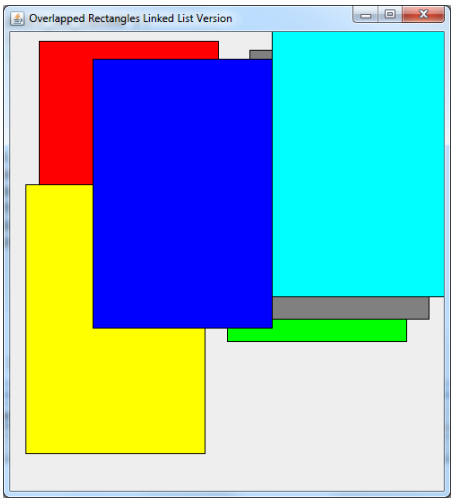
CPW 245 Summer 2018, Chapter 15, Homework Project 5

Main Idea:

Using a linked list, write the **`moveToTop`** method for the class named **`OverLappedRectangles`** that models a list of possibly overlapping rectangular two-dimensional window regions, like the windows for the programs open on your computer.



Details:

You are to complete the class **`OverLappedRectangles.java`**. You are provided classes named **`WindowsTester.java`**, **`RectangleNode.java`**, **`ColorRectangle.java`**, **`Rectangle.java`**, **`Point.java`**, and starter code for **`OverLappedRectangles.java`**.

`WindowsTester` contains the method **`main`** and instantiates a **`JFrame`** to show what is happening with your code. Mouse clicks in the **`JFrame`** are sent to your class, to determine which rectangle should be moved to the top. To run a test on your program, all you do is click points in the **`JFrame`**. Each time you do so, the GUI is updated according to your code for **`moveToTop`**. The class **`WindowsTester`** displays the rectangles graphically. It is provided by Ken Meerdink, so you can guess how reliable it is. It is not at all robust, so occasionally no rectangles will appear at all. If that happens, close the **`JFrame`** and re-launch the program until they do appear.

`RectangleNode` is a class that has a **`ColorRectangle`** as its data field and a pointer to the next **`RectangleNode`**. **`OverLappedRectangles`** uses these nodes to form a linked list of **`ColorRectangle`**s.

`ColorRectangle` is a class derived from **`Rectangle`** and extends it by adding a **`Color`** field for the GUI display. Your **`OverLappedRectangles`** class will have a field consisting of a reference to a **`RectangleNode`** named **`bottom`** that refers to a **`LinkedList`** of **`ColorRectangle`**s.

The class named **`Point`** is as we have used it before. See page 546 of *Building Java Programs* or the Practice It! website for description of **`Point`** class if you need further documentation.

Your rectangle list class, **`OverLappedRectangles`**, will keep a linked list of **`RectangleNode`**s to implement the model of possibly overlapping rectangular two-dimensional window regions.

Your rectangle list class will have a method, **`moveToTop`**, that takes a **`Point`** as a parameter, treats it as though the user clicked that **`Point`** on the screen, and moves the topmost rectangle touching that **`Point`** to the front of the list, leaving the order of the rest of the list unchanged. If the **`Point`** clicked is not on any of the rectangles, nothing is moved. To make your **`moveToTop`** method work correctly, you will need to determine which rectangle needs to be moved to the top by examining the coordinates of the **`Point`** passed to your method.

Important Points:

1. The **moveToTop** method cannot use an array or **ArrayList** or any collection other than the linked list that is already in the class **OverLappedRectangles**. You must use linked list techniques only to solve this problem. Implement the **moveToTop** method using only pointer manipulation or the **add/remove** methods
2. **Extra Credit (10 %)** Use pointer manipulation only, not **add/remove**. You may use additional **ColorRectangle** references as needed, but there is no need to allocate any additional memory to do this assignment.

3. Your code should work no matter how many or how few rectangles are on the screen. Test various numbers of rectangles by commenting out or uncommenting the addition of rectangles in **WindowsTester**.

```
olr.addRect( new ColorRectangle( 40, 40, 200, 300, Color.RED  ) );
olr.addRect( new ColorRectangle( 100, 60, 200, 300, Color.BLUE ) );
olr.addRect( new ColorRectangle( 25, 200, 200, 300, Color.YELLOW ) );
olr.addRect( new ColorRectangle( 250, 75, 200, 300, Color.GREEN ) );
//olr.addRect( new ColorRectangle( 275, 50, 200, 300, Color.GRAY ) );
//olr.addRect( new ColorRectangle( 300, 25, 200, 300, Color.CYAN ) );
```

4. You do need to comment your method so that it is easily understood by anyone in this class.

5. You don't need to java doc any of the methods provided to you, but you do need to javadoc any method you add to the class.

6. Be sure to write the header comment and document the **moveToTop** method to reflect your contribution.

Submit to Canvas: OverLappedRectangles.java