COM343 Object Oriented Programming

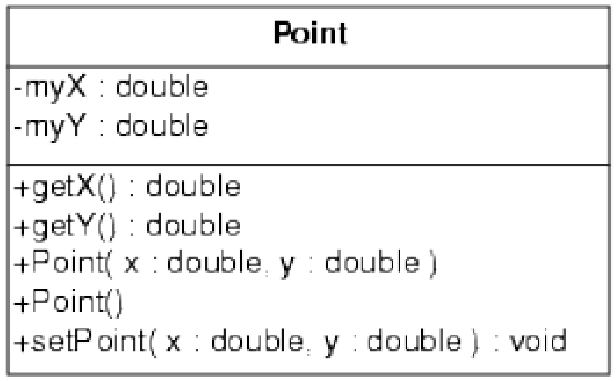
Objects in Java

Defining Classes in Java

- Each Java class is defined in a single file with the exact name of the class, but with a .java file extension
  - You can define more than one classes in a single file but only one class can be public
- **Note:** The class name examples we use actually exist in the library. In a real world application development, you would either use the library classes or define your own classes with different names

Modeling Shapes

- Each shape has an x,y origin (floating point numbers)
- UML to Java (and reverse) mapping is straightforward
  - Good UML Tools can generate basic code from the diagrams
- Point class is simple, yet it hides internal details and provides setters and getters

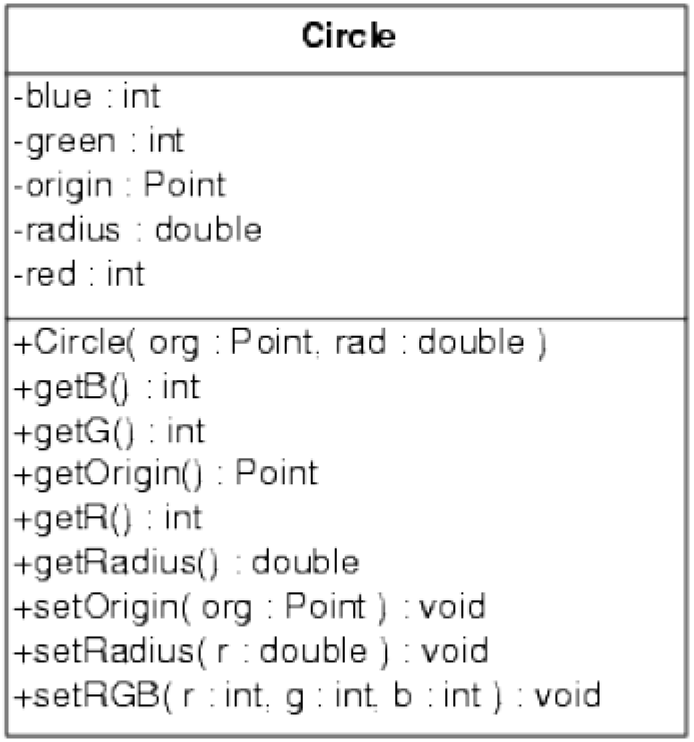

Point
-myX : double
-myY : double
+getX() : double
+getY() : double
+Point(x : double, y : double)
+Point()
+setPoint(x : double, y : double) : void

```
/**
 * Point - a double x,y coordinate
 */
public class Point
{
    // Attributes

    private double myX;
    private double myY;

    // Constructors

    public Point(double x, double y)
    {
        myX = x; myY = y;
    }

    public Point()
    {
        myX = 0.; myY = 0.;
    }

    // Methods

    public double getX()
    {
        return myX;
    }

    public double getY()
    {
        return myY;
    }

    public void setPoint(double x, double y)
    {
        myX = x; myY = y;
    }
}
```

Modeling Shapes

- You may be tempted to provide direct access to attributes
- Even library classes does it
- But libraries are not always good examples of OO design and programming
- It is **not** a good idea to allow direct access to the attributes
  - In fact, even getters and setters can expose internal details that are better kept hidden.

Modeling Shapes



```
-blue : int
-green : int
-origin : Point
-radius : double
-red : int
```

```
+Circle( org : Point, rad : double )
+getB() : int
+getG() : int
+getOrigin() : Point
+getR() : int
+getRadius() : double
+setOrigin( org : Point ) : void
+setRadius( r : double ) : void
+setRGB( r : int, g : int, b : int ) : void
```

Java Naming Conventions

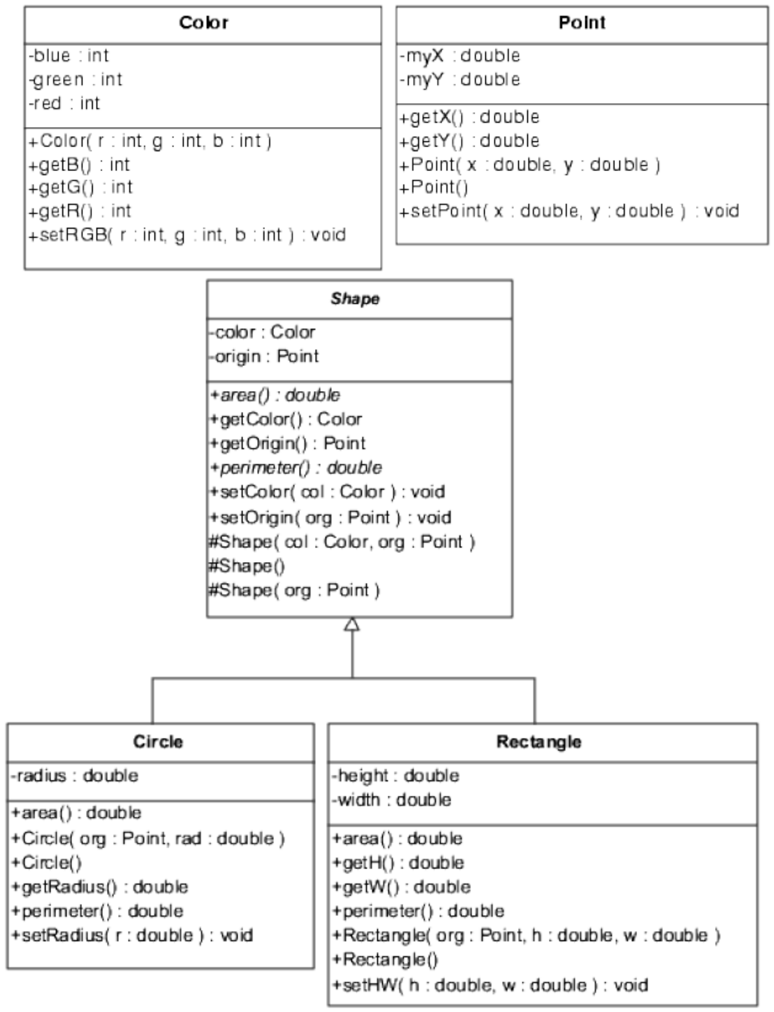
- **Classes**
  - Class (and interface) names should be nouns descriptive of the purpose of the class.
  - Names are in mixed case, beginning with a capital and with the first letter of each internal word capitalized.
  - Use complete words, and avoid abbreviations.
  - Examples: Point, Shape, MovieEditor, ClientList.
- **Methods**
  - Methods should be verbs descriptive of the purpose of the method.
  - Method names are in mixed case, with the first letter lowercase, and the first letter of each internal word capitalized.
  - There are prefix conventions for general types of methods, such as using get and set for getters and setters.
  - Examples: getOrigin, findSmallest, drawGraph, saveModel.

Java Naming Conventions

- **Variables**
  - Except when used as constants, all variables are named using mixed case with a lowercase first letter, and internal words starting with capital letters.
  - Variable names should be meaningful enough to convey their use to someone reading the code.
  - Avoid abbreviations. Use one letter variable names for only for temporary variables.
  - Using meaningful variable names is one of the most important things you can do to make your code easy to read and maintain.
  - Examples: myMovie, editedMovie, backgroundColor, lastItem.
- **Constants**
  - The names of variables used as constants should be all uppercase with words separated by underscores ("_")
  - Examples: MAX_SIZE, R_PG13, TERM_LIMIT.

Inheritance

- In the earlier Point and Circle examples, it would be better to have a Color class and to generalize the concept of a Shape with a Shape class.
- What is common to a Circle and Rectangle?
  - Both will have an origin and color.
  - Both have ways to calculate area and perimeter
- It will not make sense to have an instance of Shape class. Therefore it will be an abstract class (italic name shows this in UML)



Inheritance – Shape Class

```
public abstract class Shape {

   private Color color;
   private Point origin;

   protected Shape (Color col, Point org)
   {
      origin = new Point (org.getX(),
            org.getY());
      color = new Color (col.getR(),
            col.getG(), col.getB());
   }

   protected Shape (Point org) {
       origin = new Point (org.getX(),
            org.getY());
       color = new Color (0, 0, 0);
   }

   protected Shape () {
       origin = new Point (0, 0);
       color = new Color (0, 0, 0);
   }
```

```
   public abstract double area();
   public abstract double perimeter();

   public Color getColor() {
      return color;
   }
   public void setColor(Color col) {
      color.setRGB(col.getR(), col.getG(),
            col.getB());
   }
   public Point getOrigin() {
      return origin;
   }
   public void setOrigin(Point org) {
      origin.setPoint(org.getX(),
            org.getY());
   }
}
```

Inheritance – Circle Class

```java
import java.lang.Math; // for PI

public class Circle extends Shape {

   private double radius;

   public Circle() {
      super();
      radius = 0.0;
   }
   public Circle(final Point org,
         final double rad) {
      super(org);
      radius = rad;
   }
```

```java
   public double area() {
      return Math.PI * radius * radius;
   }

   public double getRadius() {
      return radius;
   }

   public double perimeter() {
      return 2 * Math.PI * radius;
   }

   public void setRadius(double r) {
      radius = r;
   }
}
```

Inheritance – Rectangle Class

```
public class Rectangle extends Shape {

   private double height;
   private double width;

   public Rectangle() {
      super();
      height = 0.0; width = 0.0;
   }

   public Rectangle(Point org, double h,
         double w) {
      super(org);
      height = h; width = w;
   }

   public double area() {
      return height * width;
   }
```

```
   public double perimeter() {
      return 2 * (height + width);
   }

   public double getH() {
      return height;
   }

   public double getW() {
      return width;
   }

   public void setHW(double h, double w)
   {
      height = h; width = w;
   }
}
```