IS 2104 - Rapid Application Development

Object Oriented Concepts

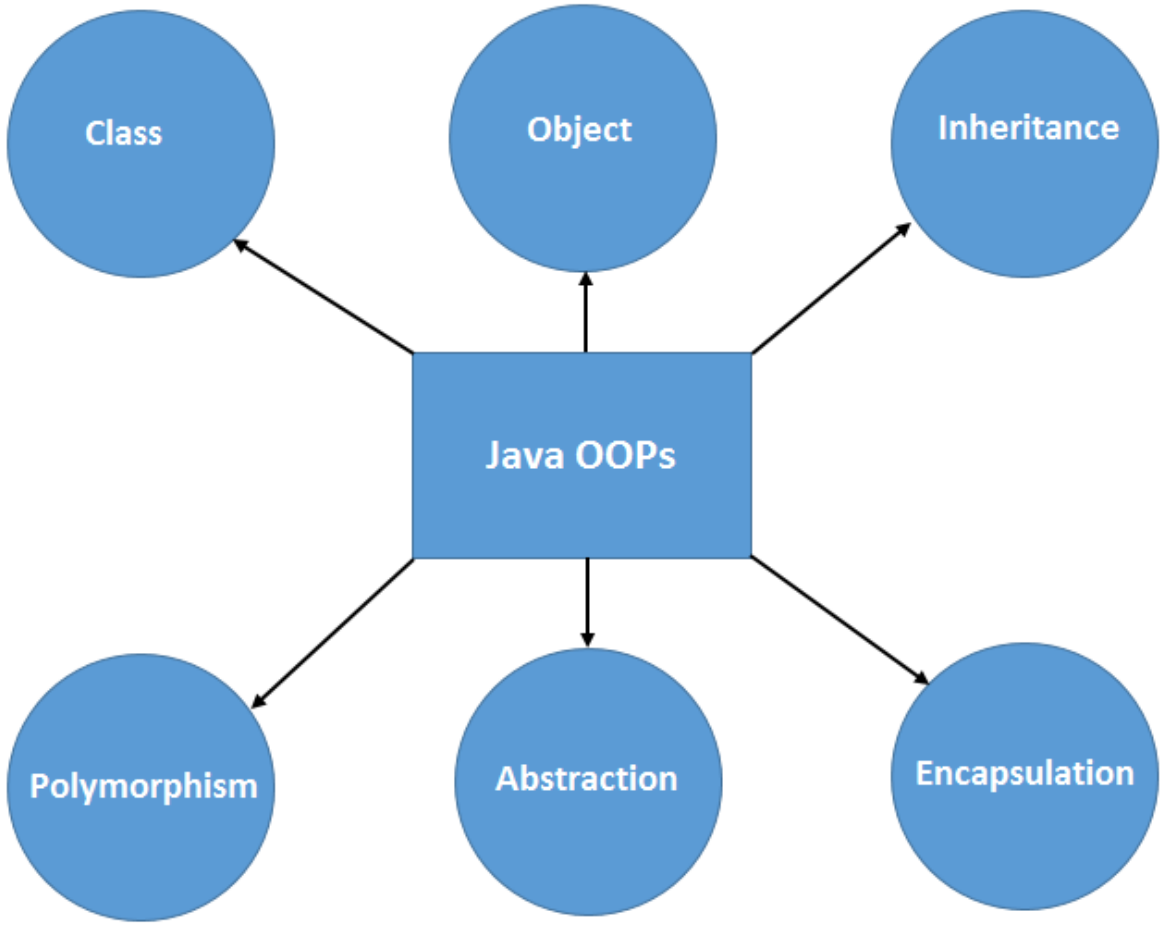
Inheritance

M.V.P. Thilini Lakshika
University of Colombo School of Computing
tlv@ucsc.cmb.ac.lk

Lesson Outline

- What is Inheritance ?
- Superclass and Subclass
- Why Inheritance ?

Object Oriented Concepts



OOPs (Object-Oriented Programming System)

What is Inheritance ?

- Derive new classes from existing classes is known as inheritance.
- Inheritance works in a treelike hierarchical order (Inheritance hierarchy).
- Inheritance creates a parent-child (Is-A) relationship between two classes.
    - Child object acquiring the properties and behaviors of the parent object.
- The key-word used in inheritance is ‘extends’.

Superclass and Subclass

- The existing class is called the superclass, and the new class is the subclass.
- Each subclass can become a superclass for future subclasses.
- The subclass exhibits the behaviors of its superclass and can modify those behaviors so that they operate appropriately for the subclass.
  - Subclass add its own fields and methods.
- Subclass is more specific than its superclass and represents a more specialized group of objects.
- This is why inheritance is referred to as specialization.
- Java supports only single inheritance.
  - Each class is derived from exactly one direct superclass.

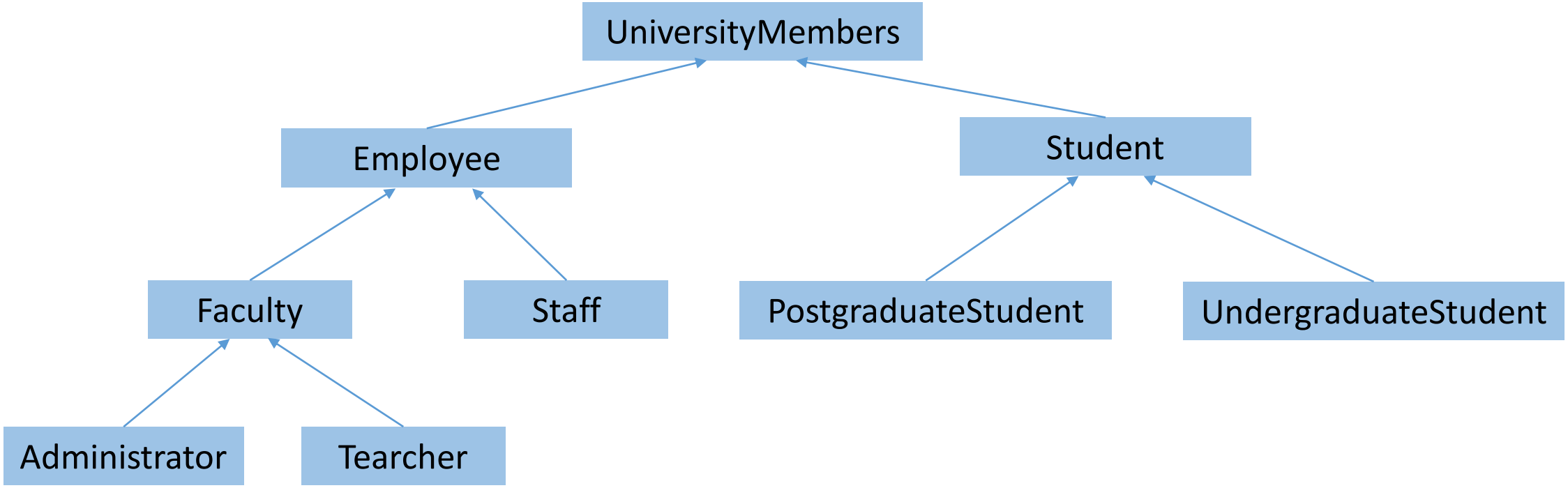
Superclass and Subclass

- In an is-a relationship, an object of a subclass can also be treated as an object of its superclass. Ex: A CarLoan is a Loan.
- superclasses tend to be more general and subclasses more specific.
  - The specialized classes inherit the properties and methods from the general class.
- Every subclass object is an object of its superclass.
- One superclass can have many subclasses.

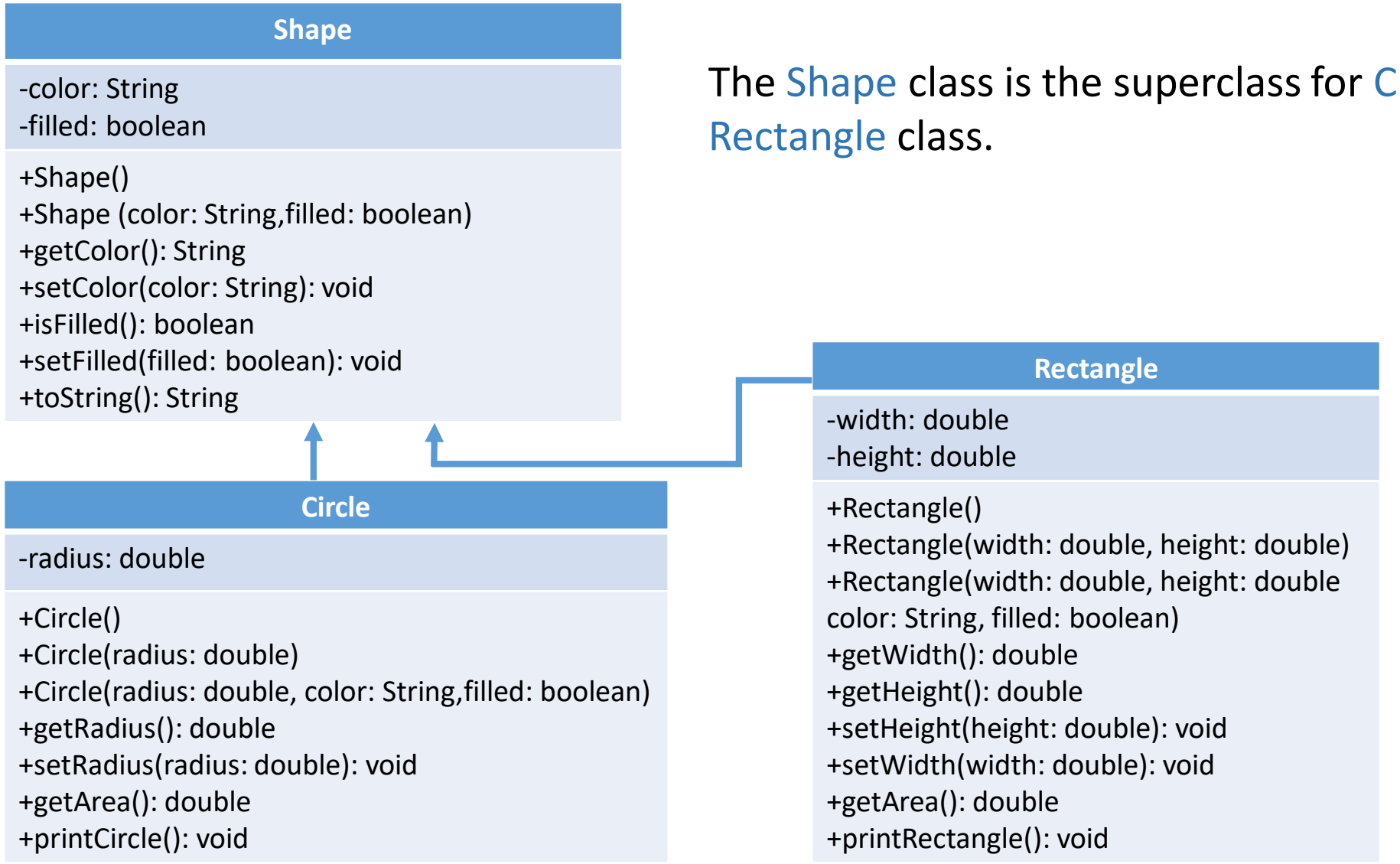
Superclass	Subclasses
Student	PostgraduateStudent, UndergraduateStudent
Shape	Circle, Triangle, Rectangle, Sphere, Cube
Loan	CarLoan, HousingLoan, MortgageLoan
BankAccount	CurrentAccount, SavingsAccount

Superclass and Subclass

Inheritance hierarchy for UniversityMembers.



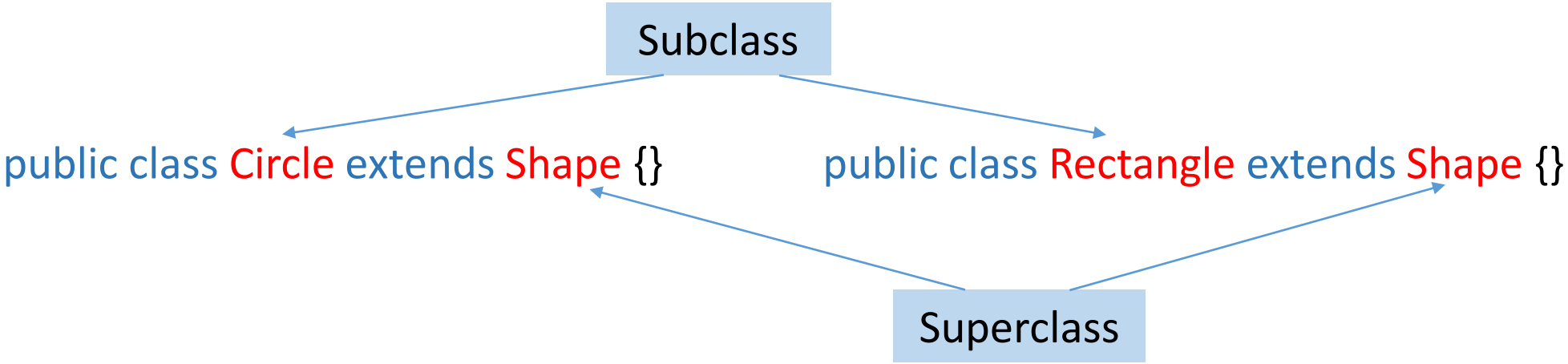
Superclass and Subclass



The Shape class is the superclass for Circle class and Rectangle class.

Superclass and Subclass

- The Circle class and Rectangle class extends the Shape class using the following syntax.



- The keyword extends tells the compiler that the both Circle class and Rectangle class extends the Shape class, thus inheriting the methods getColor, setColor, isFilled, setFilled, and toString.

Inheritance example

```java
package lesson10;

public class Circle extends Shape{
    private double radius;

    public Circle() {
    }

    public Circle(double radius) {
        this.radius = radius;
    }

    public Circle(double radius, String color, boolean filled) {
        this.radius = radius;
        setColor(color);
        setFilled(filled);
    }
    /** Return radius */
    public double getRadius() {
        return radius;
    }

    /** Set a new radius */
    public void setRadius(double radius) {
        this.radius = radius;
```

```java
package lesson10;

public class Shape {
    private String color = "white";
    private boolean filled;
    /** Construct a default shape */
    public Shape() {
    }
    /** Construct a shape with the specified color and filled value **/
    public Shape(String Color, boolean filled) {
        this.color = color;
        this.filled = filled;
    }
    /** Set a new color */
    public void setColor(String color) {
        this.color = color;
    }
    /** Set a new filled */
    public void setFilled(boolean filled) {
        this.filled = filled;
    }
    /** Return color */
    public String getColor() {
        return color;
    }
    /** Return filled. Since filled is boolean,its get method is named i
    public boolean isFilled() {
```

Reference to the Java codes of Shape, Circle and Rectangle classes are available in the UGVLE.

Inheritance example

- The keyword this is used to reference the calling object.
- You might attempt to use the data fields color and filled directly in the constructor as follows:

```
public Circle(double radius, String color, boolean filled) {
        this.radius = radius;
        this.color = color; // Illegal
        this.filled = filled; // Illegal
}
```

- This is illegal, because the private data fields color and filled in the Shape class cannot be accessed in any class other than in the Shape class itself.
- The only way to read and modify color and filled is through their get and set methods.

Inheritance example

- The keyword super refers to the superclass of the class in which super appears.
- Unlike properties and methods, the constructors of a superclass are not inherited in the subclass.
- They can only be invoked from the constructors of the subclasses, using the keyword super.
- super keyword can be used in two ways.
  - To call a superclass constructor
  - To call a superclass method

Inheritance example

To call a superclass constructor

- super() invokes the no-argument constructor of its superclass.
- super(arguments) invokes the superclass constructor that matches the arguments.
- The statement super() or super(arguments) must appear in the first line of the subclass constructor.

Example for calling a super class constructor.

```
public Circle (double radius, String color, boolean filled) {
    super(color, filled);
    this.radius = radius;
}
```

Inheritance example

To call a superclass method

- The syntax is super.method(parameters);
- You could rewrite the printCircle() method in the Circle class as follows:

```
public void printCircle() {
        System.out.println("Radius of the circle is " + radius + ", area of the circle is"
        +getArea()+ ", color is" + super.getColor());
}
```

- It is not necessary to put super before getColor(), because getColor is a method in the Shape class and is inherited by the Circle class.
- But in some cases, the keyword super is needed.

Why Inheritance ?

- An important and powerful feature in Java for reusing software.
  - A new class is created by absorbing an existing class's members and enhancing them with new or modified capabilities.
- Save time during program development by creating new classes on existing proven and debugged high-quality software.
- Increases the likelihood that a system will be implemented and maintained effectively.

Summary

- A subclass usually contains more information and methods than its superclass, it is not a subset of its superclass.
- Private data fields in a superclass are not accessible outside the class.
- Not all *is-a* relationships should be modeled using inheritance.

  Ex: A square is a rectangle, but you should not define a Square class to extend a Rectangle class. For class A to extend class B, A should contain more detailed information than B.
- Do not blindly extend a class for the sake of reusing methods.

  Ex: No sense for a Tree class to extend a Person class, even though they share common properties such as height and weight. A subclass and its superclass must have the *is-a* relationship.

END