




Assignment # 5

PRESENTED BY: ERIC DOCKERY

NOVEMBER 11, 2013



Programming Project 9.8

Design and implement an application that displays an animation of a horizontal line segment moving across the screen, eventually passing across a vertical line. As the vertical line is passed, the horizontal line should change color. The change of color should occur while the horizontal line crosses the vertical one; therefore, while crossing, the horizontal line will be two different colors.

How to Solve:

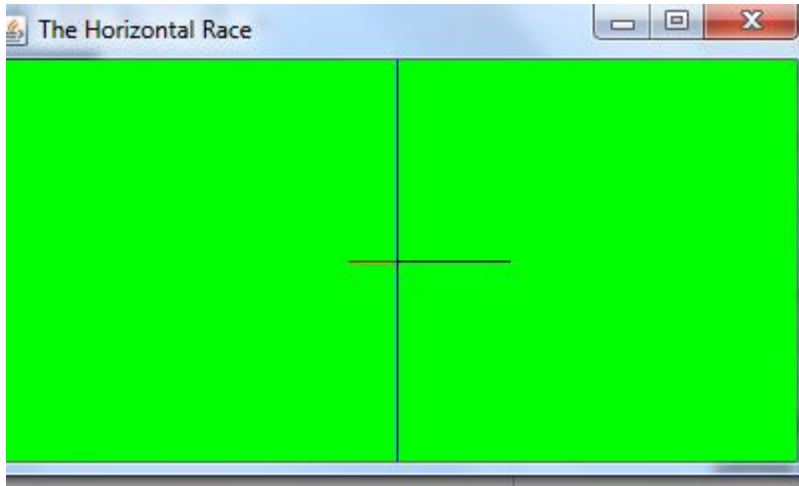
This problem involves some graphics interface with a drawing function that will change with a time delay. The best way to develop this was to build a horizontal line that will move with time across the GUI. This line when hitting the vertical line it crosses will change color. This was implemented with a set of if, else if and else statements that broke down if the horizontal line was on the right of the line one color else if it was on the other side of the line the other color and lastly if it had part on one side of the line one color while the other part a different color. The tricky part is when the horizontal line reaches the end of the GUI. For my project I would have liked to have it travel backwards from the left over the cross line. Unfortunately, at this time I don't seem to be able to run that so I have made the program restart the horizontal line back at the start. (if I have time to tweek the program before it is due then I plan on trying to fix this.

Screenshots:

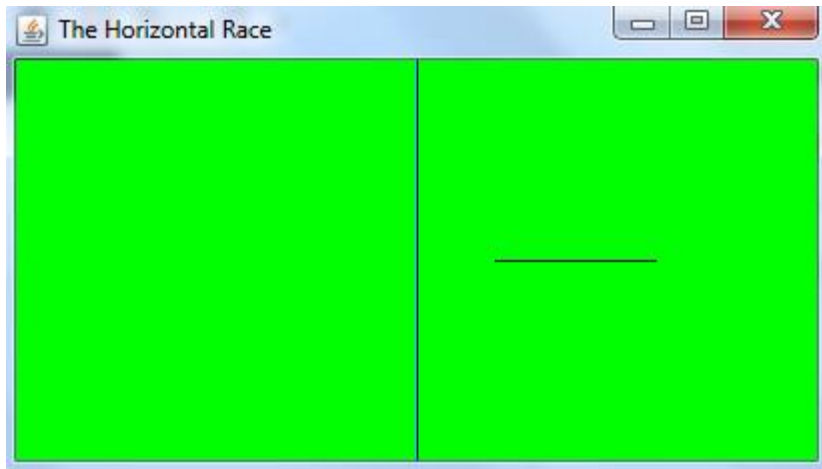
Before crossing the line:



In between the line:



After Crossing the line:



Programming Project 9.10

Design and implement an application that works as a stopwatch. Include a display that shows the time (in seconds) as it increments. Include buttons that allow the user to start and stop the time and reset the display to zero. Arrange the components to present a nice interface. Hint: use the Timer class to control the timing of the stopwatch.

How to Solve:

To solve this problem you must build first the Watch Frame work that calls a WatchPanel to build the interface. The WatchPanel will set the dimensions of the stop watch as well as add the buttons and action listeners that will start stop

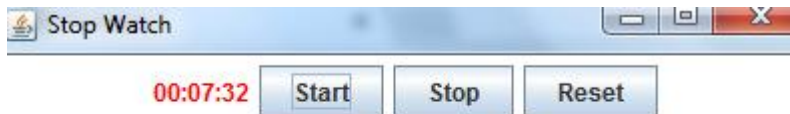
and reset the stop watch. This program is an application of the `timer.start()` and `timer.stop()` functions along with the start and stop buttons.

Screenshots:

Run(no buttons pressed):



Start:



Stop:



Reset



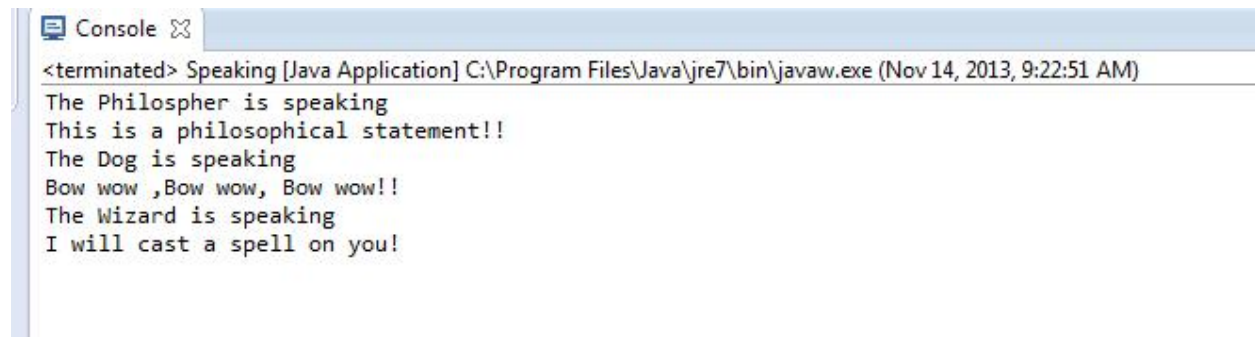
Programming Project 10.3

Implement the Speaker interface described in section 10.3, and create three classes that implement Speaker in various ways. Create a driver class whose main method instantiates some of these objects and tests their abilities.

How to Solve:

To solve this problem you simply create an interface speaker with three other classes that implements them. Then create a driver class to call speaker using a variable then calling that variable to the other three classes. In my program I used wizard, philosopher, and dog. These classes are called using `TheSpeaker.current = new (class)`. Then I set `TheSpeaker.current.announce("Name of method")` and `TheSpeaker.current.speak()`. The result is in the screenshot below.

Screenshots:



```

Console
<terminated> Speaking [Java Application] C:\Program Files\Java\jre7\bin\javaw.exe (Nov 14, 2013, 9:22:51 AM)
The Philosopher is speaking
This is a philosophical statement!!
The Dog is speaking
Bow wow ,Bow wow, Bow wow!!
The Wizard is speaking
I will cast a spell on you!

```

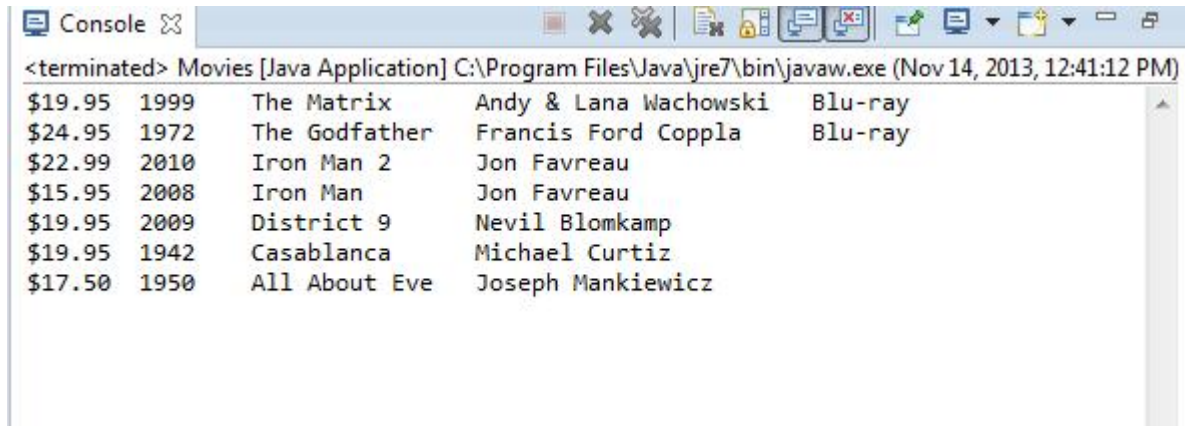
Programming Project 10.5

Modify the Movies program from Chapter 8 so that it keeps the DVDs sorted by title.

How to Solve:

To solve this problem you must compare the titles in each array using a nested loop I chose a selection sort in descending order as a method called from movies. This program was tricky to get my sort working correctly. At first I tried running a bubble sort but I couldn't get the flag to turn false. I am not sure what was wrong with the logic on it but selection sort worked fine.

Screenshots:



Programming Project 8.19

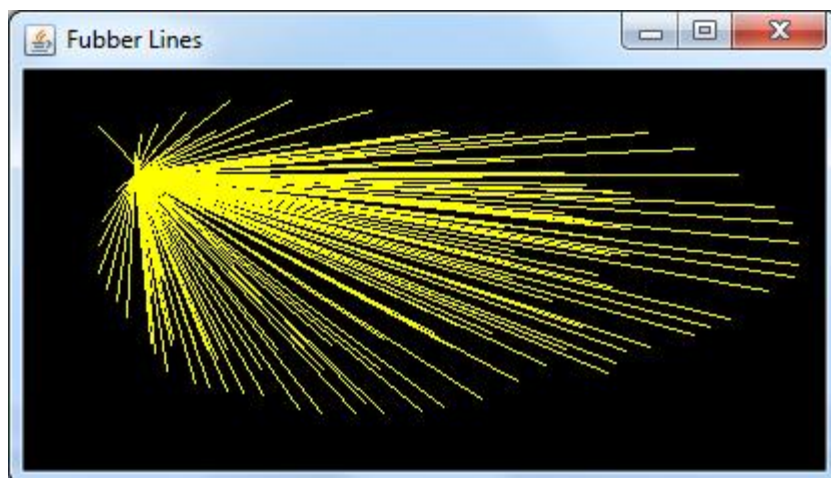
Modify the RubberLines program from this chapter so that it shows all of the lines drawn. Show only the final lines (from initial mouse press to mouse release), not the intermediate lines drawn to show the rubberbanding effect. Hint: Keep track of a list of objects that represent the lines similar to how the Dots program kept track of multiple dots.

How to Solve:

The easiest method I could think of to solve this problem is to add a method description to the mouseDragged() {} function that will show all the endpoints that I was dragging to until the mouseReleased () {} function was called from the listener.

Screenshots:

Running my pointer around:



End Point(note realized I had Fubber instead of Rubber) fixed it:

