Inheritance

Object-Oriented Programming

- Encapsulation
  Data & code bound together - > objects

- Polymorphism
  Overloading of constructors & methods

- Inheritance
  A class incorporates another class into its declaration

  A superclass is inherited by a subclass

Inheritance – the basic idea

- A class is defined

- Then a second class is defined that extends the first class. The second class inherits all the variables and methods from the first class and adds its own unique members.

- The first class is called a superclass
- The second class is called a subclass

- It is possible for the superclass to keep some of its members private:
  i.e. label them private

Example

A super class:

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

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

A subclass:

```
class Triangle extends TwoDShape {
    String style;

    double area( ) {
        return width * height / 2;
    }

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

```
class Shapes {
    public static void main(String args[]) {
      Triangle t1 = new Triangle();
      Triangle t2 = new Triangle();

      t1.width = 4.0;
      t1.height = 4.0;
      t1.style = "filled";

      t2.width = 8.0;
      t2.height = 12.0;
      t2.style = "outlined";

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

```
class Shapes {
    public static void main(String args[]) {
      Triangle t1 = new Triangle();
      Triangle t2 = new Triangle();

      t1.width = 4.0;
      t1.height = 4.0;
      t1.style = "filled";

      t2.width = 8.0;
      t2.height = 12.0;
      t2.style = "outlined";

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

Displayed text

```
Info for t1:
Triangle is filled
Width and height are 4 and 4
Area is 8
```

Private members

```
class TwoDShape {
  private double width;  // these are
  private double height; // now private

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

class Triangle extends TwoDShape {
  String style;

  double area() {
    return width * height / 2;        // Error! can't access
  }

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

Use accessor methods to set and get private members.

```
class TwoDShape {
    private double width;      // these are
    private double height;    // now private

  // 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);
  }
}
```

A subclass then uses the accessors

```
class Triangle extends TwoDShape {
    String style;

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

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

The main now uses the accessors also

```
class Shapes2 {
   public static void main(String args[]) {
      Triangle t1 = new Triangle();
      Triangle t2 = new Triangle();

      t1.setWidth(4.0);
      t1.setHeight(4.0);
      t1.style = "filled";

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

A subclass can have a constructor

```
class Triangle extends TwoDShape {
    private String style;

    Triangle(String s, double w, double h) {
        setWidth(w);
        setHeight(h);
        style = s;
    }

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

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

constructor

With constructors in the subclass, the main can specify parameters when creating objects

```
class Shapes3 {
   public static void main(String args[]) {
      Triangle t1 = new Triangle("filled", 4.0, 4.0);
      Triangle t2 = new Triangle("outlined", 8.0, 12.0);

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

   }
}
```

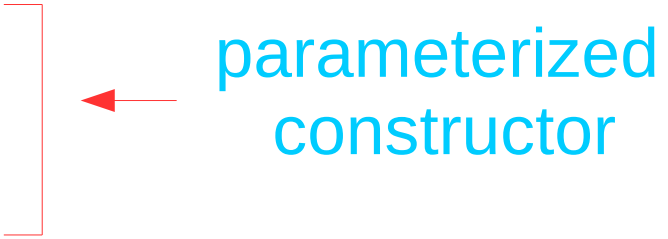
The super class can have a constructor

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

     TwoDShape(double w, double h) {
        width = w;
        height = h;
     }

  // 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; }

    }
}
```



A subclass can call a superclass constructor

```
class Triangle extends TwoDShape {
    private String style;

    Triangle(String s, double w, double h) {
        super(w, h);  <-------  // call superclass constructor
        style = s;
    }

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

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

Comments about a constructor in super

- The super class (TwoDShape) can construct its subobject anyway it wishes, even add functionality that is unknown to the subclass without breaking existing code.

- Any form of constructor defined by the superclass can be called by super( ). The constructor executed will be the one that matches the supplied parameters.