COSC 121

Computer Programming II

OOP: Inheritance

Part 1/2

Dr. Mostafa Mohamed

Acknowledgement: The slides mainly rely on the textbook of Y. D. Liang titled “Introduction to Java Programming, 10th Ed.”, Pearson Edu. Inc.

Outline

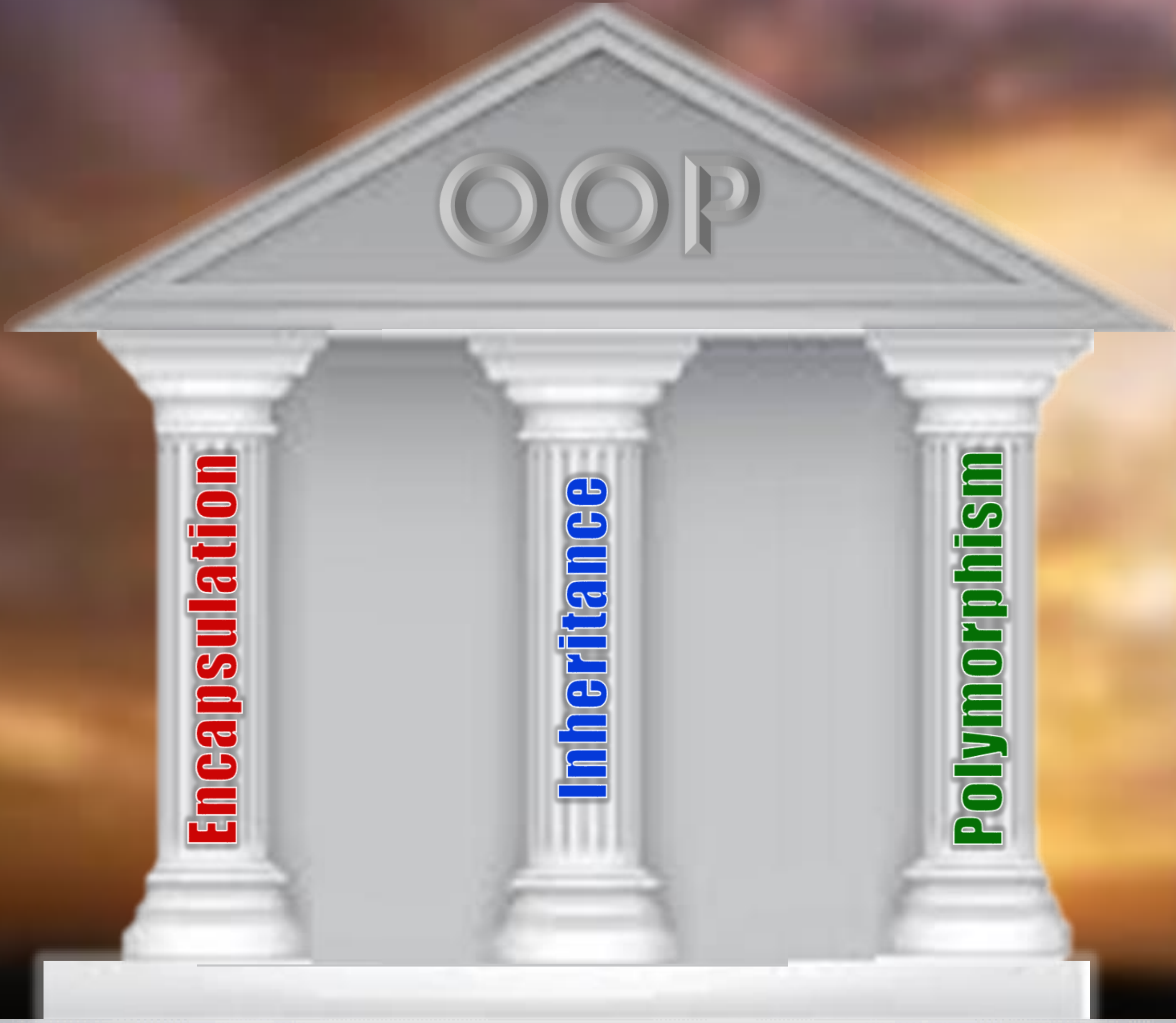
Today:

- Intro to inheritance

Next lecture:

- Method Overriding
- Accessing class members & constructors using `super` keyword
- The `final` modifier
- Visibility Modifiers Revisited
- The `Object` Class and Its Methods

The Three Pillars of OOP



Inheritance

The 2nd pillar

Inheritance Overview

Inheritance is a mechanism for enhancing and extending existing, working classes.

- In real life, you inherit some of the properties from your parents when you are born. However, you also have unique properties specific to you.
- In Java, a class that extends another class inherits some of its properties (methods, instance variables) and can also define properties of its own.

`extends` is the key word used to indicate when one class is related to another by inheritance.

Syntax: `class` *`subclass`* **`extends`** *`superclass`*

- The ***superclass*** is the existing, parent class.
- The ***subclass*** is the new class which contains the functionality of the superclass plus new variables and methods.
- A subclass may only inherit from ***one*** superclass.

Why use inheritance?

The biggest reason for using inheritance is to ***re-use code.***

- Once a class has been created to perform a certain function it can be re-used in other programs.
- Further, using inheritance the class can be extended to tackle new, more complex problems without having to re-implement the part of the class that already works.

The alternative is copy and paste which is bad, especially when the code changes.

Example:

- in the `Circle` and `Rectangle` classes we implemented a few slides ago, there was a lot of code redundancy (e.g. `setColor()` was exactly repeated).
- A better solution is to have a superclass, e.g. `Shape`, that has the common code and then have `Circle` and `Rectangle` inherit from `Shape`.

What is inherited?

When a subclass inherits (or extends) a superclass:

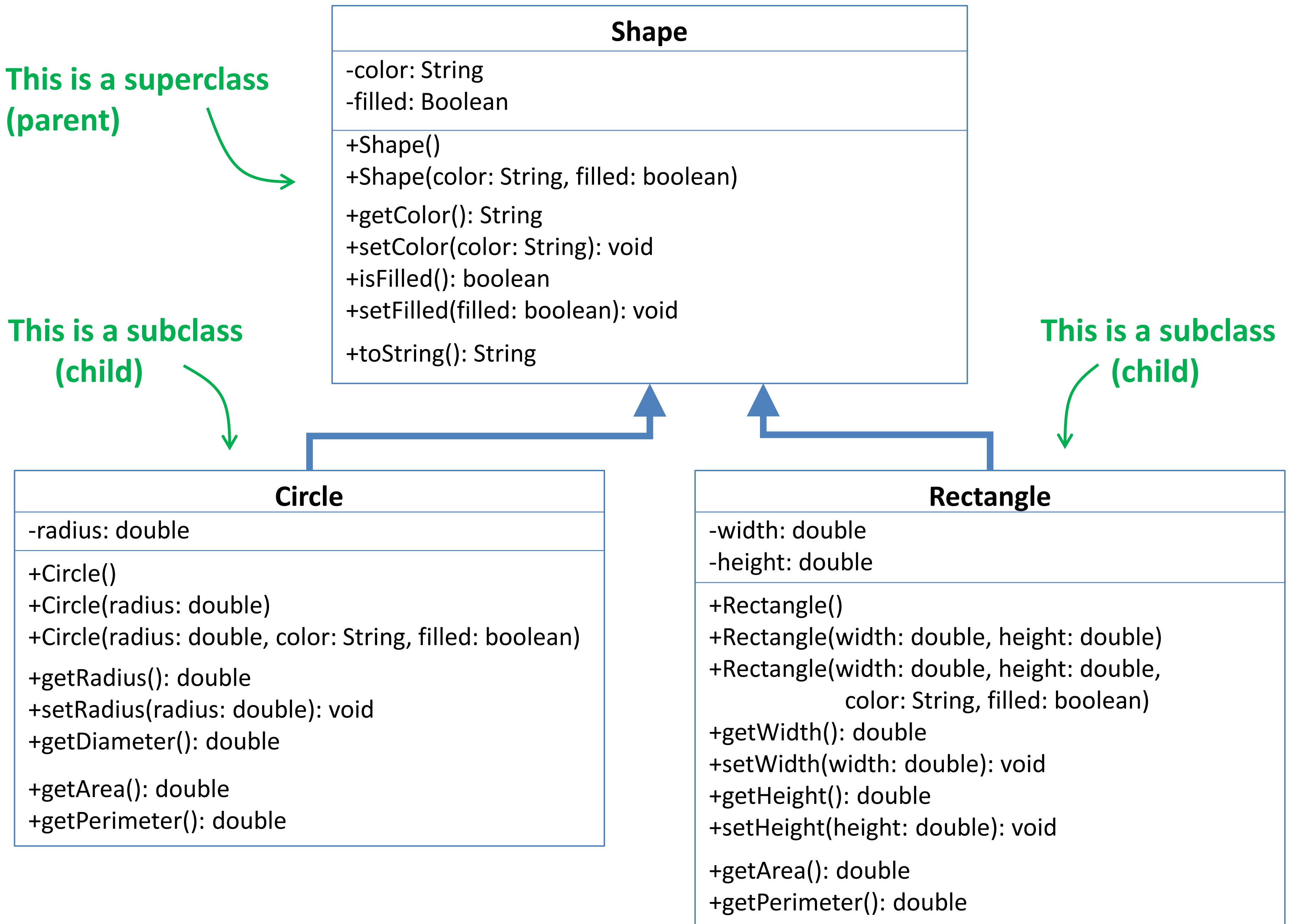
Instance variable inheritance:

- All instance variables of the superclass are inherited by the subclass.
  - However, if a variable is **`private`**, it can only be accessed using methods defined by the superclass.

Method inheritance:

- All non-private superclass methods and attributes are inherited by the subclass, but they may be ***overridden***.

Inheritance Example


This is a superclass (parent)
Shape
-color: String
-filled: Boolean
+Shape()
+Shape(color: String, filled: boolean)
+getColor(): String
+setColor(color: String): void
+isFilled(): boolean
+setFilled(filled: boolean): void
+toString(): String
This is a subclass (child)
This is a subclass (child)
Circle
-radius: double
+Circle()
+Circle(radius: double)
+Circle(radius: double, color: String, filled: boolean)
+getRadius(): double
+setRadius(radius: double): void
+getDiameter(): double
+getArea(): double
+getPerimeter(): double
Rectangle
-width: double
-height: double
+Rectangle()
+Rectangle(width: double, height: double)
+Rectangle(width: double, height: double, color: String, filled: boolean)
+getWidth(): double
+setWidth(width: double): void
+getHeight(): double
+setHeight(height: double): void
+getArea(): double
+getPerimeter(): double

Inheritance Example, cont.

```
public class Shape {
    private String color;
    private boolean filled;

    public Shape() {this("White", true);}
    public Shape(String color, boolean filled) {
        setColor(color);
        setFilled(filled);
    }

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


    public String toString() {
        return "Color: " + color + ". Filled: " + filled;
    }
}
```

Inheritance Example, cont.

```
public class Circle extends Shape {
    private double radius;
    public Circle() { this(1); }
    public Circle(double radius) { setRadius(radius); }
    public void setRadius(double radius) {
        this.radius = radius;
    }
    public double getRadius() { return radius; }
    public double getDiameter() { return 2 * radius; }
    public double getArea() {
        return Math.PI * radius * radius;
    }
    public double getPer() { return 2 * Math.PI * radius; }
    public String toString() {
        return "Color:" + getColor() + ". Filled: " +
        isFilled() + ". Radius: " + radius; }
}
```

From the parent class

Inheritance Example, cont.

```java
public class Rectangle extends Shape{
    private double width, height;
    public Rectangle() {this(1,1);}
    public Rectangle(double width, double height) {
        setWidth(width);
        setHeight(height);
    }
    public double getWidth() {return width;}
    public void setWidth(double width) {this.width = width;}
    public double getHeight() {return height;}
    public void setHeight(double height) {this.height = height;}
    public double getArea() {return width * height;}
    public double getPerimeter() {return 2 * (width+height);}
    public String toString() {
        return "Color:" + getColor() + ". Filled: " + isFilled()
        + ". Width: " + width + ", Height: " + height;
    }
}
```

From the parent class

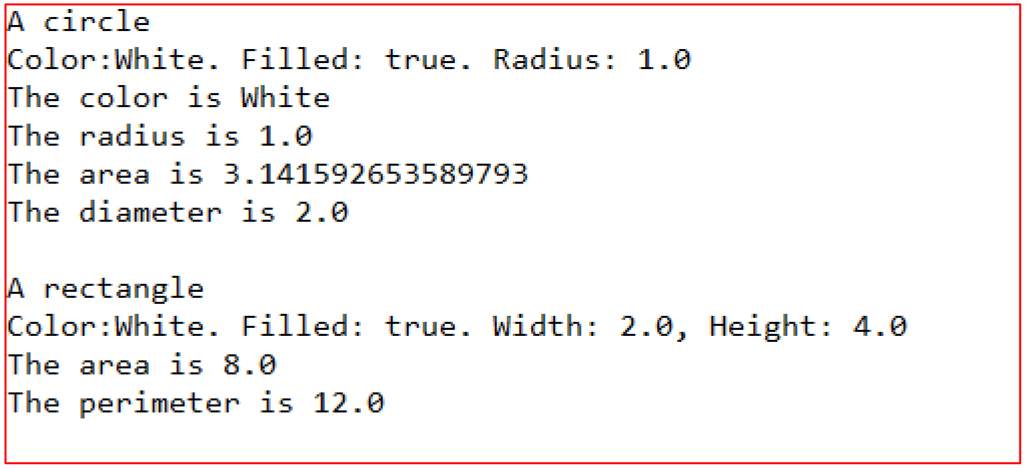
Inheritance Example, cont.

This is a test program!

```java
public class TestShape {
    public static void main(String[] args) {
        Circle circle = new Circle(1);
        System.out.println("A circle\n" + circle.toString());
        System.out.println("The color is " + circle.getColor());
        System.out.println("The radius is " + circle.getRadius());
        System.out.println("The area is " + circle.getArea());
        System.out.println("The diameter is " + circle.getDiameter());


        Rectangle rectangle = new Rectangle(2, 4);
        System.out.println("\nA rectangle\n" + rectangle.toString());
        System.out.println("The area is " + rectangle.getArea());
        System.out.println("The perimeter is " + rectangle.getPerimeter());
    }
}
```

The output



```
A circle
Color:White. Filled: true. Radius: 1.0
The color is White
The radius is 1.0
The area is 3.141592653589793
The diameter is 2.0

A rectangle
Color:White. Filled: true. Width: 2.0, Height: 4.0
The area is 8.0
The perimeter is 12.0
```