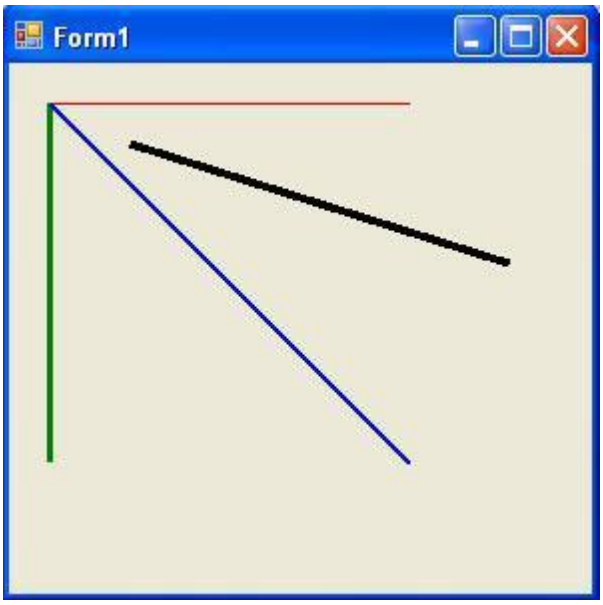
Composition and Encapsulation – The Line and Point Class

50 points

See Canvas for due date



Points and Lines

Computer graphics systems make use of two general rendering systems – rasters and vectors -- to produce images. Raster systems keep track of collections of individual pixels and vector systems keep track of lines. (Lines are more accurate, but require more computer resources.)

This assignment calls for a **Line** object that represents a graphically depicted image on the screen. This object must store this data:

- Endpoint objects (x and y coordinates as an integer). There will be two of these.
- The line color as a String
- The line width as an integer

Note that the Line object must be composed of two Point objects.

The Point Class

Create a **Point** class that contains x and y coordinates (x and y are integers) that represents a point in two-dimensional space. A point can lie only in the first quadrant in the x-y coordinate axis.

Include the following methods

- two constructors: a default that initializes x and y to 0, and another that accepts the x and y values as passed-in parameters
- get and set methods to retrieve and assign coordinate values
- a **toString** method that will return a String containing the coordinates of the point formatted as (x, y)

 an **equals** method (returns a boolean value) that compares two Points for equality. This method takes a single parameter of type Object and will cast the Object into a Point for comparison.
 anything else you deem necessary

The Line Class

Create a **Line** class that contains variables and/or methods to represent

- the end Points of the line (be sure your Point objects are declared as private)
- the length of the line (represent this as a return value from a method)
- color
- width
- anything else you deem necessary

This **Line** class should have the following methods

- A default constructor that creates Points that represent a zero-length black line at the origin with a width of 1 pixel
- A constructor that accepts four ints – x and y for point 1 and x and y for point 2, a color and a width
- A constructor that accepts four ints (color defaults to black and width to 1 pixel)
- get and set methods to retrieve values and assign values as you deem necessary
- A **toString** method that prints the coordinates of the line endpoints, the length, the color and width of the line (nicely formatted, please)
- an **equals** method (returns a boolean value) that will compare two lines for equality. Two lines are equal if their endpoints, width and color are the same. Note that Points do not need to be in same order. Be sure and use the equals method you wrote from your Point class to compare Points. Your equals method should override the equals method in Object.
- a **validateLine** method that will make sure the points are in the first quadrant. This method must be declared as static -- so that it can be called without an instance of the class.
- anything else you deem necessary

Create a driver class called **LineDriver** that will work with two lines. Create both lines as zero-length at the origin. Then provide a menu that is displayed repetitively with the following options:

1 - Enter coordinates, width and color for first line (line should be validated as part of this process -- use the validateLine method -- do not continue until valid coordinates and width are obtained from the user)
2 - Enter coordinates, width and color for second line (ditto)
3 - Compare the two lines (specify if lines are equal/not equal after comparison)
4 - Display coordinates, width, length and color for first line (the length should be derived from the coordinates using the distance formula and is represented as a real number with 2 post-decimal positions of precision.)
5 - Display coordinates, width, length and color for second line (the length should be derived from the coordinates using the distance formula and is represented as a real number with 2 post-decimal positions of precision.)
6 - Quit

Turn in all source code in a zip file named with your last name, followed by the first initial of your first name, followed by hw7 (ex: peterschw7.zip) Submit to Canvas.
Get started ASAP!