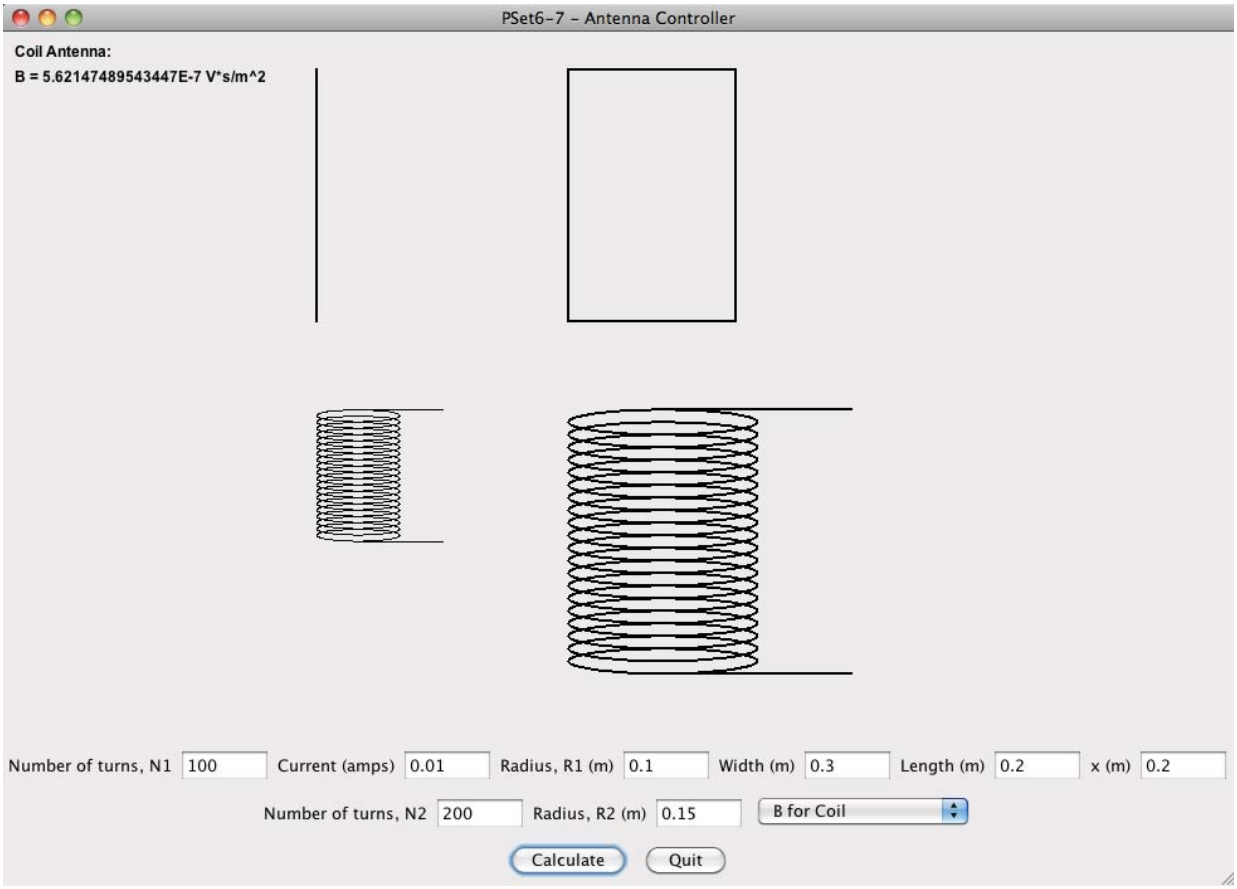
Introduction to Computers and Engineering Problem Solving
Spring 2012
Problem Sets 6 and 7: Antenna GUI
Due: 12 noon, Friday, April 6, 2012 (Problem Set 6)
12 noon, Friday, April 20, 2012 (Problem Set 7)

Problem statement

You are to write a graphical user interface (GUI) to compute and display the magnetic inductance of line, coil and rectangular antennas. You'll do it in two stages, starting in homework 6 and finishing in homework 7. We use exactly the same equations as in homework 1.

Your GUI should resemble the picture below.



Refer to homework 1 for the five equations for inductance of line, coil and rectangle antennae. You must use the model-view-controller paradigm to write the GUI.

You must design and write appropriate model classes, using inheritance to model the three types of antenna (line, coil and rectangle). The `Antenna` classes will contain the appropriate data members,set() ,and get() methods; they must not have extraneous data or methods that do not apply to them. This is a generalization of the model-view-controller paradigm; in this homework you will have multiple model classes.

The computational methods that you must implement are:

- Line antenna:
  - Compute magnetic inductance (equation 1)
- Coil antenna:
  - Compute magnetic inductance (equation 2)
  - Compute mutual inductance with another coil antenna (equation 4)
- Rectangle antenna:
  - Compute magnetic inductance (equation 3)
  - Compute mutual inductance with a line antenna (equation 5)

These methods take one argument, distance x, except for equation 4, which must also accept a reference to a second coil antenna. (In equation 1, the distance is r.)

You must also write a `AntennaView` class, which obtains all data and results from the model classes needed to draw the antennas being analyzed, and to print the inductance computed from the model. This class has a `paintComponent()` method as its key element. Its display should generally look like the figure above: it must show the relevant antennas and their characteristics.

Third, you must write a `AntennaController` class. It must meet the following requirements:

- It must have labels and text fields to allow the six inputs to be entered (current, number of windings, coil radius, rectangle length and width, distance x), as appropriate
- It must allow the user to select the analysis to be done; the 5 choices are the 5 equations. You may use a combo box.
- It must have "Quit" and "Calculate" buttons as shown in the figure above
- It must use appropriate panels and layout managers to place the input components. Your GUI must resemble the example given above.
- It must have appropriate anonymous inner classes as listeners for the buttons
  - The "Calculate" button listener should parse all the inputs, checking that all are greater than zero, and set the model parameters to them. You may require all inputs to be present, even if the requested analysis does not use all of them, for simplicity.
  - The "Quit" button should end the application.

Your program must be a model-view-controller implementation. The model classes (`Antenna` and others) will perform the calculations. A view class (`AntennaView`) will draw the antennas and display the results of the calculations. A controller class (`AntennaController`) will allow user input and control program execution. As you develop your classes, keep in mind how the model(s), the view and the controller need to interact.

Use the sample output and solution from homework 1 to check your model results.

Problem Set 6

1. Models
   Write the `Antenna` classes to implement the equations. The classes must have appropriate data members and methods to be used as the models in your final program. The controller (problem set 7) will need to transfer user input to the models. The view (problem set 7) will have to get all the necessary data from the models to draw the antennas and display the analysis results.

2. Controller
   Create a class `AntennaController` that extends `JFrame`.
   a. The class should have the appropriate textboxes, combo box, and buttons displayed and functional. Use anonymous inner classes to listen for events.
   b. No antennas need to be drawn yet, and no analysis outputs need to be displayed. For now, you may insert an empty `JPanel` in that area.
   c. The class should have a `main(String[] args)` method that creates and displays an instance of the class.

Hand in the `Antenna` classes and the `AntennaController` class to complete problem set 6. Your program should work except that nothing is drawn or output when it is run.

Problem Set 7

3. View

Write a class `AntennaView` that extends `JPanel`.

   a. You may set a fixed size for the panel.
   b. Apart from the constructor and perhaps a set() method, the only public method in this class should be `paintComponent(Graphics g)`, which draws the antenna(s) and displays the analysis results. It obtains the data values it needs by calling methods on the model classes. You are free to make changes to your model and controller classes from those that you handed in for problem set 6. The antennas must be drawn approximately to scale.

To complete problem set 7, hand in the `Antenna` model classes, `AntennaView`, and `AntennaController` classes that make up the complete application.

Turn In

For each problem set:

a. Place a comment with your full name, section, TA name and assignment number at the beginning of all .java files in your solution.
b. Place all of the files in your solution in a single zip file.
c. Do not turn in electronic or printed copies of compiled byte code (.class files) or backup source code (.java~ files)
d. Do not turn in printed copies of your solution.

2. Submit this single zip file on the 1.00 Web site under the appropriate section and problem set number. For directions see **How To: Submit Homework** on the 1.00 Web site.
3. Your solution is due at noon. Your uploaded files should have a timestamp of no later than noon on the due date.
4. After you submit your solution, please recheck that you submitted your .java file. If you submitted your .class file, you will receive **zero credit.**

Penalties

For each problem set:

- 30 points off if you turn in your problem set after Friday noon but before noon on the following Monday. You have one no-penalty late submission per term for a turn-in after Friday noon and before Monday noon.
  - Problem set 6 is due Wednesday noon because of a Friday holiday. You may turn it in for the late penalty by the following Monday at noon.

- No credit if you turn in your problem set after noon on the following Monday.

1.00 / 1.001 / 1.002 Introduction to Computers and Engineering Problem Solving
Spring 2012