CSE 11
Accelerated Intro to Programming
Lecture 6

Greg Miranda, Summer 1 2021

This lecture is being recorded

Announcements

- PA3 due Thursday @ 11:59pm
- Quiz 3 released today @ 11am
  - Due Friday @ 11:59pm

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

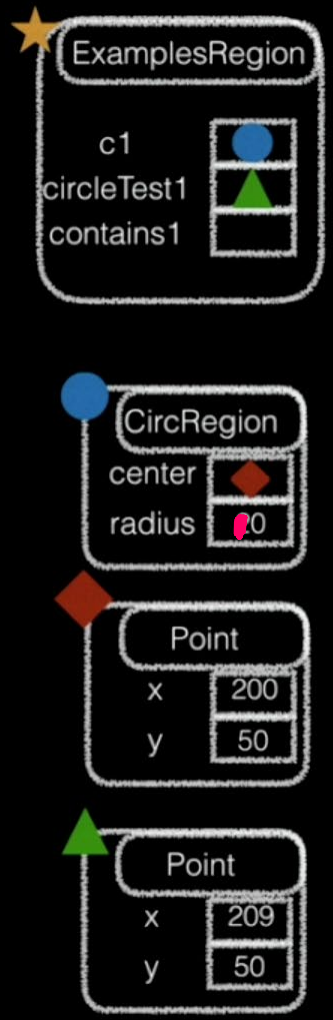
Memory → 2 parts

stack → parameters, variables, fields

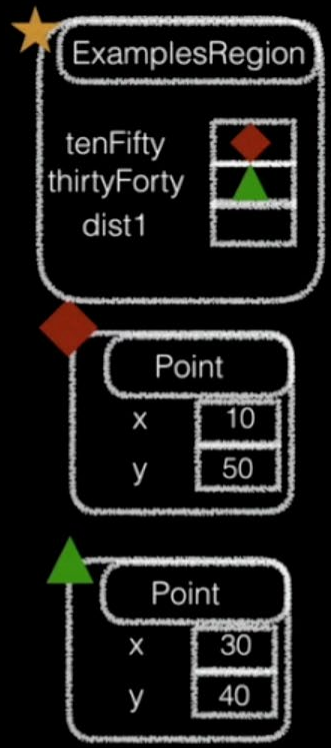
heap → objects

```
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);
}
```

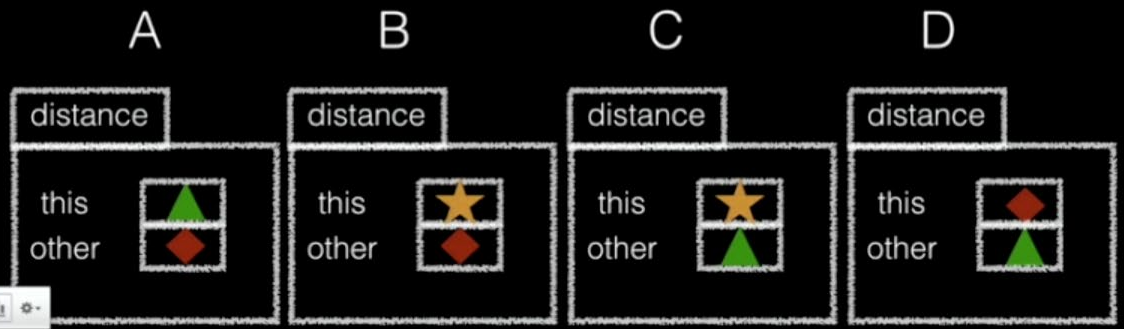
```
class Point {


  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);
```



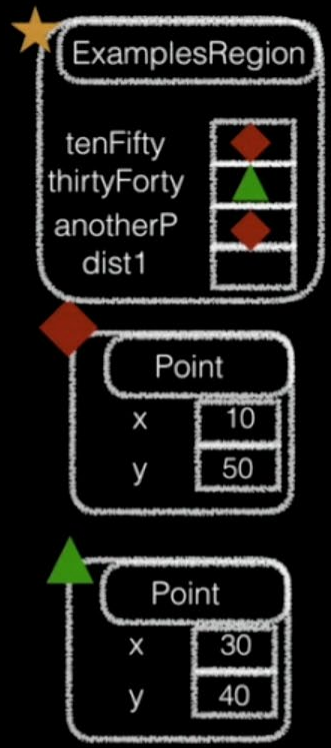
```
class Point {
  int x;
  int y;
  Point(int x, int y) {
    this.x = x;
    this.y = y;
  }
  double distance(Point other) {
    return Math.sqrt(Math.pow(this.x - other.x, 2)
      + Math.pow(this.y - other.y, 2));
  }
}
class ExamplesRegion {
  Point tenFifty = new Point(10, 50);
  Point thirtyForty = 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?

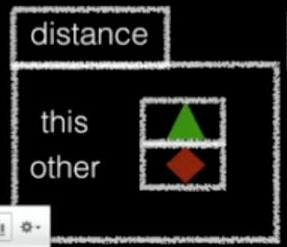


```
class Point {
  int x;
  int y;
  Point(int x, int y) {
    this.x = x;
    this.y = y;
  }
  double distance(Point other) {
    return Math.sqrt(Math.pow(this.x - other.x, 2)
      + Math.pow(this.y - other.y, 2));
  }
}
class ExamplesRegion {
  Point tenFifty = new Point(10, 50);
  Point thirtyForty = new Point(30, 40);
  Point anotherP = tenFifty;

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

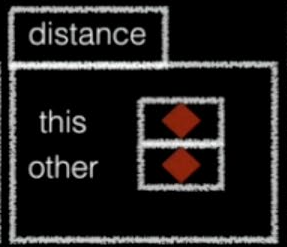


Which of these is the stack frame that's created for the call to distance?

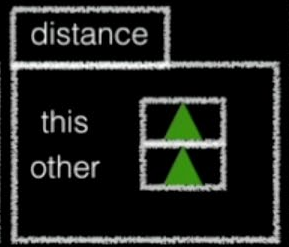
A



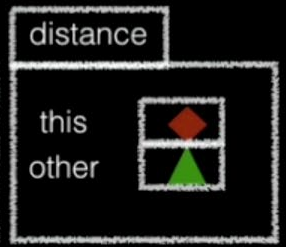
B



C



D



Constructors

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

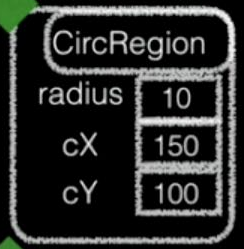
```
class Point {
  int x;
  int y;
  Point(int x, int y) {
    this.x = x;
    this.y = y;
  }



}
class ExamplesRegion {
  Point tenFifty = new Point(10, 50);

}
```

Constructor Summary

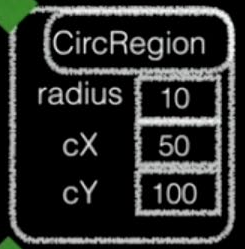
- Constructors:
  - Are special methods, called when **new** is used
  - Are passed the newly-constructor object as **this**, and any arguments
  - Typically assign values into fields using **this.field = value**
- When new is used:
  - A fresh object, with a new reference is created with uninitialized 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);

}
```

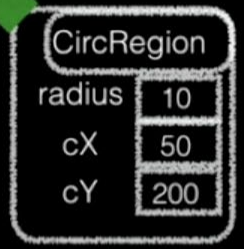
A



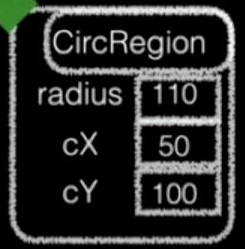
C



B

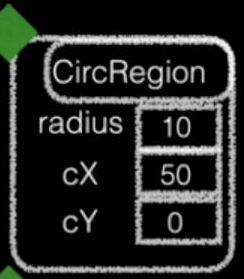


D

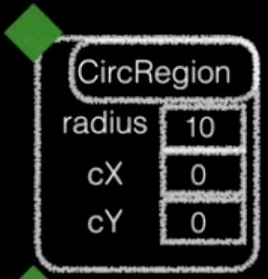


```
class CircRegion2 {
  int radius;
  int cX;
  int cY;
  CircRegion2(int radius) {
    this.radius = radius;
    this.cX = 0;
    this.cY = 0;
  }
}
class ExamplesRegion {
  CircRegion2 cr2 = new CircRegion(10);

}
```

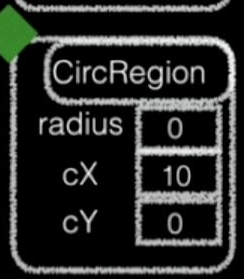
A



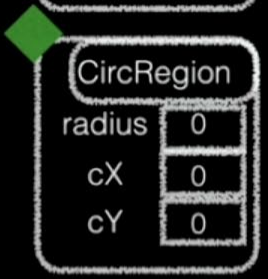
C



B



D



Tester

- import tester.*;
  - tester.jar – java archive
    - Libraries that contain classes that we can use in our own code
      - Tester
- Tester class allows us to create methods to unit test our code
  - Unit testing – compare actual values versus expected values
    - t.checkExpect(<actual value>, <expected value>);
  - Goal: get all tests to pass
    - Confidence that your code/solution is correct

Local Variables

- Local variables are defined inside the body of a method
  - They are ‘local’ to the method in which they are defined in
- Used temporarily while the method is running, then are removed
  - Similar to parameters
  - Added to the stack frame for the method
- No default value
  - Must be assigned a value before it’s read from
    - i.e. used as an expression