

CSE 11

Accelerated Intro to Programming

Lecture 7

Greg Miranda, Spring 2021

Announcements

- Quiz 7 due Wednesday @ 8am
- PA2 due Wednesday @ 11:59pm
- Survey 3 due Friday @ 11:59pm

New Point Method

- Last time...
 - Used the Point class
 - Wrote simple method quadrant()
 - No parameters
 - Just used information about the point to return a String representing what quadrant it was in
- Different method (for you to try...)
 - Write a method called add()
 - Take an existing Point and another Point and add their x and y values together to get a new Point
 - Let's do a few steps together...

- Method header:
 - What is the header for this method going to look like?
- Examples:
 - Let's write some examples...
- Take a few minutes and try and write the body of the method

New Class

- Try a new class, a new idea
 - Besides just x & y points
- Another geometry example
 - Useful to have pictures we can draw that correspond to the class
 - Another idea with a coordinate plane
 - Want to have a class that represents lines
 - What are some ways we represent lines?

- What are some methods we might want to write on a line?
 - One idea - like a calculator
 - We could provide the x value and get the y value back
 - A natural thing to want to compute about a line
 - Or about any 2D function
- Examples of lines
- What will this method look like?
 - Calculate a y value given an x value

Double

- One of the things we had issues with integers and division
 - `int n = 15 / 2;`
 - We get truncation
- Java, and most programming languages, have a way to use a different kind of arithmetic
 - `double m = 15.0 / 2.0;`
 - Does fractional arithmetic
 - A different type
 - double – floating point number

- For most of our purposes – we can trust that doubles will be pretty accurate
 - We should only use doubles to represent things where we are okay with inaccuracy
 - The way that computers can round numbers can be surprising
 - `double oneThird = 1.0 / 3.0;`
 - `double twoThirds = 2.0 / 3.0;`
 - Doesn't round off at the end
 - `double anotherOne = (0.1 + 0.2) + 0.3;`
 - The internal representation of these numbers isn't perfect
 - Will learn all these reasons in great detail if you take CSE 30
 - There is some rounding happening even on simple cases
 - `double yetAnother = 0.1 + (0.2 + 0.3);`
 - So weird the order of parenthisation can matter

- Will start using doubles now as another data type
 - Just be aware: when we use them we are expecting some kind of rounding behavior
- How to mix doubles and ints?
 - `double divided = 15 / 2;`
 - 7.5?
 - `double dividedAgain = n / 2;`
 - How do we get the right answer?
- This is going to be important to us going forward
 - To be able to use doubles
 - Able to turn ints into calculations we can do with doubles

- Let's look at a few ways to manipulate numbers using more built-in methods in Java
 - Like built-in String methods we looked at before
- Square root of a number - common operation to do
 - `double sqrt2 = Math.sqrt(2);`
 - Takes an int or a double
 - `double sqrt2FromDouble = Math.sqrt(2.0);`
 - Answer is always a double
 - An approximation of the square root – not a full answer to the square root
- Raise a number to a power
 - `double cubeOf12 = Math.pow(12, 3);`
- Both methods are defined in Java's Math library

- More math methods
 - Max
 - `double maxOf45 = Math.max(4, 5);`
 - Min
 - And several other math methods as well
- Two ways to think of this based on what we've seen before
 - Definition 1
 - Math is a built-in object
 - Definition 2
 - Math is a built-in class
 - `sqrt`, `pow`, `max`, `min` are a special kind of a method
 - Calling them with the class name before the dot
 - Instead of writing an object before the dot
 - Defn2 is the correct way to think about it
 - Another feature called **static methods** that's coming up in future weeks

Memory Models

- More practice with drawing diagrams for laying out objects
 - Build up a little more of a visual language for
 - Drawing objects
 - Drawing what's happening inside Java
- Code from the reading

```
class Point {
    int x;
    int y;
    Point(int x, int y) {
        this.x = x;
        this.y = y;
    }
    double distance(Point other) {
        return Math.sqrt(Math.pow(this.x - other.x, 2)
            + Math.pow(this.y - other.y, 2));
    }
}
class CircRegion {
    Point center;
    int radius;
    CircRegion(Point center, int radius) {
        this.center = center;
        this.radius = radius;
    }
    boolean contains(Point p) {
        return this.center.distance(p) < this.radius;
    }
}
class ExamplesRegion {
    CircRegion c1 = new CircRegion(new Point(200, 50), 10);
    Point circleTest1 = new Point(209, 50);
    boolean contains1 = this.c1.contains(this.circleTest1);
}
```

```

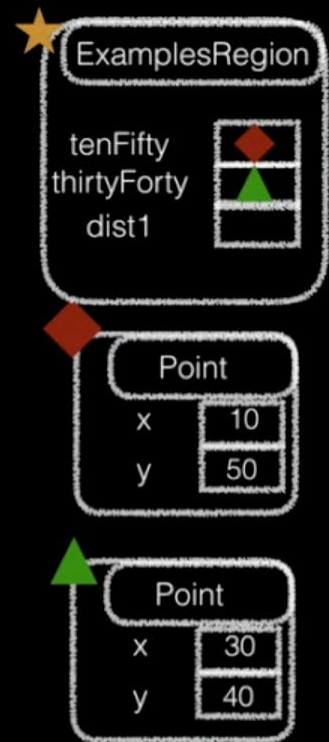
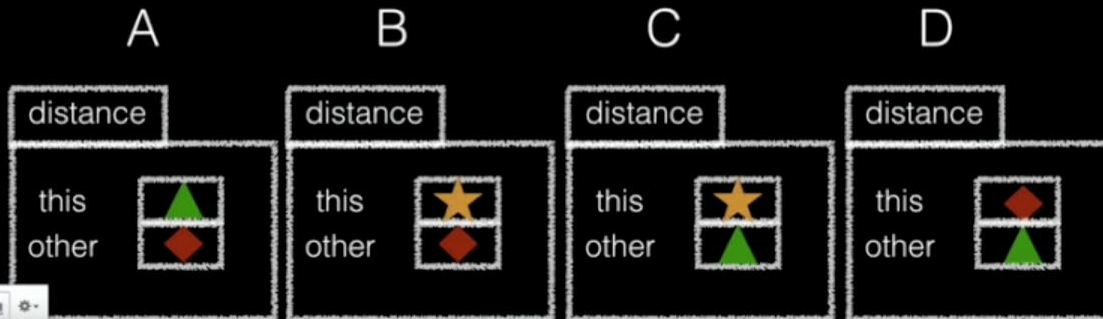
class Point {
    int x;
    int y;
    Point(int x, int y) {
        this.x = x;
        this.y = y;
    }
    double distance(Point other) {
        return Math.sqrt(Math.pow(this.x - other.x, 2)
            + Math.pow(this.y - other.y, 2));
    }
}

class ExamplesRegion {
    Point tenFifty = new Point(10, 50);
    Point thirtyForty = new Point(30, 40);

    double dist1 = this.tenFifty.distance(this.thirtyForty);
}

```

Which of these is the stack frame that's created for the call to distance?



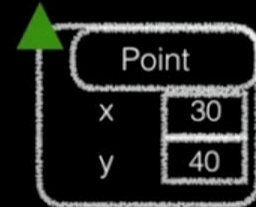
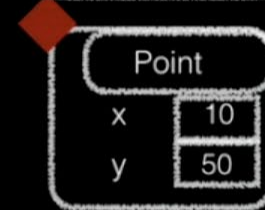
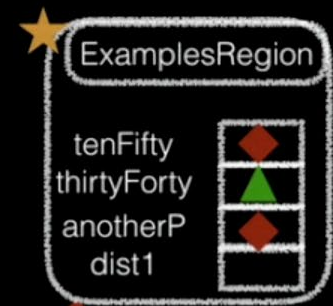
```

class Point {
    int x;
    int y;
    Point(int x, int y) {
        this.x = x;
        this.y = y;
    }
    double distance(Point other) {
        return Math.sqrt(Math.pow(this.x - other.x, 2)
            + Math.pow(this.y - other.y, 2));
    }
}

class ExamplesRegion {
    Point tenFifty = new Point(10, 50);
    Point thirtyForty = new Point(30, 40);
    Point anotherP = tenFifty;

    double dist1 = this.tenFifty.distance(this.anotherP);
}

```



Which of these is the stack frame that's created for the call to distance?

