COMSATS UNIVERSITY ISLAMABAD, VEHARI CAMPUS

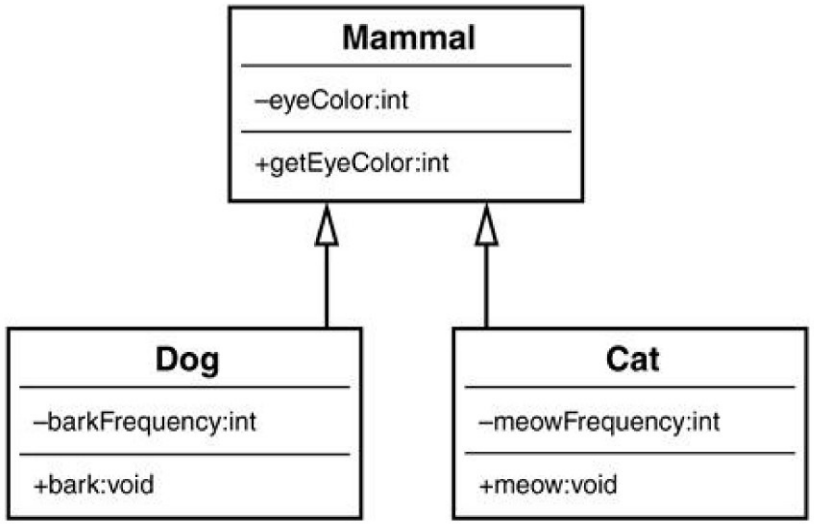
Object Oriented Programming

M. Jawad Rafeeq
Jawadrafeeq@cuivehari.edu.pk

What is Inheritance?

- **Definition**: Inheritance allows you to create a new class based on an existing class.
- **Benefits**:
- Avoids redundancy (repetition of code).
- Makes the code easier to understand and maintain.

- **Example**:
- If you need classes for Dogs, Cats, Mammals, inheritance helps you share common features among them.
- If you need classes for circles, rectangles, and triangles, inheritance helps you share common features among them.



Superclasses and Subclasses

•**Superclass**: A general class from which other classes can inherit. (e.g., GeometricObject)

•**Subclass**: A more specialized class that extends a superclass. (e.g., Circle, Rectangle)

•**Key Points**:
•Subclasses inherit properties and methods from the superclass.
•They can also have their own unique properties and methods.

GeometricObject Class Overview

- **General Class**: GeometricObject
- **Attributes:**
- color: A string representing the color of the shape (e.g., "red", "blue").
- filled: A boolean indicating if the shape is filled (true/false).
- **Methods of GeometricObject**
- getColor(): Returns the color of the shape.
- setColor(color): Sets the color of the shape.
- setFilled(): Return the value of filled.
- getFilled(): To get the value of filled, if the shape is filled.

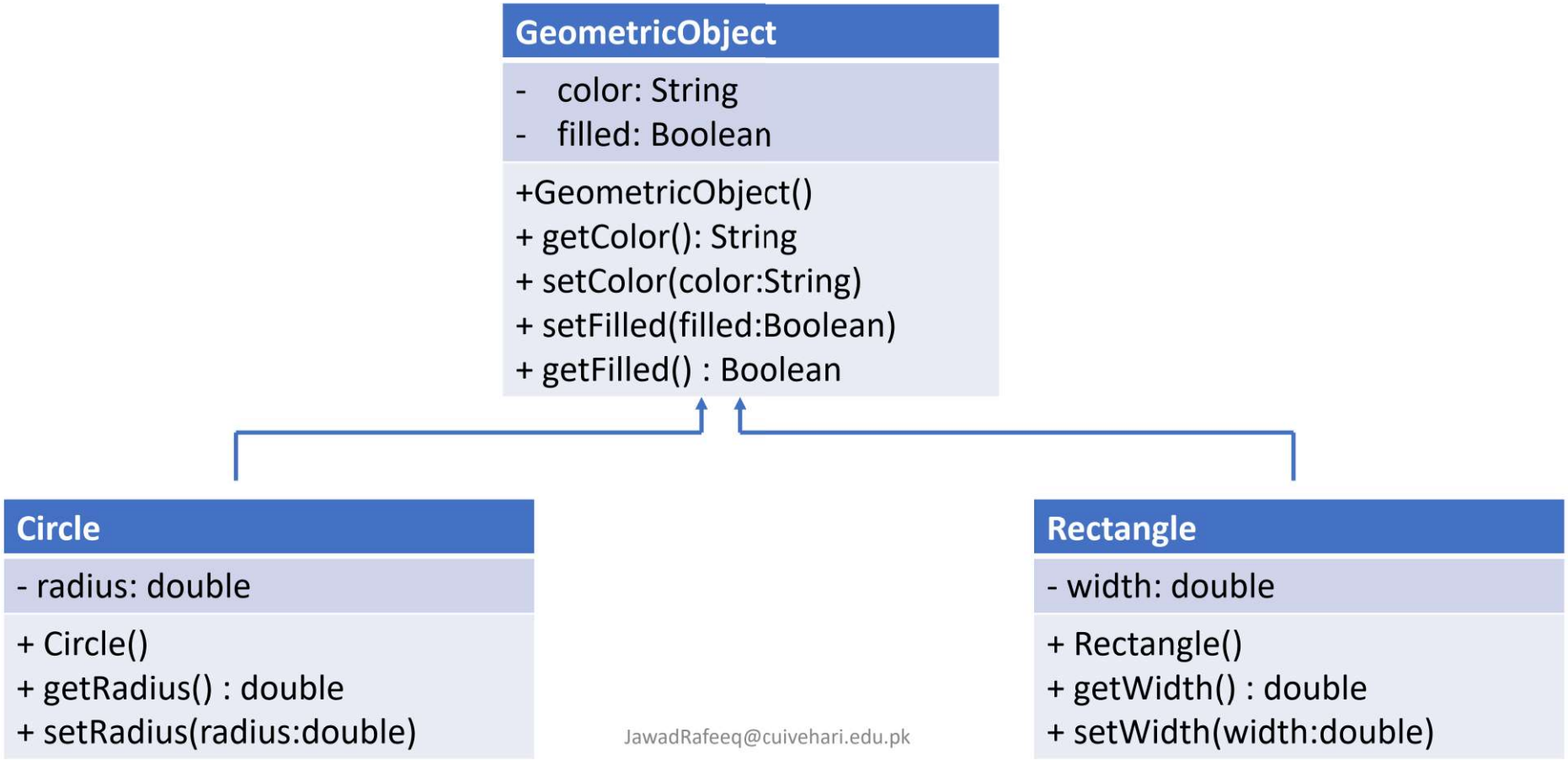
Circle Class

- Circle Class: Inherits properties and methods from GeometricObject.
- New Property:
- radius: A double representing the radius of the circle.
- Methods:
- getRadius(): Returns the radius.
- setRadius(radius): Sets the radius.

Rectangle Class

- Rectangle Class: Inherits properties and methods from GeometricObject.
- New Properties:
- width: A double representing the width of the rectangle.
- Methods:
- getWidth(): Returns the width.
- setWidth(width): Sets the width.

Visual Representation of Inheritance



JawadRafeeq@cuivehari.edu.pk

```
// Superclass
class GeometricObject {
   private String color;
   private boolean filled;

   public GeometricObject() {
      this.color = "white";  // Default color
      this.filled = false;   // Default filled state
   }
  public String getColor() {
      return color;
   }

   public void setColor(String color) {
      this.color = color;
   }
  public boolean getFilled() {
      return filled;
   }

   public void setFilled(boolean filled) {
      this.filled = filled;
   }
}
```

```
// Subclass: Circle inherits from GeometricObject
class Circle extends GeometricObject {
   private double radius;

  public Circle() { this.radius = 1.0;}
  public double getRadius() { return radius; }
   public void setRadius(double radius){ this.radius = radius; }
}
```

```
  // Subclass: Rectangle inherits from GeometricObject
  class Rectangle extends GeometricObject {
     private double width;

   public Rectangle() {
         this.width = 1.0;  // Default width
     }
   public double getWidth() {
         return width;
     }
   public void setWidth(double width) {
         this.width = width;
     }
   }
```

JawadRafeeq@cuivehari.edu.pk

Main class to test the inheritance

```
public class TestInheritance {
    public static void main(String[] args) {
        // Creating a Circle object
        Circle circle = new Circle();
        circle.setColor("red");
        circle.setRadius(5.0);
        System.out.println("Circle Color: " + circle.getColor());
        System.out.println("Circle Radius: " + circle.getRadius());

        // Creating a Rectangle object
        Rectangle rectangle = new Rectangle();
        rectangle.setColor("blue");
        rectangle.setWidth(4.0);
        System.out.println("Rectangle Color: " + rectangle.getColor());
        System.out.println("Rectangle Width: " + rectangle.getWidth());
    }
}
```

Constructors in Inheritance

```
// Superclass
class GeometricObject {
   private String color;
   private boolean filled;

   public GeometricObject() {
      this.color = "white";  // Default color
      this.filled = false;   // Default filled state
   }
  public String getColor() {
      return color;
   }

   public void setColor(String color) {
      this.color = color;
   }
  public boolean getFilled() {
      return filled;
   }

   public void setFilled(boolean filled) {
      this.filled = filled;
   }
}
```

```
// Subclass: Circle inherits from GeometricObject
class Circle extends GeometricObject {
   private double radius;

  public Circle() { this.radius = 1.0;}
 public Circle(double radius, String color, boolean filled) {
  this.radius = radius;
  setColor(color);    // Use superclass method to set color
  setFilled(filled);  // Use superclass method to set filled state
}
  public double getRadius() { return radius; }
    public void setRadius(double radius){ this.radius = radius; }
}

 // Subclass: Rectangle inherits from GeometricObject
 class Rectangle extends GeometricObject {
    private double width;
 public Rectangle() { this.width = 1.0; } // Default width for Rectangle
 public Rectangle(double width, String color, boolean filled) {
    this.width = width;
    setColor(color);    // Use superclass method to set color
    setFilled(filled);  // Use superclass method to set filled state
 }
 public double getWidth() { return width; }
 public void setWidth(double width) {this.width = width;}
 }
```

JawadRafeeq@cuivehari.edu.pk

When you call No-argument constructor

- When you create an object of the Circle class (sub class) using the default constructor or parameterized constructor. The default constructor of GeometricObject (super class) is automatically invoked.

- This sets the default values for color and filled:
- color = "white“
- filled = false

- Default constructor of the superclass is always called unless explicitly specified.

Super keyword is coming: reason

- If no constructor is explicitly called in the subclass (using super()), Java will automatically insert a call to the no-argument (default) constructor of the superclass. This only happens if the superclass has a default constructor (i.e., a constructor that takes no arguments).

- when you want to call a specific parameterized constructor of the superclass, you need to use super() explicitly in the subclass.