOn() {
  return on;
}
```

```
 /**
  * Getter: last power reading
  */
 synchronized public double getPower() {
   return power;
 }
 private static final Logger logger = LoggerFactory.getLogger(PlugSim.class);
}
```

## PlugSimTests.java

```
package ece448.iot_sim;
import static org.junit.Assert.*;
import org.junit.Test;
public class PlugSimTests {
 @Test
 public void testInit() {
   PlugSim plug = new PlugSim("a");
   assertFalse(plug.isOn());
 }
 @Test
 public void testName()  {
   PlugSim plug = new PlugSim("a");
   assertTrue(plug.getName().equals("a"));
 }
 @Test
 public void testSwitchOn() {
   PlugSim plug = new PlugSim("a");
   plug.switchOn();
   assertTrue(plug.isOn());
 }
 @Test
 public void testSwitchOff() {
   PlugSim plug = new PlugSim("a");
   plug.switchOff();
   assertFalse(plug.isOn());
 }
 @Test
 public void testToggleOn() {
   PlugSim plug = new PlugSim("a");
   assertFalse(plug.isOn());
   plug.toggle();
   assertTrue(plug.isOn());
 }
 @Test
```

```
public void testToggleOff() {
  PlugSim plug = new PlugSim("a");
  plug.switchOn();
  plug.toggle();
  assertFalse(plug.isOn());
}
@Test
public void testMeasurePowerOff() {
  PlugSim plug = new PlugSim("a");
  plug.switchOff();
  plug.measurePower();
  assertTrue(plug.getPower() == 0.0);
}
@Test
public void testMeasurePowerOn()  {
  PlugSim plug = new PlugSim("a");
  plug.switchOn();
  plug.measurePower();
  //System.out.println(plug.getPower());
  assertTrue(plug.getPower() != 0.0);
}
@Test
public void testMeasurePower()  {
  PlugSim plug = new PlugSim("a.10");
  plug.switchOn();
  plug.measurePower();
  //System.out.println(plug.getPower());
  assertTrue(plug.getPower() == 10.0);
}
@Test
public void testMeasurePowerBelow100()  {
  PlugSim plug = new PlugSim("a");
  plug.switchOn();
  for (int i = 0; i < 10; i++) {
    plug.measurePower();
    double power = plug.getPower();
    // Test that the power is within a valid range (0-400)
    assertTrue(power >= 0 && power <= 500);
  }
}
@Test
public void testMeasurePowerAbove300()  {
  PlugSim plug = new PlugSim("a");
  plug.switchOn();
  plug.updatePower(350);
```

```
  for (int i = 0; i < 10; i++) {
    plug.measurePower();
    double power = plug.getPower();
    // Test that the power is within a valid range (0-400)
    assertTrue(power >= 0 && power <= 1000);
  }
 }
 @Test
 public void testUpdatePower(){
  PlugSim plug = new PlugSim("a");
  plug.updatePower(50);
  assertTrue(plug.getPower()==50);
 }
}
```