

CS 213 : Software Methodology

Spring 2019

Sesh Venugopal

Lecture 2: Jan 24
OOP – Inheritance

Inheritance

```
public class Point {  
    int x,y;  
}
```

superclass **Point**



subclass **ColoredPoint**

```
public class ColoredPoint  
extends Point {  
    int x,y;  
}
```

subclass **ColoredPoint** inherits
x and **y** from superclass **Point**

What this means is **x** and **y** are fields
in **ColoredPoint**, without the programmer
having to write them in (CODE REUSE)

```
Point p = new Point(); // OK, x and y in instance p are zero
```

```
ColoredPoint cp =  
    new ColoredPoint(); // OK, x and y in instance cp are zero
```

Inheritance

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

```
Point p = new Point(3,4); // OK, p is (3,4)
```

```
public class ColoredPoint  
extends Point {  
  
}
```

Will this class compile?

NO

“Implicit super constructor Point() is undefined for default constructor.
Must define an explicit constructor.”

Inheritance – Subclass constructor

```
public class ColoredPoint
extends Point {
    int x,y;
    public ColoredPoint() {
        super();
    }
}
```

Default constructor

Calls superclass's constructor

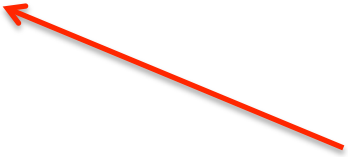
The FIRST statement in a subclass constructor should invoke a superclass constructor. (Or it should invoke another constructor in the class, with `this(...)`).

A default constructor will ALWAYS CALL a superclass no-arg constructor

Problem: the `Point` class does not have a no-arg constructor!

Inheritance – Subclass constructor

```
public class ColoredPoint
extends Point {
    int x,y;
    public ColoredPoint() {
        super();
    }
}
```



“Implicit super constructor Point() is undefined for default constructor.
Must define an explicit constructor.”

The **FIRST** statement in a subclass constructor - **ANY** constructor, not just the default - should invoke a superclass constructor. (Or it should invoke another constructor in the class, with **this(...)**).

Inheritance – Subclass constructor

```
public class ColoredPoint
extends Point {
    int x,y;
    String color;
    public ColoredPoint(int x, int y, String color) {
        super(x,y); ← Calls superclass's constructor
        this.color = color;
    }
}
```

Will the following alternative compile? **NO**

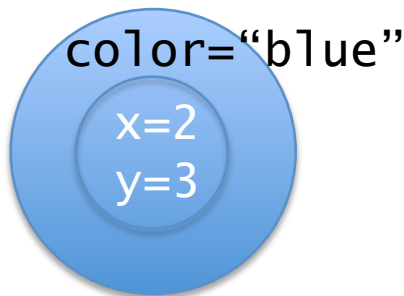
```
public ColoredPoint(int x, int y, String color) {
    super();
    this.x = x; this.y = y;
    this.color = color;
}
}
```

A `super()` call is introduced as the FIRST statement, but `Point` does not have a no-arg constructor

Inheritance – Why call super(...)?

Think of a subclass instance having two parts: the superclass part (inherited), and the additional subclass part

```
ColoredPoint cp =  
    new ColoredPoint(  
        2, 3, "blue");
```



Initialization of the superclass part is best done by a superclass constructor, no point in reinventing the wheel (Code REUSE)

Thus the call to the superclass constructor, to FIRST initialize the superclass part, then code to initialize the subclass part.

Q. When a `ColoredPoint` instance is created, is an inner `Point` instance created as well?

NO.

It's CODE reuse, not instance reuse

Inheritance – Fields and Methods

```
package geometry;
```

```
public class Point {  
    int x,y;  
    public Point(int x, int y) {  
        this.x = x; this.y = y;  
    }  
    public int getX() {  
        return x;  
    }  
    public int getY() {  
        return y;  
    }  
    public String toString() {  
        return x + "," + y;  
    }  
}
```

```
package geometry;
```

```
public class ColoredPoint  
    extends Point {  
    int x,y;  
    String color;  
    public ColoredPoint(  
        int x, int y, String color) {  
        super(x,y);  
        this.color = color;  
    }  
    public int getX() { return x; }  
    public int getY() { return y; }  
    public String toString() {  
        return x + "," + y;  
    }  
}
```

Constructor
inherited?

NO

WHY NOT?

Are we ok with
using this as is?

NO. Color should be included.

Inheritance – Overriding Method

```
package geometry;
```


```
public class ColoredPoint  
extends Point {  
    int x,y;
```

```
    String color;  
    public ColoredPoint(  
        int x, int y, String color) {  
        super(x,y);  
        this.color = color;  
    }
```

```
    public int getX() { return x; }  
    public int getY() { return y; }
```

This implementation overrides
the inherited code

```
    public String toString() {  
        return x + "," + y + "," + color;  
    }  
}
```



Inheritance – Reusing inherited method code in overriding Method

```
package geometry;

public class ColoredPoint
    extends Point {
    int x,y;

    String color;
    public ColoredPoint(
        int x, int y, String color) {
        super(x,y);
        this.color = color;
    }

    public int getX() { return x; }
    public int getY() { return y; }

    public String toString() {
        return x + "," + y + "," + color;
    }
}
```

super.toString() ← Reusing inherited method code in overriding method
Is good programming practice