coe318 Lab 1
Introduction to Netbeans and Java

Week of September 12, 2016

Objectives

- Lean 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 and some instance variables to a template for a class.
- **Duration: one week.**

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.

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` `coe318` and then a subfolder called lab1. 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 following source code into the editor.

Source code

You can copy and paste the following source code into your Netbeans editor. To open a copy in your browser for easier copying/pasting, click here.

```
/**
 * A Resistor models an ideal resistor that obeys Ohm's Law.
 *
 * @author YourName
 */
package coe318.lab1;

public class Resistor {
    //Instance (state) variables
    //TODO Add instance variables (Hint: you only need 2!)

    /**
     * Create an ideal Resistor. The initial current through and voltage
across
     * the Resistor are zero.
     *
     * @param resistance resistance in Ohms
     */
    public Resistor(double resistance) {
        //Set values of state variables
    }

    /**
     * Returns the value of the resistor in Ohms.
     *
     * @return the resistance
     */
    public double getResistance() {
        //FIX THIS so that it returns the actual resistance
        return 0.0;
    }
```

```
/**
  * Returns the voltage across the resistor.
  *
  * @return the voltage
  */
 public double getVoltage() {
     //FIX THIS so that it returns the actual voltage
     return 0.0;
 }

 /**
  * Sets the value of the voltage across the resistor.
  *
  * @param voltage the voltage to set
  */
 public void setVoltage(double voltage) {
     //FIX THIS
 }

 /**
  * Returns the current through the Resistor.
  *
  * @return the current
  */
 public double getCurrent() {
     //FIX THIS
     return 0.0;
 }

 /**
  * Sets the value of the current through the resistor.
  *
  * @param current the current to set
  */
 public void setCurrent(double current) {
     //FIX THIS
 }

 /**
  * Returns the power (in Watts) dissipated by the Resistor.
  *
  * @return the power
  */
 public double getPower() {
     //FIX THIS
     return 0.0;
 }


/**
  * A simple example of using a Resistor. <p> The output should be:
  * <pre>
  * Creating a 50 Ohm resistor (r1)
  * Its resistance is 50.0 Ohms
  * Its current is 0.0 Amps
  * Its voltage is 0.0 Volts
```

```
     * Its power is 0.0 Watts
     * Creating a 100 Ohm resistor (r2)
     * Its resistance is 100.0 Ohms
     * Setting r1's current to 2 Amps
     * Its current is 2.0 Amps
     * Its voltage is 100.0 Volts
     * Its power is 200.0 Watts
     * Setting r1's voltage to 50 Volts
     * Its current is 1.0 Amps
     * Setting r2's current to 3 Amps
     * Its voltage is 300.0 Volts
     * </pre>
     *
     * @param args (Command line arguments not used.)
     */
    public static void main(String[] args) {
        Resistor r1, r2;
        System.out.println("Creating a 50 Ohm resistor (r1)");
        r1 = new Resistor(50.0);
        System.out.println("Its resistance is "
                + r1.getResistance() + " Ohms");
        System.out.println("Its current is "
                + r1.getCurrent() + " Amps");
        System.out.println("Its voltage is "
                + r1.getVoltage() + " Volts");
        System.out.println("Its power is "
                + r1.getPower() + " Watts");
        System.out.println("Creating a 100 Ohm resistor (r2)");
        r2 = new Resistor(100.0);
        System.out.println("Its resistance is "
                + r2.getResistance() + " Ohms");
        System.out.println("Setting r1's current to 2 Amps");
        r1.setCurrent(2.0);
        System.out.println("Its current is "
                + r1.getCurrent() + " Amps");
        System.out.println("Its voltage is "
                + r1.getVoltage() + " Volts");
        System.out.println("Its power is "
                + r1.getPower() + " Watts");
        System.out.println("Setting r1's voltage to 50 Volts");
        r1.setVoltage(50.0);
        System.out.println("Its current is "
                + r1.getCurrent() + " Amps");
        System.out.println("Setting r2's current to 3 Amps");
        r2.setCurrent(3.0);
        System.out.println("Its voltage is "
                + r2.getVoltage() + " Volts");

    }
}
```

Step 3: Generate javadocs, compile and run

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

2. Your web browser will start up (or a tab will open) and display the API (Application Programming Interface) for the Resistor class.
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 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;
```

1. 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.)
5. 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) {
```
6. The value of the Resistor object to be created is given by the `resistance` parameter.
7. 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.

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 at least 24 hours prior to the start of your scheduled lab period for Lab 2.

If you did the lab on a Departmental computer, you can do the following:

```
cd coe318
zip -r lab1.zip lab1
submit coe318 lab1 lab1.zip
```

cd coe318
zip -r lab1.zip lab1
submit coe318 lab1 lab1.zip

If you did the lab on your own computer, zip the lab1 folder (remember to do this recursively so that all sub-folders are included), then transfer the zip file to a Departmental machine, logon to a Departmental machine which can be done remotely) and type in the submit command:

```
submit coe318 lab1 lab1.zip
```

submit coe318 lab1 lab1.zip