# 10-Practice. Using abstraction in Java

Lecturer: T.Kuchkorov

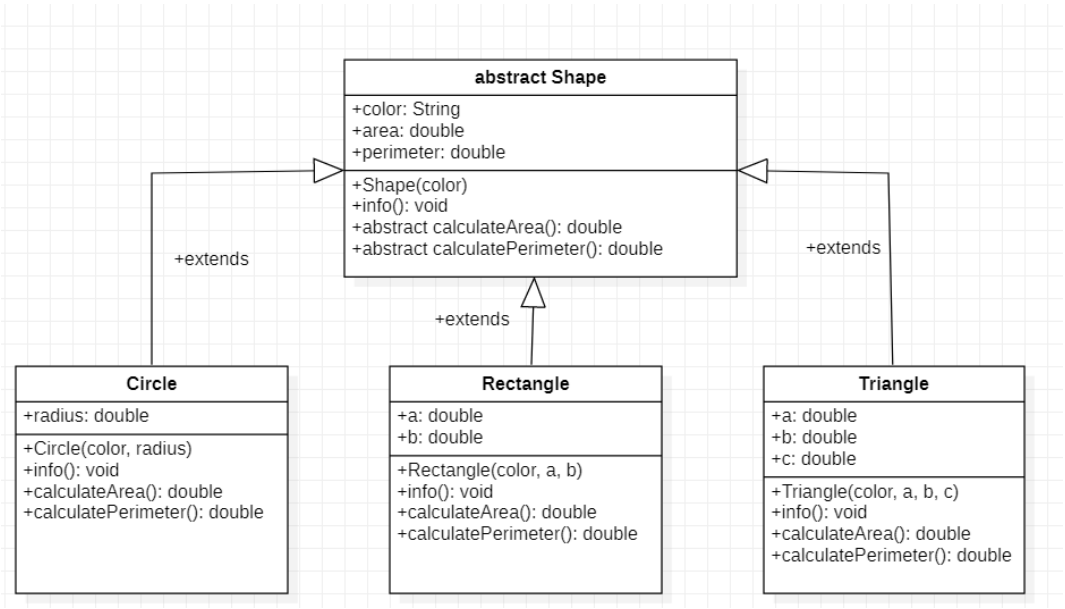
OOP | V1: 22.11.2021
V2: 28.04.2022

# Using abstraction principles between classes.

In this practical work, students will learn the features of programming based on the principle of abstraction, overriding abstract methods of abstract classes in the inherited class in the Java programming language. Also students will acquire the ability to create interfaces and implement the interface into a class, using interface final and static attributes and overriding interface methods.

## Classwork example.

Create classes according to the following hierarchy (UML Class diagram). Here the Shape class is an abstract class. There are 2 abstract methods (calculateArea () and calculatePerimeter () methods within the Shape class.




## Shape.java

```
public abstract class Shape {
    protected String color;
    protected double area;
    protected double perimeter;

    public Shape(String color) { this.color = color; }

    protected void info(){
        System.out.println("Unknown shape");
        System.out.println("Area: " + this.area);
        System.out.println("Perimeter: " + this.perimeter);
        System.out.println("----------------------------------");
    }
    protected abstract double calculateArea();
    protected abstract double calculatePerimeter();
}
```

Circle.java

```
public class Circle extends Shape {
    private double radius;

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

    public void info(){
        System.out.println("***Shape is Circle***");
        System.out.println("Color: " + this.color);
        System.out.println("Radius: " + this.radius);
        System.out.println("-------------------------------");
    }

    @Override
    public double calculateArea(){
        this.area = Math.PI*Math.pow(radius,2);
        System.out.println("Area (Circle): " + this.area);
        return this.area;
    }

    @Override
    public double calculatePerimeter(){
        this.perimeter = 2*Math.PI*this.radius;
        System.out.println("Perimeter (Circle): " +
this.perimeter);
        return this.perimeter;
    }
}
```

**Rectangle.java**

```
public class Rectangle extends Shape{
    private double a;
    private double b;

    public Rectangle(String color, double a, double b){
        super(color);
        this.a = a;
        this.b = b;
    }

    public void info(){
        System.out.println("***Shape is Rectangle***");
        System.out.println("Color: " + this.color);
        System.out.println("a (width): " + this.a);
        System.out.println("b (height): " + this.b);
        System.out.println("--------------------------------");
    }

    @Override
    public double calculateArea(){
        this.area = this.a * this.b;
        System.out.println("Area (Rectangle): " + this.area);
        return this.area;
    }

    @Override
    public double calculatePerimeter(){
        this.perimeter = 2*(this.a + this.b);
        System.out.println("Perimeter (Rectangle): " +
this.perimeter);
        return this.perimeter;
    }
}
```

Triangle.java

```
public class Triangle extends Shape {
    private double a;
    private double b;
    private double c;
    private boolean access;

    public Triangle(String color, double a, double b, double
c){
        super(color);
        this.a = a;
        this.b = b;
        this.c = c;
        if((this.a + this.b> this.c) &&
                (this.a + this.c> this.b) &&
                (this.b + this.c> this.a)
        ){
            this.access = true;
        }else{
            this.access = false;
        }
    }

    public void info(){
        System.out.println("***Shape is Triangle***");
        System.out.println("Color: " + this.color);
        System.out.println("a (side-1): " + this.a);
        System.out.println("b (side-2): " + this.b);
        System.out.println("c (side-3): " + this.c);
        System.out.println("-----------------------------");
    }

    @Override
    public double calculateArea(){
        if(this.access){
            double p;
            p = (this.a + this.b + this.c)/2;
            this.area = Math.sqrt(p*(p-this.a) * (p-this.b) *
(p-this.c));
            System.out.println("Area (Triangle): " +
this.area);
            return this.area;
        }else{
            System.out.println("Wrong sides for triangle to
calculate area");
            return 0.0;
        }
```

```
    }

    @Override
    public double calculatePerimeter(){
        if(this.access){
            this.perimeter = this.a + this.b + this.c;
            System.out.println("Perimeter (Triangle): " +
this.perimeter);
            return this.area;
        }else{
            System.out.println("Wrong sides for triangle to
calculate perimeter");
            return 0.0;
        }
    }
}
```

Main.java

```
public class Main {

    public static void main(String[] args) {
        //Can not create object from Shape
        //Shape sh1 = new Shape("Black");
        Shape sh2 = new Circle( color: "Blue",  radius: 2.5);
        Shape sh3 = new Rectangle( color: "Red",  a: 4, b: 5);
        Shape sh4 = new Triangle( color: "Green",  a: 3, b: 4, c: 5);
        sh2.info();
        sh2.calculateArea();
        sh2.calculatePerimeter();
    }
}
```

Result:

```
"C:\Program Files\Java\jdk-11.0.12\bin\java.exe"
***Shape is Circle***
Color: Blue
Radius: 2.5
-----------------------------------
Area (Circle): 19.634954084936208
Perimeter (Circle): 15.707963267948966
```

## Tasks for Abstraction in Java:

Abstraction (using abstract classes and override abstract methods)

**Task#1. List of classes (abstract and derived classes):**

◦ **Post** – super abstract class:
Attributes: postName: String
postDate: LocalDate
Methods:
create(ArrayList<Post> posts): abstract void
update(): abstract void
show(): abstract void

◦ **KunUzPost** – derived class of Post:
Attributes: author: String
category: String
postText: String
Methods:
@Override methods

◦ **DaryoPost** – derived class of Post:
Attributes: author: String
category: String
postText: String
Methods:
@Override methods

◦ **QalampirPost** – derived class of Post:
Attributes: author: String
category: String
postText: String
Methods:
@Override methods

To get result:

- Input post information using Scanner or Custom Randomizer classes
- Add different posts to the Arraylist<Post> in Main class.
- Add posts to the list using create() method.
- Update post information
- Show all posts

Abstraction (using interfaces and abstract classes, override abstract methods)

**Task#2. Classes and interfaces:**

- Transaction – interface
    - start(), end(), verify(), proceed(), cancel() – abstract methods
- Bank – super class (optional: abstract class)
- AloqaBank – derived class, implement Transaction
- IpakYuliBank – derived class, implement Transaction
- OFBank – derived class, implement Transaction

**Tip: Use your own attributes and methods for classes**

**Task#3. Classes and Interface:**

- VehiclesOperations – Interface
    - startEngine(), stopEngine(), lightOn(), lightOff – abstract methods
- Vehicle – super class (optional: abstract class)
- Car – derived class, implement VehiclesOperations
- Bus – derived class, implement VehiclesOperations
- Trunck – derived class, implement VehiclesOperations

**Tip: Use your own attributes and methods for classes**