BJP5 Exercise 8.1: quadrantPoint

Language/Type: Java classes if/else instance methods Point
Author: Marty Stepp (on 2019/09/19)

Add the following method to the `Point` class:

```
public int quadrant()
```

Returns which quadrant of the x/y plane this `Point` object falls in. Quadrant 1 contains all points whose x and y values are both positive. Quadrant 2 contains all points with negative x but positive y. Quadrant 3 contains all points with negative x and y values. Quadrant 4 contains all points with positive x but negative y. If the point lies directly on the x and/or y axis, return 0.

```
public class Point {
    private int x;
    private int y;

    // // your code goes here

}
```

Type your solution here:

```
public int quadrant(){

    if (x > 0 && y > 0)
        return 1;

    else if (x < 0 && y < 0)
        return 3;

    else if (x > 0 && y < 0)
        return 4;

    else if (x < 0 && y > 0)
        return 2;

    else
        return 0;

}
```

This is a **partial class problem.** Submit code that will become part of an existing Java class as described. You do not need to write the complete class, just the portion described in the problem.

4 Indent

Sound F/X
Highlighting

Submit

You passed 7 of 7 tests.

Go to the next problem: flipPoint

test #1: (81, 21)
console output: 1
result: pass

BJP5 Exercise 8.2: flipPoint

Language/Type: Java classes instance methods Point
Author: Marty Stepp (on 2019/09/19)

Add the following method to the `Point` class:

```
public void flip()
```

Negates and swaps the x/y coordinates of the `Point` object. For example, if the object initially represents the point (5
to `flip`, the object should represent (3, -5). If the object initially represents the point (4, 17), after a call to `flip`, the o
represent (-17, -4).

```
public class Point {
    private int x;
    private int y;

    // // your code goes here

}
```

Type your solution here:

```
1 public void flip(){
2     int c = x;
3     x = -y;
4     y = -c;
5 }
```

This is a **partial class problem.** Submit code that will become part of an existing Java class as described. You do not need to write the complete class, just the portion described in the problem.

Submit

You passed 5 of 5 tests.

Go to the next problem: manhattanDistancePoint

test #1:	(81, 21)
console output:	(-21, -81) (81, 21)
result:	pass

BJP5 Exercise 8.3: manhattanDistancePoint

Language/Type: Java classes instance methods Point
Author: Marty Stepp (on 2019/09/19)

Add the following method to the `Point` class:

```
public int manhattanDistance(Point other)
```

Returns the "Manhattan distance" between the current `Point` object and the given other `Point` object. The M
refers to how far apart two places are if the person can only travel straight horizontally or vertically, as though
Manhattan. In our case, the Manhattan distance is the sum of the absolute values of the differences in their cc
words, the difference in x plus the difference in y between the points.

```
public class Point {
    private int x;
    private int y;

    // // your code goes here

}
```

Type your solution here:

```
1 public int manhattanDistance(Point other){
2     int a = Math.abs(x - other.x);
3     int b = Math.abs(y - other.y);
4     return a + b;
5
6 }
```

This is a **partial class problem.** Submit code that will become part of an existing Java class as described. You do not need to write the co
class, just the portion described in the problem.

Submit

You passed 4 of 4 tests.

Go to the next problem: isVerticalPoint

test #1:	(5, 2) to (8, 6)
console output:	7
result:	pass

BJP5 Self-Check 8.4: ReferenceMystery3

Language/Type: Java Objects Point reference semantics
Author: Marty Stepp (on 2019/09/19)

The following program produces 4 lines of output. Write each line of output below as it would appear on the conso

```
public class ReferenceMystery3 {
    public static void main(String[] args) {
        int a = 7;
        int b = 9;
        Point p1 = new Point(2, 2);
        Point p2 = new Point(2, 2);
        addToXTwice(a, p1);
        System.out.println(a + " " + b + " " + p1.x + " " + p2.x);
        addToXTwice(b, p2);
        System.out.println(a + " " + b + " " + p1.x + " " + p2.x);
    }

    public static void addToXTwice(int a, Point p1) {
        a = a + a;
        p1.x = a;
        System.out.println(a + " " + p1.x);
    }
}
```

line 1	14 14
line 2	7 9 14 2
line 3	18 18
line 4	7 9 14 18

Submit

You passed 4 of 4 tests.

Go to the next problem: CalculatorObject

#	question	your answer	result
1	line 1	14 14	pass
2	line 2	7 9 14 2	pass
3	line 3	18 18	pass
4	line 4	7 9 14 18	pass

BJP5 Self-Check 8.11: getNormalReverseOrderName

Language/Type: Java classes implementing instance methods
Author: Marty Stepp (on 2019/09/19)

(You should complete Self-Check 8.7 before answering this question.)

Add two new methods to the `Name` class:

```
public String getNormalOrder()
```

Returns the person's name in normal order, with the first name followed by the middle initial and last name. For exar
name is `"John"`, the middle initial is `'Q'`, and the last name is `"Public"`, this method returns `"John Q. Public'`

```
public String getReverseOrder()
```

Returns the person's name in reverse order, with the last name preceding the first name and middle initial. For exam
name is `"John"`, the middle initial is `'Q'`, and the last name is `"Public"`, this method returns `"Public, John Q`

(You don't need to write the class header or declare the fields; assume that this is already done for you. Just write yc
complete code in the box provided.)

Type your solution here:

```
1 public String getNormalOrder(){
2     return (firstName + " " + middleInitial + ". " + lastName);
3 }
4
5 public String getReverseOrder(){
6     return (lastName + ", " + firstName + " " + middleInitial + ".");
7 }
```

This is a **partial class problem.** Submit code that will become part of an existing Java class as described. You do not need to write the complete class, just the portion described in the problem.

Submit

You passed 2 of 2 tests.

Go to the next problem: printableObjects

test #1: `getNormalOrder`