**Assignment 7**

// Q1. We Need to create a Drawing Application.

// Shapes can be Circle, Square, Rectangle etc.

// And we have common features which will be in a parent class (Shape Class). Use Upcasting for Common Feature Access

// and Downcasting for Specific feature access, create Polymorphic method for common operations.

class Shape {

    int height;

    int width;

    Shape() {

        height = 0;

        width = 0;

    }

    Shape(int height, int width) {

        this.height = height;

        this.width = width;

    }

    void getArea() {

        System.out.println("The area of the Shape class");

    }

    void getPerimeter() {

        System.out.println("The perimeter of Shape class");

    }

}

class Circle extends Shape {

    int radius;

    Circle() {

        radius = 0;

    }

    Circle(int radius) {

        this.radius = radius;

    }

    @Override

    void getArea() {

        System.out.println("The area of circle is-:" + Math.PI \* Math.pow(radius, 2));

    }

    @Override

    void getPerimeter() {

        System.out.println("The perimenter of circle is-:" + 2 \* Math.PI \* radius);

    }

}

class Square extends Shape {

    Square(int height) {

        super.height = height;

    }

    @Override

    void getArea() {

        System.out.println("The area of square-:" + Math.pow(super.height, 2));

    }

    @Override

    void getPerimeter() {

        System.out.println("The perimeter of perimeter-:" + 4 \* super.height);

    }

}

class Rectangle extends Shape {

    Rectangle(int height, int width) {

        super.height = height;

        super.width = width;

    }

    @Override

    void getArea() {

        System.out.println("The area of rectangle-:" + super.height \* super.width);

    }

    @Override

    void getPerimeter() {

        System.out.println("The perimeter of rectangle-:" + 2 \* (super.height + super.width));

    }

}

class ShapeCaller {

    // Polymorphic Function.........

    void callShape(Shape sh) {

        // sh = new Shape(200, 200);

        sh.getArea();

        sh.getPerimeter();

    }

}

public class Draw {

    public static void main(String[] args) {

        // No dry principle

        // DRY Principle

        ShapeCaller sh = new ShapeCaller();

        // upcasting

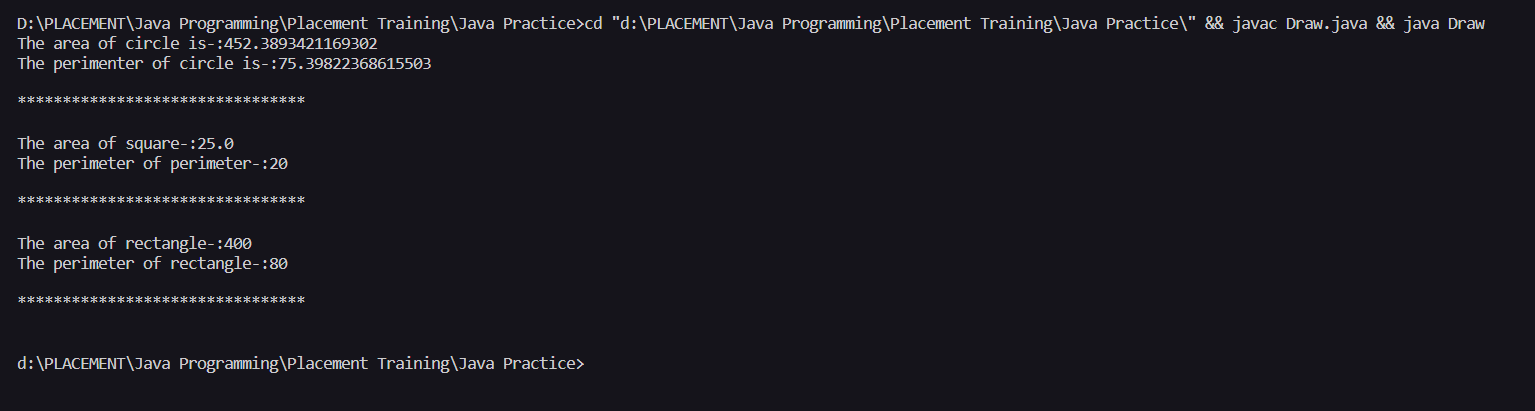
        sh.callShape(new Circle(12));

        sh.callShape(new Square(5));

        sh.callShape(new Rectangle(20, 20));

    }

}

****

Home Loan Account , Education Loan Account, Car Loan Account , We need to write a Solution which make

relationship among the classes by Using IS A or has A. So Customer can take Loan , based on the Loan He/She choose we need to prepare

it's EMI Plan. \*/

class Loan\_Account {

    int principle;

    int roi;

    int tenure;

    public Loan\_Account(int principle, int roi, int tenure) {

        this.principle = principle;

        this.roi = roi;

        this.tenure = tenure;

    }

    void rateOfInterest() {

        System.out.println("The rate of interest is-:" + roi);

    }

    void loanAmount() {

        System.out.println("The amount taken as a loan is-:" + principle);

    }

    void emiCalculator() {

        System.out.println("The emi need to be paid-:"

                + ((principle \* roi \* (float) Math.pow(1 + roi, tenure))

                        / (float) (Math.pow(1 + roi, tenure) - 1)));

    }

}

class Home\_Loan\_Account extends Loan\_Account {

    String location;

    public Home\_Loan\_Account(int principle, int roi, int tenure, String location) {

        super(principle, roi, tenure);

        this.location = location;

        // TODO Auto-generated constructor stub

    }

    void location() {

        System.out.println("The location of home is-:" + location);

    }

}

class Education\_Loan\_Account extends Loan\_Account {

    String name;

    public Education\_Loan\_Account(int principle, int roi, int tenure, String name) {

        super(principle, roi, tenure);

        this.name = name;

        // TODO Auto-generated constructor stub

    }

    void studentName() {

        System.out.println("The name of the student is-:" + name);

    }

}

class Car\_Loan\_Account extends Loan\_Account {

    String info;

    public Car\_Loan\_Account(int principle, int roi, int tenure, String info) {

        super(principle, roi, tenure);

        // TODO Auto-generated constructor stub

        this.info = info;

    }

    void numberPlateInfo() {

        System.out.println("The number plate info is-:" + info);

    }

}

class LoanCaller {

    void callLoan(Loan\_Account al) {

        al.loanAmount();

        al.rateOfInterest();

        al.emiCalculator();

        // Downcasting

        if (al instanceof Home\_Loan\_Account)

            ((Home\_Loan\_Account) al).location();

        else if (al instanceof Education\_Loan\_Account)

            ((Education\_Loan\_Account) al).studentName();

        else if (al instanceof Car\_Loan\_Account)

            ((Car\_Loan\_Account) al).numberPlateInfo();

        System.out.println();

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

        System.out.println();

    }

}

public class Loan {

    public static void main(String[] args) {

        LoanCaller lc = new LoanCaller();

        // Here we are doing upcasting...

        lc.callLoan(new Home\_Loan\_Account(10000, 5, 2, "Haldwani"));

        lc.callLoan(new Education\_Loan\_Account(20000, 12, 3, "Harry Lewis"));

        lc.callLoan(new Car\_Loan\_Account(15000, 15, 4, "UK07-SE-001"));

    }

}

