# COE318 – Lab 1: Introduction to NetBeans and Java

## Objectives

- Learn how to use the Netbeans Integrated Development Environment (IDE).
- Learn how to generate and write formatted API documentation.
- Add a constructor, some getter and setter methods, conditional statements, and some instance variables to a template for a class.

**Duration:** one week.

**Grading Scheme:**

40% submitted source code

30% in-class demonstration and questions (during week 3 lab hours)

20% in-class quiz – Held during the first 5 mins of the lab class (during week 3 lab hours)

10% lab attendance

## Overview

Object-oriented programming creates models of "things" and allows them to be manipulated through "methods".

Suppose the "thing" we want to model is a Resistor. Imagine you are given a resistor (say a 50 Ω one). You measure its resistance and, sure enough, its value is 50 Ω. Then you measure its voltage and current and get zero. Good!

Next you connect it to a 2 Amp current source. The voltage should now measure 100 Volts and the current (of course) should measure 2 Amps.

In the object-oriented world, obtaining something like a new resistor is done with a constructor. For example, *Resistor r = new Resistor(50)* would create a new Resistor object with a value of 50 Ω by invoking the code for the Resistor's constructor.

"Measuring something" in the object world is accomplished by invoking a "getter" method; for example, *double v = r.getVoltage()* would give the resistor's voltage.

Connecting a resistor to a voltage or current source is done with a "setter" method; for

example, *r.setVoltage(100)* would effectively connect the resistor to a 100 Volt source. The "setter" method should check whether the the voltage value is within the limits of the voltage source (see skeleton code for limits).

In this lab, you are given the skeleton java code for a Resistor. You will have to complete it by fixing the constructor and getter/setter methods (but this requires very little new code.) The lab exercise is mainly meant to introduce you to using the Netbeans IDE with a simple example of object-oriented programming in Java.

## Step 1: Create a Netbeans Project

1. Start Netbeans.

2. Choose *File* on the menu bar, then *New Project*.

3. A New Project window will appear. Click on *Java* in the Categories column and

*Java Class Library* in the Projects column. Then click *Next*.

4. A New Class Library window will appear. Name the project *Lab1*. For project location, click *Browse*. Create a new folder called *coe318* and then a subfolder called *lab1*.

5. Click *Finish*.

## Step 2: Create a Java class source code file

1. Choose *File* on the menu bar, then *New File*.

2. In the New File window, click on *Java* as the category and *Java Class* as the file type. Click *Next*.

3. Name the class Resistor.

4. Name the package coe318.lab1

5. Click *Finish*.

6. An editor window tab named Resistor will appear.

7. Delete everything in it; then copy and paste the source code into the editor.

**Source code**

The skeleton code is provided with the lab handout in the file named “Resistor.java”. Copy the code and paste it into your own Resistor class.

## Step 3: Generate javadocs, compile and run

1. Select *Run* in the menu bar and then click on *Generate javadoc for Lab1*.

2. Your web browser will start up (or a tab will open) and display the API (Application Programming Interface) for the Lab1 project.

3. Look at the web page and click Resistor, then click on main. The documentation for "main" tells you what the output of running the program should be.

4. You can also compile and run the class by selecting Run in the menu bar and clicking on *Run Project*.

5. Note that although it will run, the program's output is incorrect. For example, it says that the resistance of a 50 Ohm resistor is 0!

## Step 4: Add an instance variable, fix constructor, and getResistance()

1. The getResistance() method you are given is:

*public double getResistance() { return 0.0; }*

2. You need to have it return the actual value of the Resistor's resistance. This should be an instance (or state) variable of a Resistor object.

3. You have to declare this instance variable. It should have private visibility (no need to understand what "visibility" means for this lab) and be of type double.

4. Of course you also have to choose a name for it. The name should be descriptive of its meaning. For example, this is a very poor choice (although it would be legal):

```
private double guessWhatIam;
```

5. However, **do not** use the name resistance. (You will soon learn why; you will also learn how you can use this name but it requires a little bit more Java knowledge.)

6. You have to initialize its value in the constructor and use its value in the

getResistance() method. Note that the constructor is declared:

```
public Resistor(double resistance) {
```

7. The value of the Resistor object to be created is given by the resistance parameter.

8. You simply have to assign this parameter to whatever name you have chosen as the object's instance variable.

## Step 5: Fix remaining methods

To finish the lab, you need to fix the remaining getter and setter methods (`getVoltage()`, `setCurrent(double current)`, etc.) Once this is done, the output from the program should be correct. Don't forget voltage and current limits, use conditional statements!

Hint: you will have to add at least one instance variable; however, you do not need to add an instance variable for each value that has a "getter" method. In particular, if the resistance is known along with any one of the resistor's voltage, current or power, the other two can be calculated using Ohm's Law. The calculation would be performed in a "getter" method.

## Step 6: Submit your lab

You must submit your lab electronically on D2L. Please make sure you hand over the quiz answer sheet to the TA at the end of the in-class quiz.

Please zip up your NetBeans project containing all source files and submit to the respective assignment folder on D2L.