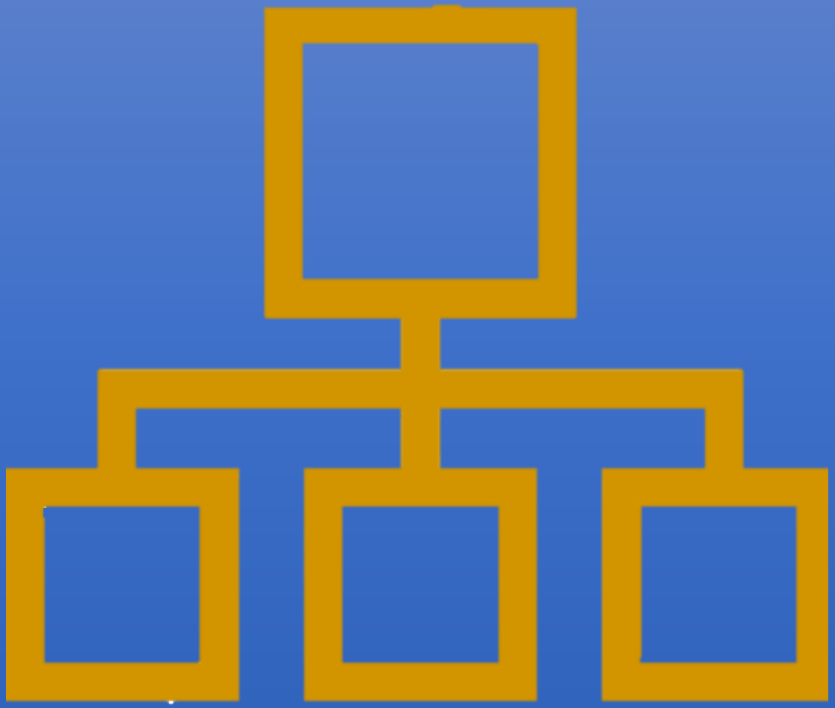


Lab 5
Class Inheritance

KEYNOTE

Inheritance

- **Definition:**
  - A class (called *subclass, derived class*, *extended class*, or *child class*) that is derived from another class (called *superclass, base class* or a *parent class*).

```
public class ClassName extends SuperClass {
    ...
}
```

- **What You Can Do in a Subclass**
  - A subclass inherits all of the *public* and *protected* members of its parent. (NOT vice versa)
  - You can declare a field in the subclass with the same name as the one in the superclass, thus *hiding* it (**Not recommended**)
  - The inherited methods can be used directly as they are.
  - You can write a new instance method in the subclass that has the same signature as the one in the superclass, thus overriding it.
  - You can write a subclass constructor that invokes the superclass`s constructor, either implicitly or by using the keyword *super*.

```
  super(parameter);   //call parent class 's constructor
  super.parentMethodName(parameter);
  super.parentFieldName;
```

PRACTICE

Exercise

- Objective:
  - Use inheritance to create hierarchies of related classes
  - Extend behavior and override existing behavior
- To Do:
  - Create the hierarchies with classes: Circle, Rectangle, Square, TwoDimensionalShape.
  - Implement constructor for all class.
  - Implement method to return the String of the current class and direct super class.
  - Implement method to return the area of object.

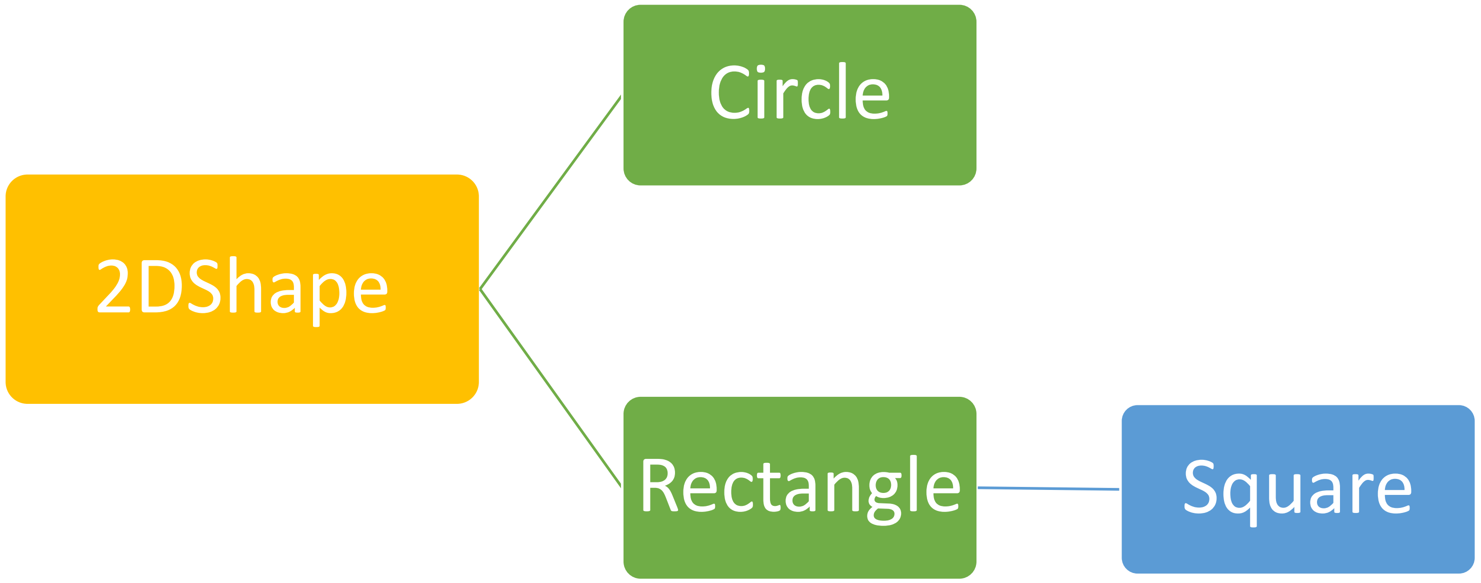
Exercise

- Create objects and print the result like this:

```
Four shapes have been created:
1. CirObject is a [Circle], and is a [2D Shape]
CirObject's area is 28.27, radius is 3.00
2. RecObject  is a [Rectangle], and is a [2D Shape]
RecObject's area is 12.00, width is 3.00, height is is 4.00
3. SquareObject1 is a [Square], and is a [Rectangle]
SquareObject 's area is 36.00, side is 6.00

Is SquareObject1 a TwoDimensionalShape? true
Is SquareObject1 a Rectangle? true
Is SquareObject1 a Square? true
```

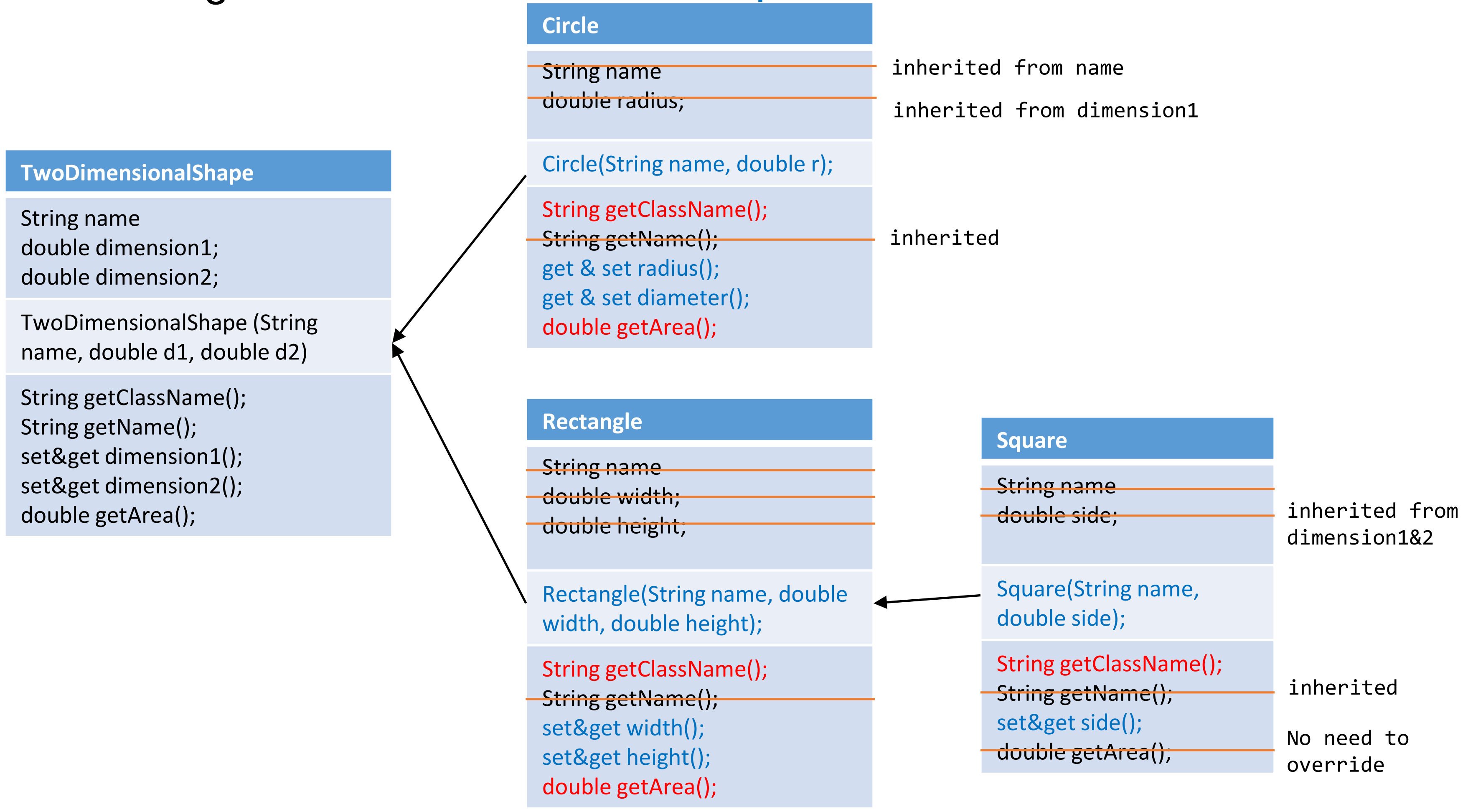
Class hierarchy

- Design superclasses to store common characteristics
- Design the subclasses to store specialized characteristics



Class hierarchy

- Design superclasses to store common characteristics
- Design the subclasses to store specialized characteristics



TwoDimensionalShape

String name
double dimension1;
double dimension2;

TwoDimensionalShape (String name, double d1, double d2)

String getClassName();
String getName();
set&get dimension1();
set&get dimension2();
double getArea();

Circle

~~String name~~ inherited from name
~~double radius;~~ inherited from dimension1

Circle(String name, double r);

String getClassName();
~~String getName();~~ inherited
get & set radius();
get & set diameter();
double getArea();

Rectangle

~~String name~~
~~double width;~~
~~double height;~~

Rectangle(String name, double width, double height);

String getClassName();
~~String getName();~~
set&get width();
set&get height();
double getArea();

Square

~~String name~~
~~double side;~~ inherited from dimension1&2

Square(String name, double side);

String getClassName();
~~String getName();~~ inherited
set&get side();
~~double getArea();~~ No need to override

TwoDimensionalShape Class

Field(s) and Method(s)	Description

String name
double dimension1;
double dimension2;	

TwoDimensionalShape (String name, double d1, double d2)	*initialize all class fields*
String getClassName() String getName() Get & Set dimension1() Get & Set dimension2(); double getArea();	*return the "2D Shape“* *return the name of the class* *return 0, will be overrided by child class*

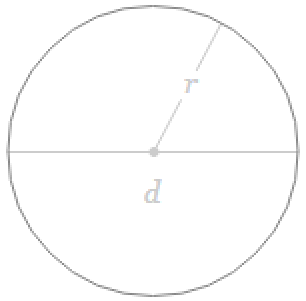
TwoDimensionalShape Class

```java
/**
 * Create class of 2Dshape and can be
subclassed.
 */
public class TwoDimensionalShape {
  private String name;
  private double dimension1;
  private double dimension2;

  // constructor
  public TwoDimensionalShape(String name,
double d1, double d2) {
      this.name = name;
      this.dimension1 = d1;
      this.dimension2 = d2;
  }

  // get name and class name
  public String getClassName() {
      return "2D Shape";
  }

  public String getName() {
      return name;
  }

  // get & set methods for dimension 1
  public double getDimension1() {}
  public void setDimension1(double d) {}

  // get & set methods for dimension 2
  public double getDimension2() {}
  public void setDimension2(double d) {}

  // don't know the kind of current shape
  // so return 0 only
  // must be implement in subclass
  // OR public abstract double getArea();
  public double getArea() {
      return 0;
  }
}
```

Circle Class

Field(s) and Method(s)	Description
Circle(String name, double radius);	*initialize all class fields* *The name is inherited from name of super class* *The radius is inherited from dimension1 of super class*

String getClassName();
get & set radius();
get & set diameter();
double getArea();
	return "Circle"
	Call the super method to update the dimension1
	Calculation and call the super method to update the dimension1
	return the area of the circle



$$A = \pi r^2$$

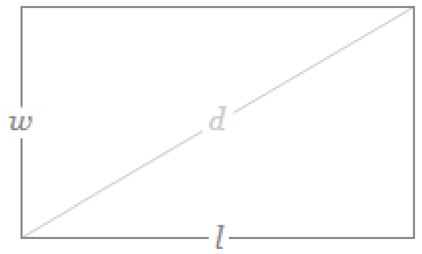
Circle Class

```
public class Circle extends TwoDimensionalShape {
  public Circle(String name, double r) {
      super(name, r, r);
  }

  @Override
  // return current class name
  public String getClassName() {
      return "Circle";
  }

  public double getRadius() {
      return super.getDimension1();
  }

  public void setRadius(double r) {
      super.setDimension1(r);
      super.setDimension2(r);
  }

  @Override
  public double getArea() {
      return Math.PI * super.getDimension1() * super.getDimension1();
  }

  @Override
  public String toString() {
      return String.format("%s is a [%s], and is a [%s]",
          super.getName(), getClassName(), super.getClassName());
  }
}
```

Rectangle Class

Field(s) and Method(s)	Description
Rectangle (String name, double width, double height);	*initialize all class fields* *The name is inherited from name of super class* *The width is inherited from dimension1 of super class* *The height is inherited from dimension2 of super class*

String getClassName();
double getWidth();
double getHeight()
setSize(double w, double h);
double getArea();
	return "Rectangle"
	Call the super method to get the dimension1
	Call the super method to get the dimension2
	Call the super method to set the dimension1&dimension2
	return the area of the Rectangle



$$A = w\,l$$

Rectangle Class

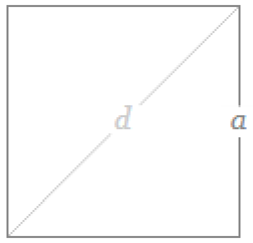
- Similar with Circle class

```
public class Rectangle extends TwoDimensionalShape {
  public Rectangle(String name, double width, double height) {
     // store width in field demension1, height in field demension2
     super(name, width, height);
  }

  public String getClassName(){
     return "Rectangle";
  }

  public double getWidth() {
     return super.getDimension1();
  }

  public double getHeight() {
     return super.getDimension2();
  }

  public void setSize(double w, double h){
     super.setDimension1(w);
     super.setDimension2(h);
  }

  public double getArea() {
     return super.getDimension1() * super.getDimension2();
  }

  public String toString() {
     return String.format("%s is a [%s], and is a [%s]",
        super.getName(), getClassName(), super.getClassName());
  }
}
```

Square Class

Square	Description
Square(String name, double side);	*initialize all class fields* *The name is inherited from name of (indirect) super class* *The side is inherited from dimension1 and dimension2*

String getClassName();
Get & Set side();
	return “Square”
	Call the super method to get & set both the dimension1 and the dimension2



$$A = a^2$$

Square Class

- Inheritance from Rectangle Class

```
public class Square extends Rectangle {
  public Square(String name, double side) {
     // this will call the constructor of Rectangle
     super(name, side, side);
  }

  public String getClassName(){
     return "Square";
  }

  public double getSide() {
     return getWidth();
  }

  public void setSide(int side) {
     super.setSize(side, side);
  }

  public String toString() {
     return String.format("%s is a [%s], and is a [%s]",
        super.getName(), getClassName(), super.getClassName()); }
  }
}
```

Test Program

- Create these objects:
  - Circle cir1 = new Circle("Cir One", 3.0);
  - Rectangle rec1 = new Rectangle("Rect One", 3.0, 4.0);
  - Square sq1 = new Square("Square One", 6.0);
- Print the fields of the class

Circle:	area, radius, diameter
Rectangle:	area, width, height
Square :	area, side

Test Program

```java
public class ShapeTester {
  public static void main(String[] args) {

       // create an object
       Circle cir1 = new Circle("Cir One", 3.0);
       Rectangle rec1 = new Rectangle("Rec One", 3.0, 4.0);
       Square sq1 = new Square("Square One", 6.0);

       TwoDimensionalShape sq2 = new Square("Square Two", 4.0);
       System.out.println("Four shapes have been created:");

       // print the object properties
       System.out.println("1." + cir1);
       System.out.printf( "%s's area is %.2f, radius is %.2f\n",
           cir1.getName(),cir1.getArea(), cir1.getRadius());

       System.out.println("2." + rec1);
       System.out.printf( "%s's area is %.2f, width is %.2f, height is %.2f\n",
           rec1.getName(),rec1.getArea(), rec1.getWidth(), rec1.getHeight());

       System.out.println("3." + sq1);
       System.out.printf( "%s's area is %.2f, side is %.2f\n",
           sq1.getName(), sq1.getArea(), sq1.getSide());

       // print all circle shape
       System.out.printf("Is %s a TwoDimensionalShape? %s\n",sq1.getName(), sq1 instanceof
TwoDimensionalShape);
       System.out.printf("Is %s a Rectangle? %s\n",sq1.getName(), sq1 instanceof Rectangle);
       System.out.printf("Is %s a Square? %s\n", sq1.getName(), sq1 instanceof Square);
  }
}
```

QUIZ

Quiz

1. What is the base class of all classes

java.lang.Object

2. Does Java support Multiple Inheritances?

No

3. What is the method overriding? Can we override a static method?

Child class has the signature same as parent class, it overrides the method of parent class

static method는 런타임에 실행X 컴파일타임에 실행
->실제 객체를 찾는 작업을 시행하지 않기 때문

No

4. What is method overloading

A method with the same name but different number, sequence or types of arguments

5. Can we create 2 methods with the same name, same parameter but different return type?

No.