CSC 212
PROGRAMMING WITH DATA STRUCTURES

SPRING 2016

PROF. SARA SHEEHAN

SMITH COLLEGE

CLASS 5: FEB 9 OUTLINE

- **Static example**
- **Recap Inheritance and Polymorphism**
  - Debrief Lab 2
- **Graphics and tips for Homework 2**
- **Start Interfaces (activity)**
- **Honor Code interlude**
- **Debrief Homework 1**
  - Demo
- **Demo Eclipse debugging (if time!)**

RECAP INHERITANCE

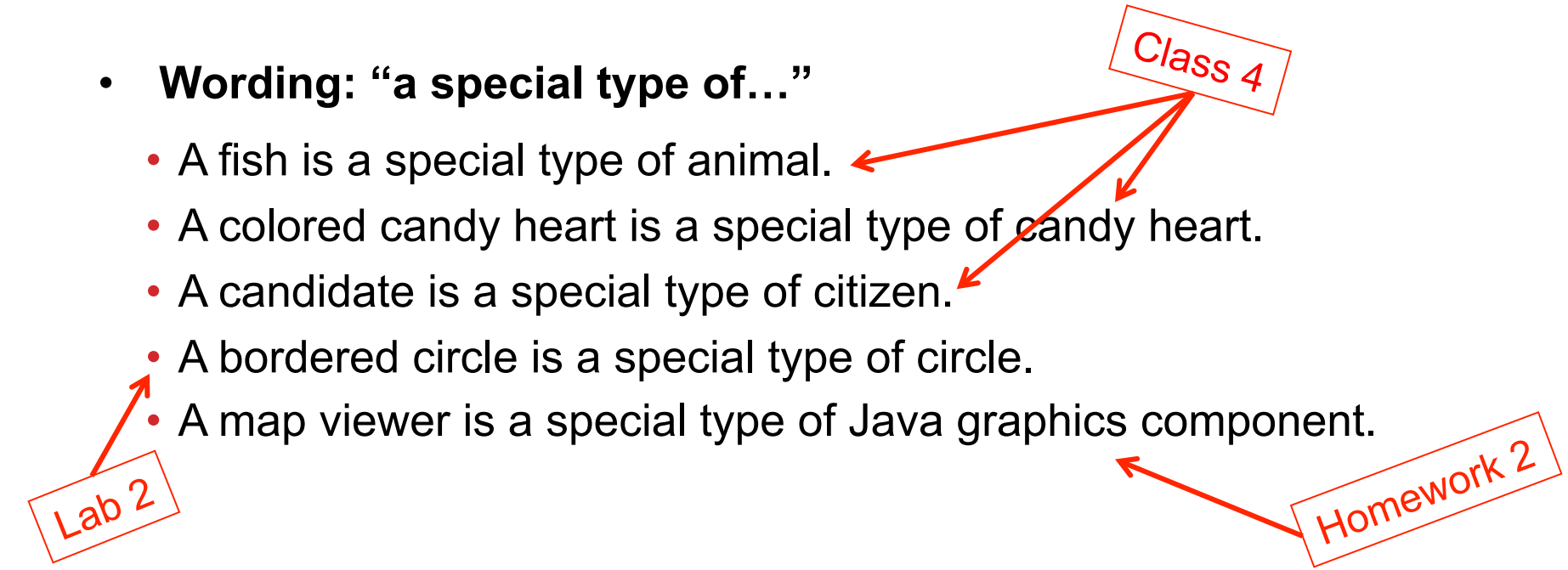
- **Inherit: existing methods and fields**
- **Augment: add new methods and fields**
- **Override: methods (not fields)**

RECAP INHERITANCE

- **Inherit: existing methods and fields**
- **Augment: add new methods and fields**
- **Override: methods (not fields)**
- **Wording: “a special type of…”**
  - A fish is a special type of animal.
  - A colored candy heart is a special type of candy heart.
  - A candidate is a special type of citizen.
  - A bordered circle is a special type of circle.
  - A map viewer is a special type of Java graphics component.

RECAP INHERITANCE

- **Inherit: existing methods and fields**
- **Augment: add new methods and fields**
- **Override: methods (not fields)**
- **Wording: "a special type of…"**
  - A fish is a special type of animal.
  - A colored candy heart is a special type of candy heart.
  - A candidate is a special type of citizen.
  - A bordered circle is a special type of circle.
  - A map viewer is a special type of Java graphics component.



RECAP INHERITANCE

```
public class Animal {

    public String location;

    public Animal(String location) {
        this.location = location;
    }

    public void move(String newLoc) {
        this.location = newLoc;
        System.out.println("animal move");
    }
}
```

RECAP INHERITANCE

```
public class Animal {

    public String location;

    public Animal(String location) {
        this.location = location;
    }

    public void move(String newLoc) {
        this.location = newLoc;
        System.out.println("animal move");
    }
}
```

```
public class Fish extends Animal {

    public int depth;

    public Fish(String location, int depth) {
        super(location);
        this.depth = depth;
    }

    public void move(String location) {
        System.out.println("fish move");
        super.move(location);
    }
}
```

RECAP INHERITANCE

```
public class Animal {

    public String location;

    public Animal(String location) {
        this.location = location;
    }

    public void move(String newLoc) {
        this.location = newLoc;
        System.out.println("animal move");
    }
}
```

```
public class Fish extends Animal {

    public int depth;

    public Fish(String location, int depth) {
        super(location);
        this.depth = depth;
    }

    public void move(String location) {
        System.out.println("fish move");
        super.move(location);
    }
}
```

Q: What does this print?

```
public class AnimalApp {

    public static void main(String[] args) {
        Animal myFish = new Fish("northampton", 2);
        myFish.move("springfield");
    }
}
```

RECAP INHERITANCE

```
public class Animal {

    public String location;

    public Animal(String location) {
        this.location = location;
    }

    public void move(String newLoc) {
        this.location = newLoc;
        System.out.println("animal move");
    }
}
```

```
public class Fish extends Animal {

    public int depth;

    public Fish(String location, int depth) {
        super(location);
        this.depth = depth;
    }

    public void move(String location) {
        System.out.println("fish move");
        super.move(location);
    }
}
```

Q: What does this print?

```
public class AnimalApp {

    public static void main(String[] args) {
        Animal myFish = new Fish("northampton", 2);
        myFish.move("springfield");
    }
}
```

A:

```
fish move
animal move
```

LAB 2 (JCIRCLES)

GRAPHICS ESSENTIAL CLASSES

- JFrame: represents and controls a window
- Pane: content area within window (JFrame)
- JComponent: maintains and draws region inside Pane
- Graphics: class with all the methods for drawing
- Color: already used a lot, represents RGB color
- Dimension: encapsulates width and height of component

INTERFACES

INTERFACES

- One of my favorite things in Java
- Similar concept as inheritance, but with a more abstract “super class”
- Inheritance is very flexible, Interfaces are a very strict contract