More on Inheritance

```
// Create a super class.
class A {
  A() {
    System.out.println("Constructing A.");
  }
}

// Create a subclass by extending class A.
class B extends A {
  B() {
    System.out.println("Constructing B.");
  }
}

// Create another subclass by extending B.
class C extends B {
  C() {
    System.out.println("Constructing C.");
  }
}
```

When Do Constructors Execute?

```
class OrderOfConstruction {
    public static void main(String args[]) {
        C c = new C();
    }
}
```

What will the output be?

When Do Constructors Execute?

```
class OrderOfConstruction {
    public static void main(String args[]) {
        C c = new C();
    }
}
```

What will the output be?

```
Constructing A.
Constructing B.
Constructing C.
```

```
class X {
  int a;
  X(int i) { a = i; }
}

class Y {
  int a;
  Y(int i) { a = i; }
}

class IncompatibleRef {
  public static void main(String args[]) {
    X x = new X(10);
    X x2;
    Y y = new Y(5);

    x2 = x;     // OK, both of same type
    x2 = y;     // Error, not of same type
  }
}
```

Incorrect reference

```
// A superclass reference can refer to a subclass object.

class X {
  int a;

  X(int i) { a = i; }                  // class X constructor
}

class Y extends X {
  int b;

  Y(int i, int j) {                    // class Y constructor
    super(j);                          //   super class call
    b = i;
  }
}
```

(see main on next slide)

```
class SupSubRef {
  public static void main(String args[]) {
    X x = new X(10);
    X x2;
    Y y = new Y(5, 6);

    x2 = x;    // OK, both of same type
    System.out.println("x2.a: " + x2.a);

    x2 = y;    // still Ok because Y is derived from X
    System.out.println("x2.a: " + x2.a);

    // X references know only about X members
    x2.a = 19;      // OK
//    x2.b = 27;    // Error, X doesn't have a b member
  }
}
```

It is the type of the reference variable,

not the type of the object it refers to,

that determines what members can be accessed.

```
class TwoDShape {
  private double width;
  private double height;

  // A default constructor.
  TwoDShape() {
    width = height = 0.0;
  }

  // Parameterized constructor.
  TwoDShape(double w, double h) {
    width = w;
    height = h;
  }

  // Construct object with equal width and height.
  TwoDShape(double x) {
    width = height = x;
  }
```

(continued next slide)

```
  // Accessor methods for width and height.
   double getWidth() { return width; }
   double getHeight() { return height; }
   void setWidth(double w) { width = w; }
   void setHeight(double h) { height = h; }

   void showDim() {
     System.out.println("Width and height are " +
                        width + " and " + height);
   }
}
```

(next slide shows a sub class)

```
// A subclass of TwoDShape for triangles.
class Triangle extends TwoDShape {
  private String style;

  Triangle() {                  // a default constructor
    super();
    style = "none";
  }

                                // Constructor for Triangle.
  Triangle(String s, double w, double h) {
    super(w, h); // call superclass constructor
    style = s;
  }

                                // One argument constructor.
  Triangle(double x) {
    super(x); // call superclass constructor
    style = "filled";
  }
```

(continued on next slide)

```
                    // Construct an object from an object.
  Triangle(Triangle ob) {
    super(ob); // pass object to TwoDShape constructor
    style = ob.style;
  }

  double area() {
    return getWidth() * getHeight() / 2;
  }

  void showStyle() {
    System.out.println("Triangle is " + style);
  }
}
```

```
class Shapes7 {
  public static void main(String args[]) {
    Triangle t1 = new Triangle("outlined", 8.0, 12.0);

                                // make a copy of t1
    Triangle t2 = new Triangle(t1);

    System.out.println("Info for t1: ");
    t1.showStyle();
    t1.showDim();
    System.out.println("Area is " + t1.area());

    System.out.println();

    System.out.println("Info for t2: ");
    t2.showStyle();
    t2.showDim();
    System.out.println("Area is " + t2.area());
  }
}
```