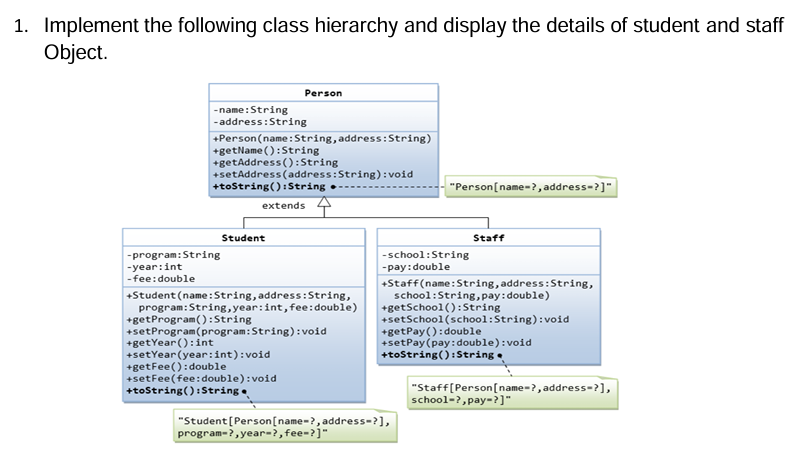
****

**package** oops.practice;

**class** Person {

**private** String name;

**private** String address;

Person(String name, String address) {

**this**.name = name;

**this**.address = address;

}

**void** setAddress(String address) {

**this**.address = address;

}

**void** setName(String name) {

**this**.name = name;

}

String getAddress() {

**return** address;

}

String getName() {

**return** name;

}

}

**class** Student **extends** Person {

**private** String program;

**private** **int** year;

**private** **double** fee;

Student(String name, String address, String program, **int** year, **double** fee) {

**super**(name, address);

**this**.program = program;

**this**.year = year;

**this**.fee = fee;

}

**void** setProgram(String program) {

**this**.program = program;

}

**void** setYear(**int** year) {

**this**.year = year;

}

**void** setFee(**double** fee) {

**this**.fee = fee;

}

**int** getYear() {

**return** year;

}

**double** getFee() {

**return** fee;

}

**public** String toString() {

**return** "Student[Person[name="+getName()+",address="+ getAddress()+"],program="+program+", year="+year+",fee="+fee + "]";

}

}

**class** Staff **extends** Person {

**private** String school;

**private** **double** pay;

Staff(String name, String address, String school, **double** pay) {

**super**(name, address);

**this**.school = school;

**this**.pay = pay;

}

**void** setPay(**double** pay) {

**this**.pay = pay;

}

**void** setSchool(String school) {

**this**.school = school;

}

**double** getPay() {

**return** pay;

}

String getSchool() {

**return** school;

}

**public** String toString() {

**return** "Staff[Person[name="+getName()+",address="+getAddress()+"], school="+school+", pay="+pay + "]";

}

}

**public** **class** Practice {

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

Student obj1 = **new** Student("jeevi", "123", "abcd school",2023,500);

Staff obj2 = **new** Staff("jee", "456", "ssp school",300);

System.***out***.println(obj1.toString());

System.***out***.println(obj2.toString());

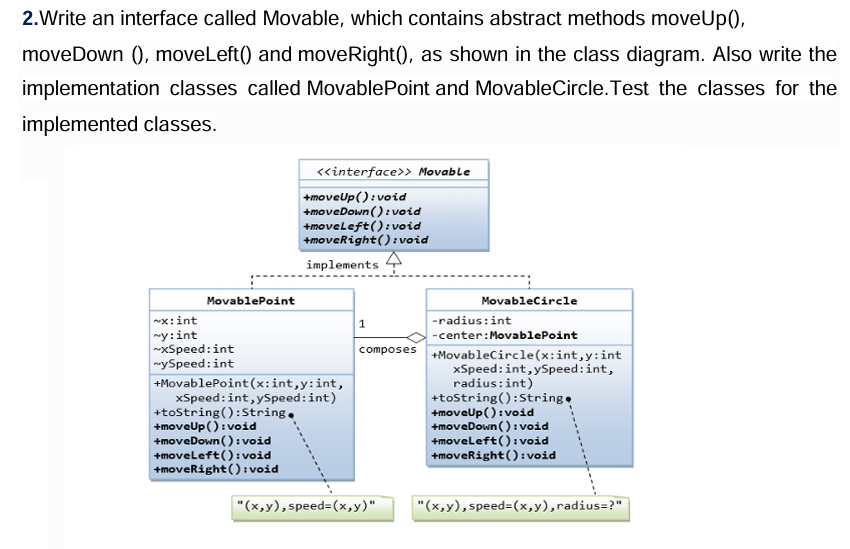
}

}

Output:

Student[Person[name=jeevi,address=123],program=abcd school, year=2023,fee=500.0]

Staff[Person[name=jee,address=456], school=ssp school, pay=300.0]



**package** oops.practice;

**interface** Movable {

**void** moveUp();

**void** moveDown();

**void** moveLeft();

**void** moveRight();

}

**class** MovablePoint **implements** Movable {

**int** x, y, xSpeed, ySpeed;

MovablePoint(**int** x, **int** y, **int** xSpeed, **int** ySpeed) {

**this**.x = x;

**this**.y = y;

**this**.xSpeed = xSpeed;

**this**.ySpeed = ySpeed;

}

@Override

**public** **void** moveUp() {

y += ySpeed;

}

@Override

**public** **void** moveDown() {

y -= ySpeed;

}

@Override

**public** **void** moveLeft() {

x -= xSpeed;

}

@Override

**public** **void** moveRight() {

x += xSpeed;

}

@Override

**public** String toString() {

**return** "(" + x + "," + y + "), speed=(" + xSpeed + "," + ySpeed + ")";

}

}

**class** MovableCircle **implements** Movable {

**int** radius;

MovablePoint center;

MovableCircle(**int** x, **int** y, **int** xSpeed, **int** ySpeed, **int** radius) {

**this**.center = **new** MovablePoint(x, y, xSpeed, ySpeed);

**this**.radius = radius;

}

@Override

**public** **void** moveUp() {

center.moveUp();

}

@Override

**public** **void** moveDown() {

center.moveDown();

}

@Override

**public** **void** moveLeft() {

center.moveLeft();

}

@Override

**public** **void** moveRight() {

center.moveRight();

}

@Override

**public** String toString() {

**return** center + ", radius=" + radius;

}

}

**public** **class** TestMovable {

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

MovablePoint point = **new** MovablePoint(2, 3, 1, 1);

System.***out***.println("Initial Point: " + point);

point.moveUp();

point.moveRight();

System.***out***.println("After moving: " + point);

MovableCircle circle = **new** MovableCircle(5, 5, 2, 2, 10);

System.***out***.println("\nInitial Circle: " + circle);

circle.moveDown();

circle.moveLeft();

System.***out***.println("After moving: " + circle);

}

}

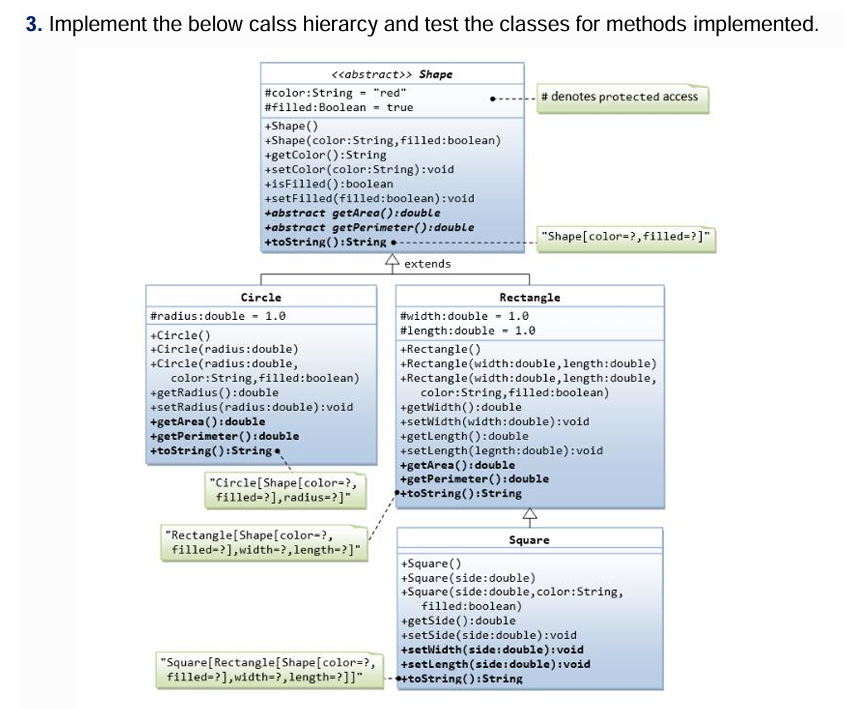
Output:

Initial Point: (2,3), speed=(1,1)

After moving: (3,4), speed=(1,1)

Initial Circle: (5,5), speed=(2,2), radius=10

After moving: (3,3), speed=(2,2), radius=10



**package** oops.practice;

**abstract** **class** Shape {

**protected** String color = "red";

**protected** **boolean** filled = **true**;

**public** Shape() {}

**public** Shape(String color, **boolean** filled) {

**this**.color = color;

**this**.filled = filled;

}

**public** String getColor() {

**return** color;

}

**public** **void** setColor(String color) {

**this**.color = color;

}

**public** **boolean** isFilled() {

**return** filled;

}

**public** **void** setFilled(**boolean** filled) {

**this**.filled = filled;

}

**public** **abstract** **double** getArea();

**public** **abstract** **double** getPerimeter();

@Override

**public** String toString() {

**return** "Shape[color=" + color + ", filled=" + filled + "]";

}

}

**class** Circle **extends** Shape {

**protected** **double** radius = 1.0;

**public** Circle() {}

**public** Circle(**double** radius) {

**this**.radius = radius;

}

**public** Circle(**double** radius, String color, **boolean** filled) {

**super**(color, filled);

**this**.radius = radius;

}

**public** **double** getRadius() {

**return** radius;

}

**public** **void** setRadius(**double** radius) {

**this**.radius = radius;

}

@Override

**public** **double** getArea() {

**return** Math.***PI*** \* radius \* radius;

}

@Override

**public** **double** getPerimeter() {

**return** 2 \* Math.***PI*** \* radius;

}

@Override

**public** String toString() {

**return** "Circle[" + **super**.toString() + ", radius=" + radius + "]";

}

}

**class** Rectangle **extends** Shape {

**protected** **double** width = 1.0;

**protected** **double** length = 1.0;

**public** Rectangle() {}

**public** Rectangle(**double** width, **double** length) {

**this**.width = width;

**this**.length = length;

}

**public** Rectangle(**double** width, **double** length, String color, **boolean** filled) {

**super**(color, filled);

**this**.width = width;

**this**.length = length;

}

**public** **double** getWidth() {

**return** width;

}

**public** **void** setWidth(**double** width) {

**this**.width = width;

}

**public** **double** getLength() {

**return** length;

}

**public** **void** setLength(**double** length) {

**this**.length = length;

}

@Override

**public** **double** getArea() {

**return** width \* length;

}

@Override

**public** **double** getPerimeter() {

**return** 2 \* (width + length);

}

@Override

**public** String toString() {

**return** "Rectangle[" + **super**.toString() + ", width=" + width + ", length=" + length + "]";

}

}

**class** Square **extends** Rectangle {

**public** Square() {}

**public** Square(**double** side) {

**super**(side, side);

}

**public** Square(**double** side, String color, **boolean** filled) {

**super**(side, side, color, filled);

}

**public** **double** getSide() {

**return** width;

}

**public** **void** setSide(**double** side) {

**this**.width = **this**.length = side;

}

@Override

**public** **void** setWidth(**double** side) {

**this**.width = **this**.length = side;

}

@Override

**public** **void** setLength(**double** side) {

**this**.width = **this**.length = side;

}

@Override

**public** String toString() {

**return** "Square[" + **super**.toString() + "]";

}

}

**public** **class** Abstract {

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

Shape circle = **new** Circle(5.5, "blue", **false**);

System.***out***.println(circle);

System.***out***.println("Area: " + circle.getArea());

System.***out***.println("Perimeter: " + circle.getPerimeter());

Shape rectangle = **new** Rectangle(4.0, 5.0, "green", **true**);

System.***out***.println(rectangle);

System.***out***.println("Area: " + rectangle.getArea());

System.***out***.println("Perimeter: " + rectangle.getPerimeter());

Shape square = **new** Square(3.0, "yellow", **false**);

System.***out***.println(square);

System.***out***.println("Area: " + square.getArea());

System.***out***.println("Perimeter: " + square.getPerimeter());

}

}

Output:

Circle[Shape[color=blue, filled=false], radius=5.5]

Area: 95.03317777109123

Perimeter: 34.55751918948772

Rectangle[Shape[color=green, filled=true], width=4.0, length=5.0]

Area: 20.0

Perimeter: 18.0

Square[Rectangle[Shape[color=yellow, filled=false], width=3.0, length=3.0]]

Area: 9.0

Perimeter: 12.0

4.Banking: All the banks operating in India are controlled by RBI. RBI has set a well-defined guideline (e.g. minimum interest rate, minimum balance allowed, maximum withdrawal limit etc) which all banks must follow. For example, suppose RBI has set minimum interest rate applicable to a saving bank account to be 4% annually; however, banks are free to use 4% interest rate or to set any rates above it. Write a JAVA program to implement bank functionality in the above scenario and demonstrate the dynamic polymorphism concept. Note: Create few classes namely Customer, Account, RBI (Base Class) and few derived classes (SBI, ICICI, PNB etc). Assume and implement required member variables and functions in each class.

**package** oops.practice;

**class** Customer {

**private** String name;

**private** String customerID;

**public** Customer(String name, String customerID) {

**this**.name = name;

**this**.customerID = customerID;

}

**public** String getName() {

**return** name;

}

**public** String getCustomerID() {

**return** customerID;

}

@Override

**public** String toString() {

**return** "Customer[Name: " + name + ", ID: " + customerID + "]";

}

}

**class** Account {

**private** String accountNumber;

**private** **double** balance;

**public** Account(String accountNumber, **double** balance) {

**this**.accountNumber = accountNumber;

**this**.balance = balance;

}

**public** String getAccountNumber() {

**return** accountNumber;

}

**public** **double** getBalance() {

**return** balance;

}

**public** **void** deposit(**double** amount) {

balance += amount;

System.***out***.println("Deposit"+ amount + "New Balance" + balance);

}

**public** **void** withdraw(**double** amount) {

**if** (amount <= balance) {

balance -= amount;

System.***out***.println("Withdraw" + amount + "Remaining Balance" + balance);

} **else** {

System.***out***.println("Insufficient balance");

}

}

@Override

**public** String toString() {

**return** "Account[Number: " + accountNumber + ", Balance: ₹" + balance + "]";

}

}

**class** RBI {

**protected** Customer customer;

**protected** Account account;

**public** **double** getInterestRate() {

**return** 4.0;

}

**public** **double** getWithdrawalLimit() {

**return** 50000;

}

**public** RBI(Customer customer, Account account) {

**this**.customer = customer;

**this**.account = account;

}

**public** **void** displayDetails() {

System.***out***.println(customer);

System.***out***.println(account);

System.***out***.println("Interest Rate" + getInterestRate());

System.***out***.println("Withdraw" + getWithdrawalLimit());

}

}

**class** SBI **extends** RBI {

**public** SBI(Customer customer, Account account) {

**super**(customer, account);

}

@Override

**public** **double** getInterestRate() {

**return** 5.0;

}

@Override

**public** **double** getWithdrawalLimit() {

**return** 60000;

}

}

**class** ICICI **extends** RBI {

**public** ICICI(Customer customer, Account account) {

**super**(customer, account);

}

@Override

**public** **double** getInterestRate() {

**return** 6.5;

}

@Override

**public** **double** getWithdrawalLimit() {

**return** 75000;

}

}

**class** PNB **extends** RBI {

**public** PNB(Customer customer, Account account) {

**super**(customer, account);

}

@Override

**public** **double** getInterestRate() {

**return** 5.5;

}

@Override

**public** **double** getWithdrawalLimit() {

**return** 55000;

}

}

**public** **class** Banking {

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

Customer cust1 = **new** Customer("jeev", "CUST101");

Customer cust2 = **new** Customer("dharani", "CUST102");

Customer cust3 = **new** Customer("gayathri", "CUST103");

Account acc1 = **new** Account("SBIN001", 50000);

Account acc2 = **new** Account("ICIC002", 75000);

Account acc3 = **new** Account("PNB003", 60000);

RBI bank1 = **new** SBI(cust1, acc1);

RBI bank2 = **new** ICICI(cust2, acc2);

RBI bank3 = **new** PNB(cust3, acc3);

System.***out***.println("SBI Bank");

bank1.displayDetails();

System.***out***.println("\nICICI Bank");

bank2.displayDetails();

System.***out***.println("\nPNB Bank");

bank3.displayDetails();

}

}

Output:

SBI Bank

Customer[Name: jeev, ID: CUST101]

Account[Number: SBIN001, Balance: ₹50000.0]

Interest Rate5.0

Withdraw60000.0

ICICI Bank

Customer[Name: dharani, ID: CUST102]

Account[Number: ICIC002, Balance: ₹75000.0]

Interest Rate6.5

Withdraw75000.0

PNB Bank

Customer[Name: gayathri, ID: CUST103]

Account[Number: PNB003, Balance: ₹60000.0]

Interest Rate5.5

Withdraw55000.0

5. Painting Shapes:

Develop a class hierarchy of shapes and write a program that computes the amount

of paint needed to paint different objects. The hierarchy will consist of a parent class

Shape with three derived classes - Sphere, Rectangle, and Cylinder. For the purposes

of this exercise, the only attribute a shape will have is a name and the method of

interest will be one that computes the area of the shape (surface area in the case of

three-dimensional shapes). Do the following.

1. Write an abstract class Shape with the following properties:

• An instance variable shapeName of type String

• An abstract method area()

• A toString method that returns the name of the shape

2. The file Sphere.java contains a class for a sphere which is a descendant of Shape. A

sphere has a radius and its area (surface area) is given by the formula 4\*PI\*radius^2.

Define similar classes for a rectangle and a cylinder. Both the Rectangle class and the

Cylinder class are descendants of the Shape class. A rectangle is defined by its length

and width and its area is length times width. A cylinder is defined by a radius and

height and its area (surface area) is PI\*radius^2\*height. Define the toString method in

a way similar to that for the Sphere class.

3. The file Paint.java contains a class for a type of paint (which has a "coverage" and a

method to compute the amount of paint needed to paint a shape). Correct the return

statement in the amount method so the correct amount will be returned. Use the fact

that the amount of paint needed is the area of the shape divided by the coverage for

the paint. (NOTE: Leave the print statement - it is there for illustration purposes, so

you can see the method operating on different types of Shape objects.)

4. The file PaintThings.java contains a program that computes the amount of paint

needed to paint various shapes. A paint object has been instantiated.

Add the following to complete the program:

• Instantiate the three shape objects: deck to be a 20 by 35 foot rectangle, bigBall

to be a sphere of radius 15, and tank to be a cylinder of radius 10 and height 30.

• Make the appropriate method calls to assign the correct values to the three

amount variables.

• Run the program and test it. You should see polymorphism in action as the

amount method computes the amount of paint for various shapes

abstract class Shape {

protected String shapeName;

public Shape(String name) {

this.shapeName = name;

}

public abstract double area();

@Override

public String toString() {

return shapeName;

}

}

class Sphere extends Shape {

private double radius;

public Sphere(double radius) {

super("Sphere");

this.radius = radius;

}

@Override

public double area() {

return 4 \* Math.PI \* radius \* radius;

}

}

class Rectangle extends Shape {

private double length, width;

public Rectangle(double length, double width) {

super("Rectangle");

this.length = length;

this.width = width;

}

@Override

public double area() {

return length \* width;

}

}

class Cylinder extends Shape {

private double radius, height;

public Cylinder(double radius, double height) {

super("Cylinder");

this.radius = radius;

this.height = height;

}

@Override

public double area() {

return 2 \* Math.PI \* radius \* (radius + height);

}

}

class Paint {

private double coverage;

public Paint(double coverage) {

this.coverage = coverage;

}

public double amount(Shape s) {

System.out.println("Computing amount for " + s);

return s.area() / coverage;

}

}

public class PaintThings {

public static void main(String[] args) {

Paint paint = new Paint(350); // 350 square feet per gallon

Shape deck = new Rectangle(20, 35);

Shape bigBall = new Sphere(15);

Shape tank = new Cylinder(10, 30);

double deckPaint = paint.amount(deck);

double bigBallPaint = paint.amount(bigBall);

double tankPaint = paint.amount(tank);

System.out.printf("Paint needed:\nDeck: %.2f gallons\nBig Ball: %.2f gallons\nTank: %.2f gallons\n",

deckPaint, bigBallPaint, tankPaint);

}

}