Project 1: Java Warm Up

I Overview

While it is still at the beginning of the semester and we haven't learned much of the Java eco-system, it is a good time for us to utilize Project 1 to warm up your Java skills, incorporate new knowledge of logging and testing, and make sure the tools and the Git repository properly for a productive semester.

You probably don't need to write many lines of code for this project. Nevertheless, we would still like to practice Agile software development by going through a complete cycle. In particular, we'll talk about the requirements analysis and the testing in this document.

I Project Requirements

Requirement analysis addresses the needs of the end-users of your software products. Many tools and procedures are developed to help analyze the requirements better such as by using user stories. Nevertheless, our focus is to warm up your Java skills in this project, we won't face end users and will just describe the requirements in plain English.

State of a Plug

One should be able to get the state of a plug by calling **PlugSim.isOn()**, which either returns true if the plug is on, or returns false if the plug is off.

Switch On a Plug

One should be able to switch on a plug by calling **PlugSim.switchOn().**

Switch Off a Plug

One should be able to switch off a plug by calling **PlugSim.switchOff().**

Toggle a Plug

One should be able to toggle a plug by calling **PlugSim.toggle().**

Measure Power Consumption

One should be able to request to measure the power consumption of a plug by calling **PlugSim.measurePower()** and to read the last power measurement by **PlugSim.getPower()**

Power Reading of Plugs in Off State

The power reading should be 0 if the plug is off.

Power Reading of Specific Plugs

If the plug has a name of **"n_a_m_e_._P_"** _and the plug is on, the power reading should be P. This will help us to test the power measurement functionality.

I Testing Procedures

Testing for Project 1 is separated into two parts: unit testing and acceptance testing.

Unit testing is applied to individual classes to make sure they work as designed. The only class we work with for Project 1 is PlugSim, so you will need to write your own unit tests to make sure that PlugSim works. Two examples of unit tests are provided in **src/test/java/ece448/iot_sim/PlugSimTests.java**.

Acceptance testing is used to make sure all project requirements are met. Our acceptance testing procedures are implemented in **ece448.grading.GradeP1** and you can use **gradle grade_p1** to execute all of them. It should be fairly straightforward to verify all requirements are covered.

Note that in Project 1, we only work with a single class, unit testing, and acceptance testing may look pretty much the same. In upcoming projects, when we need to work with multiple classes, they should be quite different.