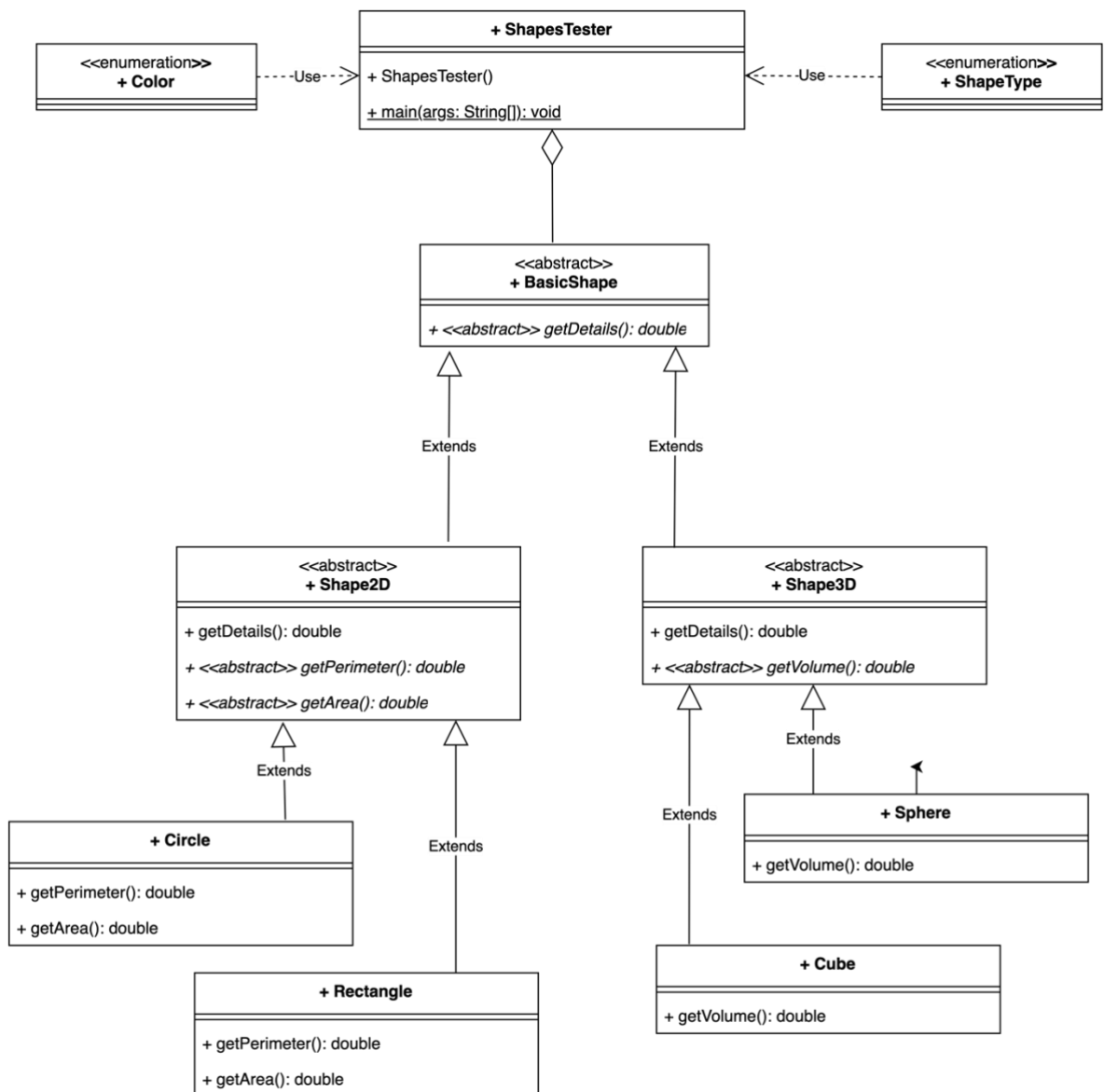


# CMM024 Object Oriented Programming

## Practical Session: 9

This lab is dedicated to Object Oriented Programming; focusing on inheritance with a strong focus on Abstract classes and Enums. You are going to create your own classes and test them

### What you are going to model (and more)



## Brief:

This lab continues the demonstration done during the lecture i.e., the Shape modelling.

- Download the lecture source code.
- Create a folder Lab9Coding somewhere
- Copy and paste the Color, Shape, Circle, Rectangle and ShapesTester java files found downloaded from Moodle into it.
- Open Visual Code, create a new window and open the folder you just created
- Run the project.
- You are ready to proceed with the lab!

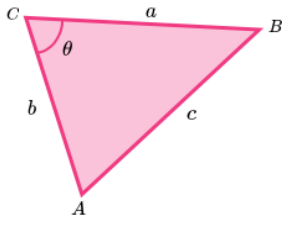
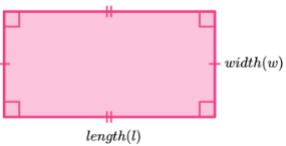
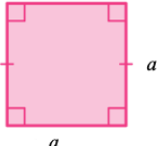
## Inspiration: Code demonstrated during the lecture


```
public Color getColor() {
    return this.color;
}
public int getXPos() {
    return this.xPos;
}
public int getYPos() {
    return this.yPos;
}
public abstract double getPerimeter();
public abstract double getArea();
public String toString() {
    String st = "";
    st += "\nColor: " + this.getColor();
    st += "\nX position: " + this.getXPos();
    st += "\nY position: " + this.getYPos();
    st += "\nPerimeter: " + this.getPerimeter();
    st += "\nArea: " + this.getArea();
    return st;
}
```

Color: BLUE  
X position: 10  
Y position: 10  
Perimeter: 172.787595947438  
Area: 2375.8294442772813

Color: BLUE  
X position: 20  
Y position: 20  
Perimeter: 140.0  
Area: 1000.0  
jean-claude@MacBook-Pro-2 S

## Formulas for 2D shapes

2D Shape		Perimeter	Area
Triangle		$P = a + b + c$	$A = \frac{1}{2} * ab * \sin(\theta)$
Rectangle		$P = (width * 2) + (length * 2)$	$A = l * w$
Square		$P = a * 4$	$A = a^2$

Circle		$P = 2 * \pi * r$	$A = \pi * r^2$
--------	---	-------------------	-----------------

## Coding missing shape classes: enum ShapeType

- Code an enum called ShapeType as shown below

<<enumeration>> + ShapeType
TRIANGLE
RECTANGLE
SQUARE
CIRCLE

ShapeType.java > ...

```
1 public enum ShapeType {
2     TRIANGLE, RECTANGLE, SQUARE, CIRCLE
3 }
```

- Add this enum as a field to the BasicShape class and to the constructor parameter so you can initialise it.

<<abstract>> + BasicShape
- color: Color - shapeType: ShapeType
+ BasicShape(shapeType: ShapeType, color: Color, int xPos, int yPos)
+ getShapeType(): ShapeType
+ getColor(): Color
+ <<abstract>> getDetails(): double

```
J BasicShape.java > ...
1 public abstract class BasicShape {
2     private Color color;
3     private ShapeType shapeType;
4
5     public BasicShape(ShapeType shapeType, Color color) {
6         this.color = color;
7         this.shapeType = shapeType;
8     }
9     public ShapeType getShapeType(){
10         return this.shapeType;
11     }
12     public Color getColor() {
13         return this.color;
14     }
15
16     public abstract String getDetails();
17
18     public String toString(){
19         return getDetails();
20     }
21 }
22
```

## Coding Shape2D

Write a class in the same manner as Shape2D.java.

Don't forget to initialise the superclass using the super keyword

<b>&lt;&lt;abstract&gt;&gt; + Shape2D</b>
- xPos: double - yPos: double
+ Shape2D(shapeType: ShapeType, color: Color, int xPos, int yPos) + getXPos(): double + getYPos(): double + <<abstract>> getPerimeter(): double + <<abstract>> getArea(): double + getDetails(): String

Shape2D.java > ...

```
1 public abstract class Shape2D extends BasicShape{
2
3     private int xPos;
4     private int yPos;
5     public Shape2D(ShapeType shapeType, Color color, int xPos, int yPos) {
6         super(shapeType, color);
7         this.xPos = xPos;
8         this.yPos = yPos;
9     }
10    public int getXPos() {
11        return this.xPos;
12    }
13    public int getYPos() {
14        return this.yPos;
15    }
16    public abstract double getPerimeter();
17    public abstract double getArea();
18
19    public String getDetails() {
20        String st = "";
21        st += "\nColor: " + this.getColor();
22        st += "\nShape Type: " + this.getShapeType();
23        st += "\nX position: " + this.getXPos();
24        st += "\nY position: " + this.getYPos();
25        st += "\nPerimeter: " + String.format(format:"%.2f", this.getPerimeter());
26        st += "\nArea: " + String.format(format:"%.2f", this.getArea());
27        return st;
28    }
29 }
```

## Coding Shape3D

Write a class in the same manner as Shape3D.java

Don't forget to initialise the superclass using the super keyword

<b>&lt;&lt;abstract&gt;&gt; + Shape3D</b>
- xPos: double - yPos: double - zPos: double
+ Shape3D(shapeType: ShapeType, color: Color, int xPos, int yPos, zPos: double) + getXPos(): double + getYPos(): double + getZPos(): double + <<abstract>> <i>getVolume(): double</i> + getDetails(): String

Shape3D.java > Shape3D > getVolume()

```
1 public abstract class Shape3D extends BasicShape {
2
3     private int xPos;
4     private int yPos;
5     private int zPos;
6     public Shape3D(ShapeType shapeType, Color color, int xPos, int yPos, int zPos) {
7         super(shapeType, color);
8         this.xPos = xPos;
9         this.yPos = yPos;
0         this.zPos = zPos;
1     }
2     public int getXPos() {
3         return this.xPos;
4     }
5     public int getYPos() {
6         return this.yPos;
7     }
8     public int getZPos() {
9         return this.zPos;
0     }
1     public abstract double getVolume();
2     public String getDetails() {
3         String st = "";
4         st += "\nColor: " + this.getColor();
5         st += "\nShape Type: " + this.getShapeType();
6         st += "\nX position: " + this.getXPos();
7         st += "\nY position: " + this.getYPos();
8         st += "\nZ position: " + this.getZPos();
9         st += "\nVolume: " + String.format(format:"%.2f", this.getVolume());
0         return st;
1     }
2 }
```

## Coding shape: finishing Rectangle

Amend the Rectangle class downloaded from Moodle to add the ShapeType

+ Rectangle
- width: double - length: double
+ Rectangle(shapeType: ShapeType, color: Color, int xPos, int yPos, width: double, length: double) + getWidth(): double + getLength(): double + getPerimeter(): double + getArea(): double

Rectangle.java > ...

```
1 public class Rectangle extends Shape2D{
2     private double width;
3     private double length;
4     public Rectangle(ShapeType shapeType, Color color, int xPos, int yPos, double width, double length){
5         super(shapeType, color, xPos, yPos);
6         this.width = width;
7         this.length = length;
8     }
9     public double getWidth(){
10        return this.width;
11    }
12    public double getLength(){
13        return this.length;
14    }
15    @Override
16    public double getPerimeter() {
17        //P= 2 * π * r
18        return (2 * this.width) + (2 * this.length);
19    }
20    @Override
21    public double getArea() {
22        //A = π * r ^ 2
23        return width * length;
24    }
25 }
```

## Coding shape: finishing Circle

Amend the Circle class downloaded from Moodle to add the ShapeType

+ Circle
- radius: double
+ Circle(shapeType: ShapeType, color: Color, int xPos, int yPos, radius: double)
+ getRadius(): double
+ getPerimeter(): double
+ getArea(): double

## Coding shape: Square

A Square is a rectangle with equal Width and Length. Thus, Square can extend Rectangle, and simply use the inherited functionality!

Don't forget to initialise the Rectangle superclass!

+ Square
+ Square(shapeType: ShapeType, color: Color, int xPos, int yPos, width: double, length: double)

Square.java > ...

```

1 public class Square extends Rectangle{
2     public Square(ShapeType shapeType, Color color, int xPos, int yPos, double width, double length){
3         super(shapeType, color, xPos, yPos, width, length);
4     }
5 }

```

## Coding shape: Triangle

Here we start with 2 lengths of a triangle i.e., sideA and sideB, and we are given the angle between them. To calculate the area, this is enough. However, when it comes to calculate the perimeter, we need to find the length of the third side. This is done in the getSide3() method. Once you have created the class, code the getSide3, getArea and getPerimeter methods.

+ Triangle
- sideA: double - sideB: double - angle: double
+ Triangle(shapeType: ShapeType, color: Color, int xPos, int yPos, sideA: double, sideB: double, angle: double) + getSideA(): double + getSideB(): double + getAngle(): double + getSide3(): double + getPerimeter(): double + getArea(): double

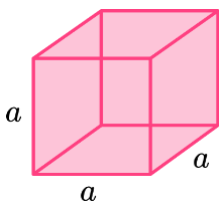
$$\text{side3} = \sqrt{\text{sideA}^2 + \text{sideB}^2 - (2 * \text{sideA} * \text{sideB} * \cos(\text{angle}))}$$

```

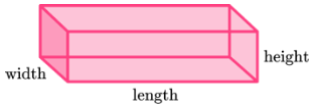
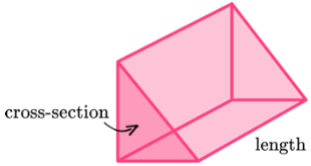
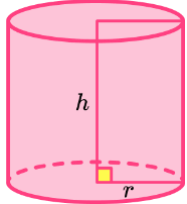
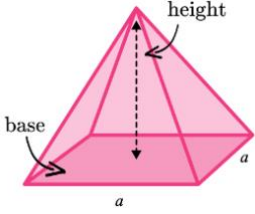
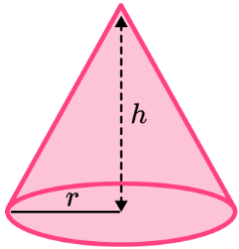
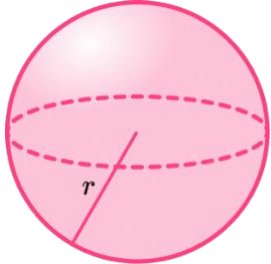
public double getSide3() {
    // c^2 = a^2 + b^2 - (2abcos(angle))
    double side3 = (sideA * sideA) + (sideB * sideB) - (2 * sideA * sideB * Math.cos(angle));
    return side3;
}

```

## Formulas for 3D shapes

3D shape		Volume
Cube		$V = a^3$
		V = area of square * side



Cuboid		$V = length * width * height$
		$V = \text{area of rectangle} * height$
Prism		$V = \left(\frac{1}{2} * ab * \sin(\theta)\right) * length$
		$V = \text{area of triangle} * length$
Cylinder		$V = \pi * r^2 * h$
		$V = \text{area of circle} * height$
Pyramid		$V = \left(\frac{1}{3}\right) * a^2 * height$
		$1/3 \text{ area of square} * height$
Cone		$V = \left(\frac{1}{3}\right) * \pi r^2 * h$
		(Volume=1/3 ×Area of circle × Height)
Sphere		$V = \left(\frac{4}{3}\right) * \pi r^3$
		(Volume=4/3 ×Area of circle × radius)

## Coding shape: Sphere

The Sphere is a 3D shape and therefore will extend the Shape3D superclass

+ Sphere
- radius: double
+ Sphere(shapeType: ShapeType, color: Color, int xPos, int yPos, zPos: double, radius: double) + getRadius(): double + getVolume(): double

Sphere.java > ...

```
1 public class Sphere extends Shape3D{
2
3     private double radius;
4
5     public Sphere(ShapeType shapeType, Color color, int xPos, int yPos, int zPos, double radius){
6         super(shapeType, color, xPos, yPos, zPos);
7         this.radius = radius;
8     }
9
10    public double getRadius(){
11        return this.radius;
12    }
13
14    @Override
15    public double getVolume() {
16        return (4.0/3.0) * Math.PI * Math.pow(radius, 3);
17    }
18 }
19 }
```

## Coding shape: Cuboid

The Cuboid is a 3D shape and therefore will extend the Shape3D superclass

+ Cuboid
- width: double - length: double - height: double
+ Rectangle(shapeType: ShapeType, color: Color, int xPos, int yPos, width: double, length: double, height: double) + getWidth(): double + getLength(): double + getHeight(): double + getVolume(): double

+ Cuboid
- width: double - length: double - height: double
+ Cuboid(shapeType: ShapeType, color: Color, int xPos, int yPos, width: double, length: double, height: double) + getWidth(): double + getLength(): double + getHeight(): double + getVolume(): double

## Coding shape: Cube

The Cube is a 3D shape and therefore will extend the Shape3D superclass

+ Cube
- side: double
+ Cube(shapeType: ShapeType, color: Color, int xPos, int yPos, width: double, length: double, height: double) + getSide(): double + getVolume(): double

## Testing: creating the ShapesTester class

+ ShapesTester
+ ShapesTester() + main(args: String[]): void

apesTester.java > ...

```
import java.util.ArrayList;
```

```
public class ShapesTester {
```

Run | Debug

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

```
    BasicShape s1 = new Circle(ShapeType.CIRCLE, Color.BLUE, xPos:10, yPos:10, radius:27.5);
```

```
    BasicShape s2 = new Rectangle(ShapeType.RECTANGLE, Color.BLUE, xPos:20, yPos:20, width:20, length:50);
```

```
    BasicShape s3 = new Sphere(ShapeType.SPHERE, Color.BLUE, xPos:30, yPos:30, zPos:30, radius:50);
```

```
    ArrayList<BasicShape> shapes = new ArrayList<>();
```

```
    shapes.add(s1);
```

```
    shapes.add(s2);
```

```
    shapes.add(s3);
```

```
    for (BasicShape shape : shapes) {
```

```
        System.out.println(shape);
```

```
    }
```

```
}
```

```
}
```

/Lab9\_2023\_66cb3afb/bin ShapesTester

Color: BLUE  
Shape Type: CIRCLE  
X position: 10  
Y position: 10  
Perimeter: 172.79  
Area: 2375.83

Color: BLUE  
Shape Type: RECTANGLE  
X position: 20  
Y position: 20  
Perimeter: 140.00  
Area: 1000.00

Color: BLUE  
Shape Type: SPHERE  
X position: 30  
Y position: 30  
Y position: 30  
Volume: 523598.78

jean-claude@PC49672 Lab9\_2023 %

## **Important: Create all remaining 3D shapes to practice!**

We only have done a Sphere, Cuboid and Cube

Create all remaining 3D shapes

In the test class, create these shapes and run it