Introduction to Computers and Engineering Problem Solving

Spring 2005

Problem Sets 6 and 7: User interface for catenary height

Problem Set 6 Due: 12 Noon, Session 22
Problem Set 7 Due: 12 Noon, Session 25

Swing application

You are to write a graphical user interface for the catenary calculations that you coded in problem set 2. You'll do it in two stages, starting in homework 6 and finishing in 7.

Overall Assignment

Create a graphical user interface using Swing that resembles, but looks better than, the screen shot in Figure 1.

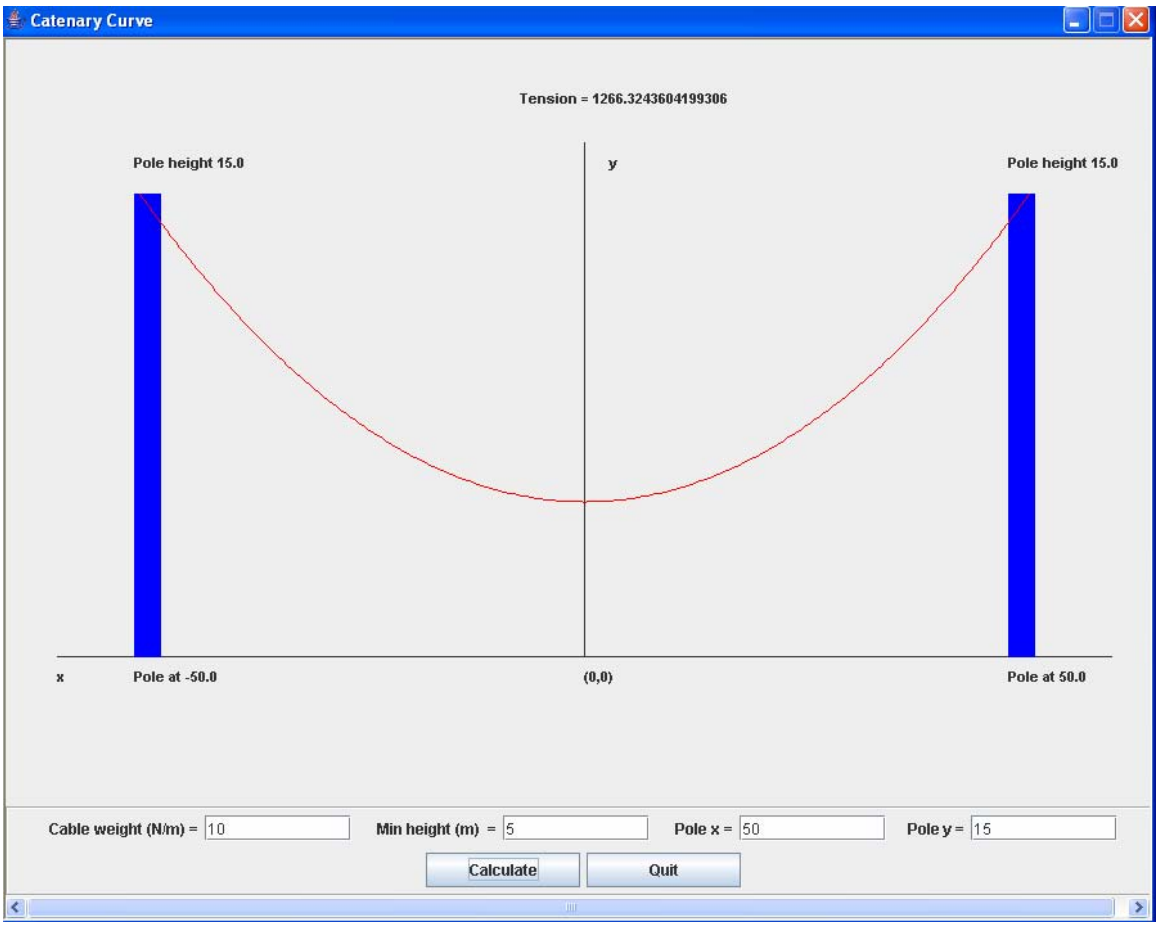

Figure 1. User interface example

The user interface should prompt the user to enter the following data via text fields:

- Weight w
- Minimum y (yMin)
- Pole location xPole
- Pole height yPole

The interface must also have a 'calculate' button to do the calculation of catenary height between the two poles and display the result, and a 'quit' button to exit the program. When the 'calculate' button is clicked, the interface must draw a picture showing:

- The x and y axes
- Two poles, labeled as poles with their locations +/-xPole and height yPole
- Cable tension
- The catenary between the two poles, drawn with enough points to look like a smooth curve to the user. You must determine the number of points.

You are free to choose the colors, fonts, size, etc. Make them look good.

Suggested Solution Method

1. Write a CatenaryModel class to implement the calculations described in problem set 2. This class should have appropriate data members and methods to be used as the 'model' by the GUI code.
2. Write a CatenaryView class that extends JPanel:
    a. The CatenaryModel object is the only data member of CatenaryView.
    b. It has a one argument constructor, which takes a CatenaryModel object as its parameter.
    c. The only method this class has is paintComponent(Graphics g). This method:
        i. Invokes `super.paintComponent(g);`
        ii. Casts g to a Graphics2D object.
        iii. Gets needed results from CatenaryModel, using its get() methods.
        iv. Sets the font for the text strings to be output: `g2.setFont(…);`
        v. Draws the strings on the figure (see figure 1): label axes, poles, tension using `g2.drawString(…)`
        vi. Draws the figure at the top of the canvas:
            1. Draw horizontal lines for the x and y axes.
            2. Draw the two poles
            3. Draw the catenary.

        Use lines, rectangles, polylines or other shapes as needed.

3. Create a class CatenaryController that extends JFrame; this will be your user interface window.

a. Define the data members that you'll need:

a. Components like buttons, text fields and labels
b. JPanel(s) to hold all of the components
c. Size of the frame
d. A reference to a CatenaryModel object
e. A reference to a CatenaryView object

b. In the CatenaryController class, write a main method that creates a new instance of your CatenaryController class and displays it on the screen.

c. Constructor, which:

- Gets the contentPane
- Creates the panel holding buttons, etc. and adds it to the pane
- Creates the text fields, labels and buttons and adds them to their panel

d. Within the constructor, you should write two anonymous inner classes:

- ActionListener, with an actionPerformed() method for the calculation button. The method should:

  - ✓ Use the getText() method to obtain the values of all the inputs in text boxes
  - ✓ Parse the inputs to doubles
  - ✓ Create a CatenaryModel object with the input parameters
  - ✓ Create a CatenaryView object with the CatenaryModel object as its parameter
  - ✓ Add the CatenaryView object to the contentPane
  - ✓ Call repaint() on the CatenaryView object to invoke its paintComponent() method, which does all the calculations and draws the output.

  (Alternatively, you may create one instance of the CatenaryModel and CatenaryView objects in the constructor, and use their set() methods in each run instead of creating new objects each time.)

- ActionListener, with an actionPerformed() method for the quit button. The method should call System.exit(0).

e. Make sure exceptions and invalid inputs are handled properly. For example, display error messages if

- ✓ There are empty input text fields
- ✓ The height of pole Y is not greater than the minimum height
- ✓ Other exceptions that might occur...

Important: Use message dialog window to display messages. No message is to be displayed on console!

Problem Set 6

For problem set 6:

1. Complete the CatenaryModel class (step 1 in the Suggested Solution Method above)
2. Complete the Swing visual interface (steps 3a, 3b and 3c in the Suggested Solution Method above). Your visual interface doesn't have to, and won't, do anything. You must just display it, including all the text boxes and buttons, as shown in Figure 1. This will be a partial implementation of the CatenaryController class. It must compile and run (that is, it must display on the screen but not do anything).

Hand in the CatenaryModel and CatenaryController classes.

Problem Set 7

For problem set 7, complete the remainder of the application: write the CatenaryView class and complete CatenaryController. Hand in all three classes that make up your fully working application.

Turn In

1. Place a comment with your full name, MIT server username, section, TA name and assignment number at the beginning of all `.java` files in your solution.
2. Place all of the files in your solution in a single zip file.
    a. Do not turn in electronic or printed copies of compiled byte code (`.class` files) or backup source code (`.java~` files)
    b. Do not turn in printed copies of your solution.
3. Submit this single zip file.
4. Your solution is due at noon. Your uploaded files should have a timestamp of no later than noon on the due date.

Penalties

- 30% 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.
- No credit if you turn in your problem set after noon on the following Monday.