COE318 – Simple Resistive Circuits Solver

Objectives

- Develop an application based on requirements.
- Use JUnit for testing.

In this lab, you will model and solve simple DC circuits composed of any number of resistors and voltage sources.

Unlike previous labs, you are not given the design for this application. It is up to you to decide what classes, interfaces, methods, etc. that you will need. The one exception is that you must have a class called `UserMain` that has a `main()` method that reads and interprets input from *stdin* (by default, the keyboard).

Duration: two weeks.

Grading Scheme:

50% submitted source code

20% in-class demonstration and questions (during week 7 lab hours)

25% in-class quiz – Held during the first 5 mins of the lab class (during week 7 lab hours)

5% attendance

Overview

An electric circuit will be described by the user. Each line will describe either a Resistor or a DC Voltage source or be a single word command.

The format for describing (for example) a 5.2 Ohm resistor connected between nodes 2 and 3 is:

```
r 2 3 5.2
```

The format for describing a 6.5 Volt source connected between nodes 1 and 2 (where the positive side of the source is connected to node 1) is:

```
v 1 2 6.5
```

A complete circuit could be described as follows:

```
 v 1 0 2.0
 r 1 2 0.25
 v 2 0 3
 r 2 3 0.5
 r 3 0 1.0
end
```

To be correct, a circuit with *n* nodes must name the nodes 0, 1,...*n-1*. The order in which the components are described does not matter. For non-polarized components (such as a resistor), the order in which the nodes are named does not matter. For example,

```
r 1 2 0.25
```

is equivalent to:

```
r 2 1 0.2
```

For polarized components (such as a voltage source), the order does matter. Thus:

```
v 1 0 2.0
```

is equivalent to:

```
v 0 1 -2.0
```

In addition to lines describing the components of a circuit, there are 2 other single word commands that can be entered: `spice` and `end`.

The `end` is the simplest to understand and implement. When the `end` command is entered, the program should print `All Done` and terminate.

The `spice` command should print the spice description of the circuit entered so far. In the spice description, uppercase letters are used, components are numbered sequentially and `DC` is used in the description of voltage sources. An example session follows (the lines in bold denote output from the program; the non-bold lines are input):

```
v 1 0 2.0
r 1 2 0.25
v 2 0 3
r 2 3 0.5
r 3 0 1.0
 spice

V1 1 0 DC 2.0
R1 1 2 0.25

V2 2 0 DC 3.0

R2 2 3 0.5

R3 3 0 1.0

end

All Done
```

Source Code

No source code template is given for this lab. You will have to write the code from scratch.

Step 1: Create a Netbeans Project and implement the *end* command

1. Create a Netbeans project called `AnalogCircuit` which should be placed in a folder called `lab6` (all lowercase and no spaces). The `lab6` folder should itself be in your `coe318` folder.

2. Create a class `UserMain` with a `main` method that reads *stdin* and interprets the `end` command. This and all classes should be in a package called `coe318.lab6.`

Step 2: Interpret the circuit and Implement the *spice* command

You need to define classes that will allow you to model the circuit. Include javadoc comments for all public methods, classes, interfaces and constructors. Once you can interpret circuit components, you should be able to implement the `spice` command.

Step 3: Write Unit Tests for one of the classes

Write JUnit tests for testing at least two methods of one of the classes that you write.

Step 4: 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.