

CSE 11

Accelerated Intro to Programming

Lecture 8

Greg Miranda, Spring 2021

Announcements

- Quiz 8 due Friday @ 8am
- PA2 due tonight @ 11:59pm
- Survey 3 due Friday @ 11:59pm
- PA0.5 Resubmission due Friday @ 11:59pm
 - Or see a tutor during lab hours and demo it

Math

- 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

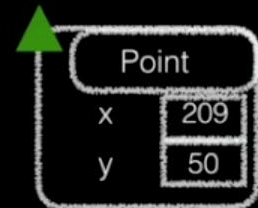
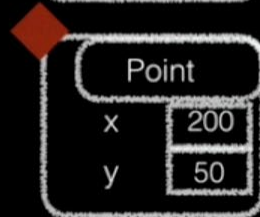
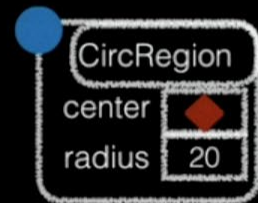
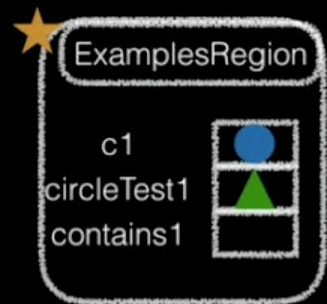
```
class Point {
    int x;
    int y;
    Point(int x, int y) {
        this.x = x;
        this.y = y;
    }
    double distance(Point other) {
        return Math.sqrt(Math.pow(this.x - other.x, 2)
            + Math.pow(this.y - other.y, 2));
    }
}
class CircRegion {
    Point center;
    int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

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

```
    boolean contains(Point p) {  
        return this.center.distance(p) < this.radius;  
    }  
}
```

```
    this.c1.contains(this.circleTest1);
```



```

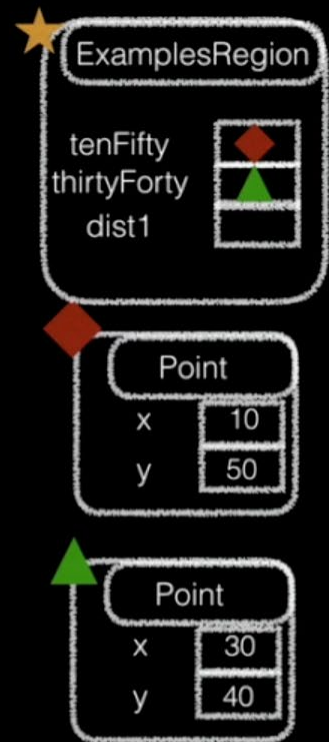
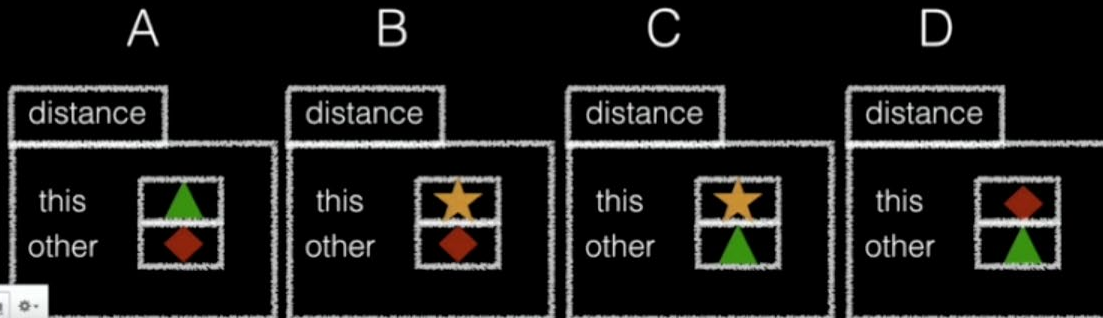
class Point {
    int x;
    int y;
    Point(int x, int y) {
        this.x = x;
        this.y = y;
    }
    double distance(Point other) {
        return Math.sqrt(Math.pow(this.x - other.x, 2)
            + Math.pow(this.y - other.y, 2));
    }
}

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?




```

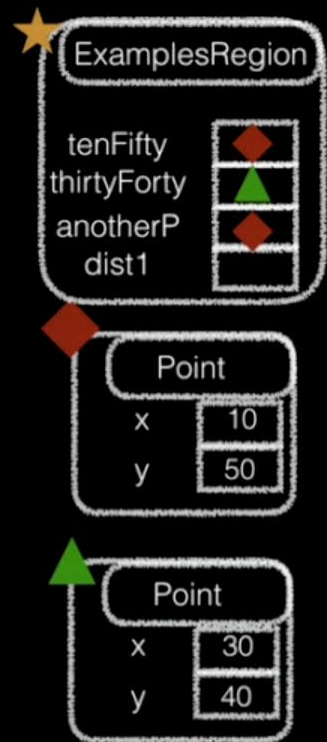
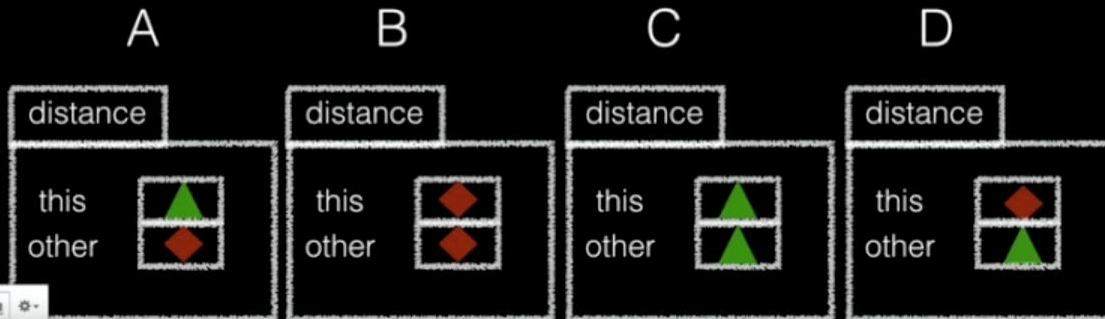
class Point {
    int x;
    int y;
    Point(int x, int y) {
        this.x = x;
        this.y = y;
    }
    double distance(Point other) {
        return Math.sqrt(Math.pow(this.x - other.x, 2)
            + Math.pow(this.y - other.y, 2));
    }
}

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

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

```

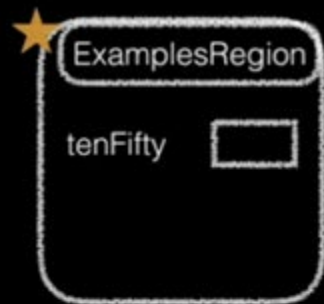
Which of these is the stack frame that's created for the call to distance?



Constructors

- Now that we understand the Stack, we have what we need to understand constructors

```
class Point {  
    int x;  
    int y;  
    Point(int x, int y) {  
        this.x = x;  
        this.y = y;  
    }  
  
}  
  
class ExamplesRegion {  
    Point tenFifty = new Point(10, 50);  
}
```

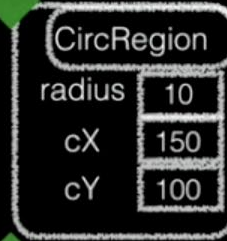


Constructor Summary

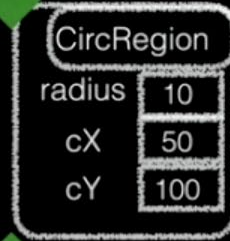
- Constructors:
 - Are special methods, called when **new** is used
 - Are passed the newly-constructed object as **this**, and any arguments
 - Typically assign values into fields using **this.field = value**
- When new is used:
 - A fresh object, with a new reference is created with uninitialized fields
 - The constructor with parameters that match the arguments is called
 - The whole new expression evaluates to the new reference

```
class CircRegion2 {  
    int radius;  
    int cX;  
    int cY;  
    CircRegion2(int radius, int cX, int cY) {  
        this.radius = radius;  
        this.cX = cX + 100;  
        this.cY = cY;  
    }  
}  
class ExamplesRegion {  
    CircRegion2 cr2 = new CircRegion(10, 50, 100);  
}
```

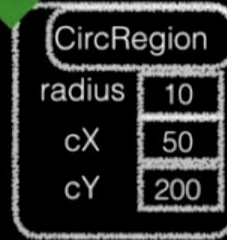
A



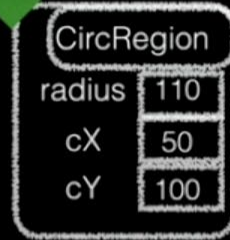
C



B

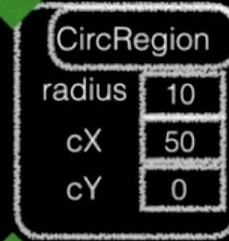


D

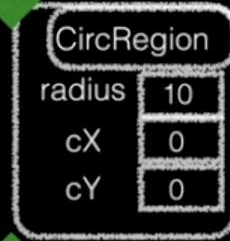


```
class CircRegion2 {  
    int radius;  
    int cX;  
    int cY;  
    CircRegion2(int radius) {  
        this.radius = radius;  
        this.cX = 0;  
        this.cY = 0;  
    }  
}  
class ExamplesRegion {  
    CircRegion2 cr2 = new CircRegion(10);  
}
```

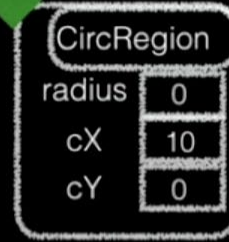
A



C



B



D

