[UAB](http://main.uab.edu/) -> [ENG](http://www.eng.uab.edu/) -> [ [ECE](http://www.uab.edu/engineering/departments-research/ece) / [ECE Intranet](http://www-ece.eng.uab.edu/) ] -> [ [DGreen](http://www-ece.eng.uab.edu/DGreen/redirect/dgreen-internet.html) / [DGreen Intranet](http://www-ece.eng.uab.edu/DGreen/) ] -> [EE333](http://www-ece.eng.uab.edu/DGreen/ee333/) -> P6–7

**EE333 Programmette 6–7 - Fall 2011**

**Purpose**

The purpose of this project is to produce a **Object-Oriented Design** and demonstration **Java program** that simulates the behavior of a house on emergency power with smart power distribution system (SPDS).

**System Description**

The goal of this project is to demonstrate your ability to do **object oriented design** and then to implement a demonstration of the quality of **that design**. The specific task is to model a house with a generator with specific characteristics, a smart power distribution system, loads with “smart loads”, and a logging system.

The generator has a run time (on a tank of gas) and a maximum current. The generator is connected to the “source” side of the smart power distribution system.

The smart power distribution system (SPDS) has many outlets on it. The outlets are numbered from one to the number of outlets on the system. When the generator starts, the SPDS slowly enables the power to outlets one at at time (starting at 1) and then continues to bring loads on line one at a time no more often than every second. If the maximum current of the generator is exceeded, the SPDS immediately shuts down loads from the highest outlet down towards one until the total current falls below the maximum of the generator. For simplicity, the distribution system contributes no significant resistance to the equivalent electrical circuits. The generator is connected to the SPDS and turned on before any load is turned on. It should be possible add and remove devices to the SPDS both before and after the generator is started. The generator can also be shut down which will turn off all devices immediately. Once this happens and the generator comes back on the SPDS will perform it’s sequencing behavior again.

The loads are of two types: 1) loads which are a constant current and 2) loads that have a startup (or cold) current and another current (hot) in steady state operation. The loads that change do so over a per device time period in a linear manner. Each load has an RFID device that has its own energy source that can supply it’s name and the current into it to the SPDS. The SPDS can send data to up to four data loggers.

Loads can be attached to any presently unused outlet and some outlets may not be in use at particular times. The home owner plugs in important loads to lower numbered outlets and unimportant loads to higher numbered outlets to convey prioritization to the SPDS.

Multiple models of data loggers exist. Some log to a display, some log data to a small flash drive connected to the SPDS, some send data to a remote device connected thru computer networking, some update a LCD monitoring display with current values, etc. At least two different types of data loggers shall be implemented.

A sample simulation is attached as an [Excel File](http://www-ece.eng.uab.edu/DGreen/ee333/P6-7-Sample-SPDS-Operation.xlsx) and as a [PDF file](http://www-ece.eng.uab.edu/DGreen/ee333/P6-7-Sample-SPDS-Operation.pdf).

\**Note: you are expected to have an object oriented design and a matching implementation with a demonstration of quality. You may have to reduce scope in implementation but do not eliminate the object oriented nature of the design nor the demonstration of quality. \**

**Obey Java Documentation Style**

Continue to use the specified [documentation standard for Java source code][JavaDocStd]. Ensure that the javadoc comments contain

1. class responsibilities
2. class and instance behaviors

**P6 Delivery**

You shall deliver two **paper** copies detailing your designed classes and the draft P7 report. As a minimum, the submission should include:

1. Draft P7 Report including at least
2. Relationships between classes
3. Key interactions
4. UML Diagrams expressing the above
5. Statement of design problem
   1. Goals
   2. Constraints
   3. Standards
6. Design process
7. Design alternatives thus far
8. Data ‘maintained’ by the class (will end up in code documentation)
9. Public methods to implement functionality (will end up as code)
10. One or more lines of text describing the data and methods. (will end up as JavaDoc comments)

P6 is **strictly due at the start of class**. After P6 is discussed in class no further submissions will be accepted.

A [template](http://www-ece.eng.uab.edu/DGreen/ee333/p6-7report.doc) for the report is available.

**P7 Report**

Accompanying your P7 submission, you shall produce a report about your **object-oriented design**. This report should include

1. Project definition based on specifications, constraints, goals, and applicable standards
2. Is this a good problem for object oriented solution? Why or why not?
3. Alternatives considered in designing the project
4. Alternatives selected and why
5. Final object approach
6. Relationships between classes
7. Key interactions
8. UML Diagrams expressing the above (at least one class diagram and one interaction diagram are minimum requirement)
9. Process - the approach you chose to follow in developing your solution (beginning with receipt of assignment). You are trying to show you know how to do software design and development.
10. Debug - discuss one of the problems you had in implementation and the steps you took to debug the program
11. Results - how you tested your model and how well it worked. Relate to item 1.
12. Include actual data (listings of runs) to show your project’s execution.
13. How would you do things different next design project?

**P7 is expected to be of the quality that you would use the document as an example to a potential employer of the type of work that you can do.**

A [template](http://www-ece.eng.uab.edu/DGreen/ee333/p6-7report.doc) for the report is available.

**Bonus (for P7, 20 points)**

Deliver JUnit tests for all modeling classes in addition to fulfilling other requirements.

**P7 Delivery**

You shall produce source code that complies with the documentation standards. Your program MUST show your name and BlazerID near the top of the listing and display out your name and BlazerID at the start of the test run. Produce a blazerid-p7.zip file containing a directory blazerid-p7 which contains all files (at least the report file, javadoc files, .java and .class files) and submit it using the assignment tool of *BBLearn* by class time of the due date (or if you attend class and sign the class roll for the day, you can automatically have an extension to 11:59pm of the same day.)

Last modified: 29 September 2011  
David Green [.](http://validator.w3.org/check?uri=referer)