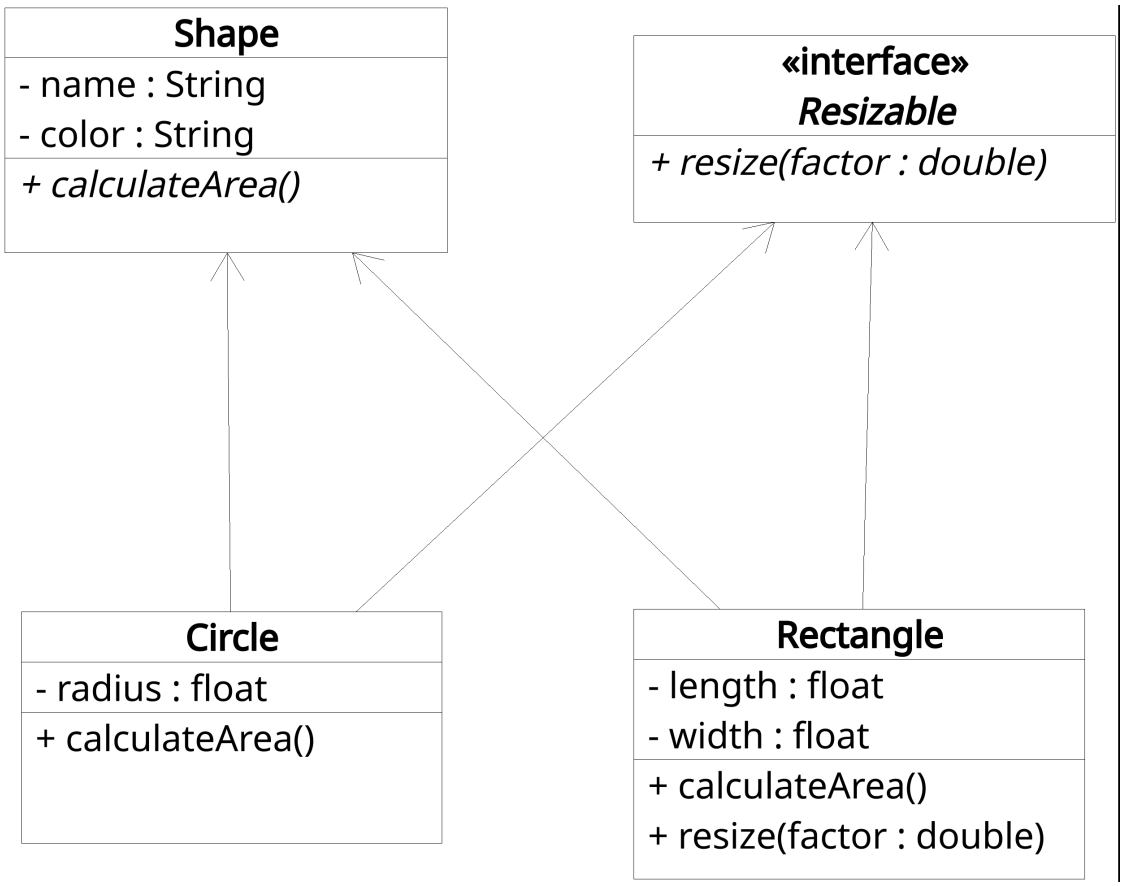
Lab 4: Abstract Class and Interface

Objective:

- Understanding abstract methods and classes.
- Declare and implement Interface.

Programs:

1. Program to demonstrate the use of abstract class and interface to represent Shapes



- /shapes/Shape.java

```
package shapes;

public abstract class Shape {
    private String name;

    public String getName() {
        return name;
    }

    private String color;

    public String getColor() {
        return color;
    }

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

```
    abstract float area();

    public void displayShapeInfo() {
        System.out.println("Shape: " + name);
        System.out.println("Color: " + color);
        System.out.println("Area: " + area());
    }

}
```

- shapes/Circle.java

```
package shapes;

public class Circle extends Shape {
    private float radius;

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

    @Override
    public float area() {
        return (float) Math.PI * radius * radius;
    }

}
```

- shapes/Rectangle.java

```
package shapes;

public class Rectangle extends Shape {
    float length;
    float width;

    public Rectangle(String name, String color, float length, float width) {
        super(name, color);
        this.length = length;
        this.width = width;
    }

    @Override
    public float area() {
        return length * width;
    }


    public float getLength() {
        return length;
    }
    public float getWidth() {
        return width;
    }

}
```

- shapes/ResizeRectangle.java

```
package shapes;

interface resizeable {
    void resize(double factor);
}
public class ResizeRectangle extends Rectangle implements resizeable {
    public ResizeRectangle(String name, String color, float length, float width) {
        super(name, color,length,width);
        this.length = length;
        this.width = width;
    }

    @Override
    public void resize(double factor) {
        this.length *= factor;
        this.width *= factor;
    }

}
```

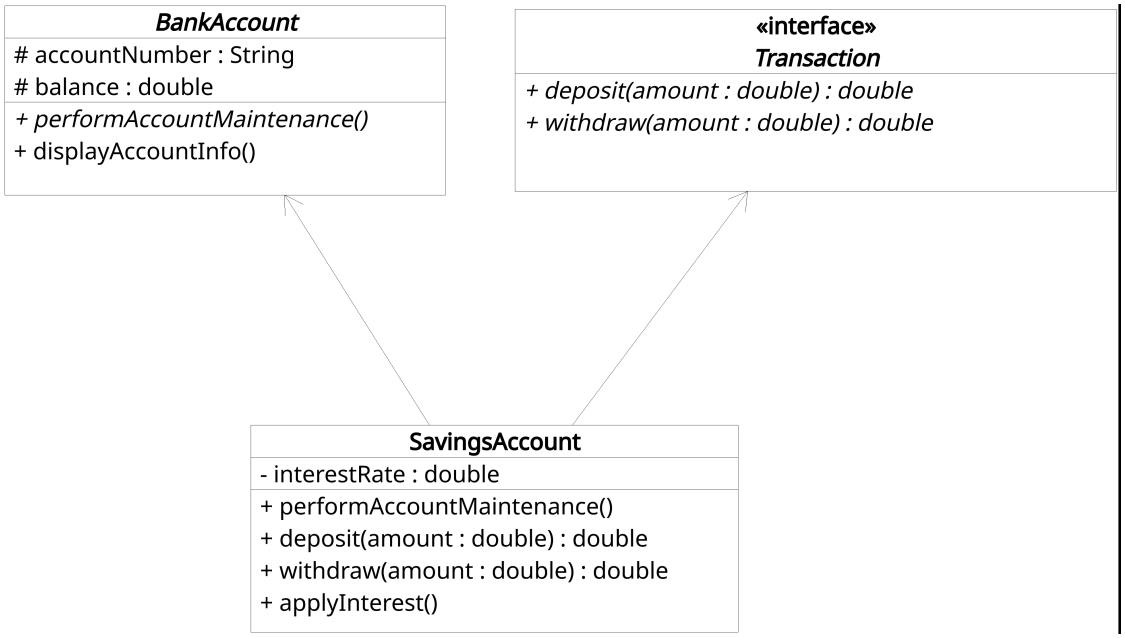
- ShapeTest.java

```
import shapes.Circle;
import shapes.ResizeRectangle;

public class ShapeTest {
    public static void main(String[] args) {
        Circle circle = new Circle("Circle", "Red", 5.0f);
        ResizeRectangle rectangle = new ResizeRectangle("Rectangle", "Blue", 4.0f, 6
        // Display information
        circle.displayShapeInfo();
        System.out.println("--------------------");
        rectangle.displayShapeInfo();
        // Resize rectangle and display updated information
        rectangle.resize(1.5);
        System.out.println("--------------------");
        rectangle.displayShapeInfo();

    }
}
```

Output:

```
Shape: Circle
Color: Red
Area: 78.53982
--------------------
Shape: Rectangle
Color: Blue
Area: 24.0
--------------------
Shape: Rectangle
Color: Blue
Area: 54.0
```

Program to demonstrate the use of abstract class and interface to represent Shapes




```java
abstract class BankAccount {
    protected String accountNumber;
    protected double balance;

    public BankAccount(String accountNumber, double balance) {
        this.accountNumber = accountNumber;
        this.balance = balance;
    }

    void displayAccountInfo() {
        System.out.println("Account Number: " + accountNumber);
        System.out.println("Balance: " + balance);
    }

    abstract void performAccountMaintenance();
}

interface Transaction {
    double deposit(double amount);

    double withdraw(double amount);
}

class SavingsAccount extends BankAccount implements Transaction {
    double interestRate;

    public SavingsAccount(String accountNumber, double balance, double interestRate) {
        super(accountNumber, balance);
        this.interestRate = interestRate;
    }

    @Override
    public double deposit(double amount) {
        balance += amount;
        return balance;
    }
```

```
    @Override
    public double withdraw(double amount) {
        if(balance >= amount) {
            balance -= amount;
        }
        return balance;
    }

    @Override
    void performAccountMaintenance() {
        System.out.println("Performing Savings Account Maintenance");
    }

    void applyInterest() {
        balance += balance * interestRate;
    }

}



public class BankingSystem {
    public static void main(String[] args) {
        // Create a SavingsAccount object
        SavingsAccount savingsAccount = new SavingsAccount("123456",1000.0,0.05);

        // Display initial account information
        System.out.println("Initial Account Information:");
        savingsAccount.displayAccountInfo();
        printSeparator();

        // Deposit some amount
        double depositedAmount = 500.0;
        System.out.println("Depositing $" + depositedAmount);
        savingsAccount.deposit(depositedAmount);
        savingsAccount.displayAccountInfo();
        printSeparator();

        // Withdraw some amount
        double withdrawnAmount = 200.0;
        System.out.println("Withdrawing $" + withdrawnAmount);
        savingsAccount.withdraw(withdrawnAmount);
        savingsAccount.displayAccountInfo();
        printSeparator();

        // Perform account maintenance
        System.out.println("Performing Account Maintenance:");
        savingsAccount.performAccountMaintenance();
        printSeparator();

        // Apply interest
        System.out.println("Applying Interest:");
        savingsAccount.applyInterest();
        savingsAccount.displayAccountInfo();
```

```
        }

        private static void printSeparator() {
            System.out.println("--------------------------------");
        }
    }
```

Output:

Conclusion:

- We learned about abstract classes and methods.
- We learned about interfaces.