**Lab 9, Inheritance and Polymorphism**

**Name: Shahzaman Abbasi**

**CMS ID: 023-20-0122**

**Exercise 1:**

**Code:**

class Message {

    String text;

    Message(String *text*) {

        this.text = text;

    }

    public void setText(String *text*) {

        this.text = text;

    }

    public String toString() {

        return this.text;

    }

}

class SMS extends Message {

    String recipientContactNo;

    SMS(String *text*, String *recipientContactNo*) {

        super(text);

        this.recipientContactNo = recipientContactNo;

    }

    public void setRecipientContactNo(String *recipientContactNo*) {

        this.recipientContactNo = recipientContactNo;

    }

    public String getRecipientContactNo() {

        return recipientContactNo;

    }

    @Override

    public String toString() {

        return "SMS: " + getRecipientContactNo() + " " + text;

    }

}

class Email extends Message {

    String sender;

    String receiver;

    String subject;

    Email(String *text*, String *sender*, String *receiver*, String *subject*) {

        super(text);

        this.sender = sender;

        this.receiver = receiver;

        this.subject = subject;

    }

    public void setSender(String *sender*) {

        this.sender = sender;

    }

    public void setReceiver(String *receiver*) {

        this.receiver = receiver;

    }

    public void setSubject(String *subject*) {

        this.subject = subject;

    }

    public String getSender() {

        return sender;

    }

    public String getReceiver() {

        return receiver;

    }

    public String getSubject() {

        return subject;

    }

    @Override

    public String toString() {

        return "Email: \nSender: " + getSender() + " Receiver: " + getReceiver() + " " + getSubject() + " " + "\n" + text;

    }

}

public class task1 {

    static boolean ContainsKeyword(Message *messageObject*, String *keyword*) {

        if (messageObject.toString().indexOf(keyword) >= 0)

            return true;

        return false;

    }

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

        Message msg = new Message("This is Java ");

        Email email = new Email("This is java", "sender@email.com", "receiver@email.com", "How are you bro......?");

        SMS sms = new SMS("How are you bro.....?", "0300-1231231");

        System.out.println(sms);

        System.out.println("");

        System.out.println(email);

        System.out.println("");

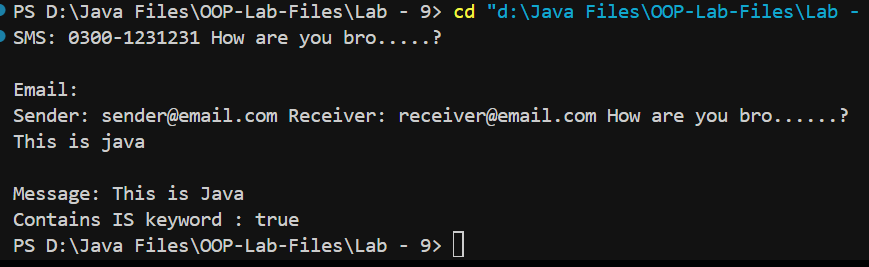
        System.out.println("Message: " + msg);

        System.out.println("Contains IS keyword : " + ContainsKeyword(msg ,"is"));

    }

}

**Output:**

****

**Exercise 2:**

**Code:**

class Employee {

    private String firstName;

    private String lastName;

    private String CNIC;

    public Employee() {}

    public Employee(String *firstName*, String *lastName*, String *CNIC*) {

        this.firstName = firstName;

        this.lastName = lastName;

        this.CNIC = CNIC;

    }

    public String getFirstName() {

        return firstName;

    }

    public void setFirstName(String *firstName*) {

        this.firstName = firstName;

    }

    public String getLastName() {

        return lastName;

    }

    public void setLastName(String *lastName*) {

        this.lastName = lastName;

    }

    public String getCNIC() {

        return CNIC;

    }

    public void setCNIC(String *CNIC*) {

        this.CNIC = CNIC;

    }

    @Override

    public String toString() {

        return firstName + " " + lastName + " CNIC# " + CNIC;

    }

    public double earnings() {

        return 0.00;

    }

}

class SalariedEmployee extends Employee {

    private double weeklySalary;

    public SalariedEmployee() {}

    public SalariedEmployee(String *firstName*, String *lastName*, String *CNIC*, double *weeklySalary*) {

        super(firstName, lastName, CNIC);

        if (weeklySalary >= 0)

            this.weeklySalary = weeklySalary;

        else

            System.out.println("Weekly salary cannot be negative.");

    }

    public double getWeeklySalary() {

        return weeklySalary;

    }

    public void setWeeklySalary(double *weeklySalary*) {

        if (weeklySalary >= 0)

            this.weeklySalary = weeklySalary;

        else

        System.out.println(

            "Weekly salary cannot be negative."

        );

    }

    @Override

    public String toString() {

        return "\nSalaried employee: " + super.toString();

    }

    @Override

    public double earnings() {

        return weeklySalary;

    }

}

class HourlyEmployee extends Employee {

    private double wage;

    private double hours;

    public HourlyEmployee(String *firstName*, String *lastName*, String *CNIC*, double *wage*, double *hours*) {

        super(firstName, lastName, CNIC);

        if (wage >= 0)

            this.wage = wage;

        else

            throw new IllegalArgumentException("Wage cannot be negative.");

        if (hours >= 0)

            this.hours = hours;

        else

            throw new IllegalArgumentException("Hours cannot be negative.");

    }

    public double getWage() {

        return wage;

    }

    public void setWage(double *wage*) {

        if (wage >= 0)

            this.wage = wage;

        else

            throw new IllegalArgumentException("Wage cannot be negative.");

    }

    public double getHours() {

        return hours;

    }

    public void setHours(double *hours*) {

        if (hours >= 0)

            this.hours = hours;

        else

            throw new IllegalArgumentException("Hours cannot be negative.");

    }

    @Override

    public String toString() {

        return "\nHourly employee: " + super.toString();

    }

    @Override

    public double earnings() {

        if (hours <= 40) {

            return wage \* hours;

        } else {

            return 40 \* wage + (hours - 40) \* wage \* 1.5;

        }

    }

}

class CommissionEmployee extends Employee {

    private double grossSales;

    private double commissionRate;

    public CommissionEmployee(String *firstName*, String *lastName*, String *CNIC*, double *grossSales*, double *commissionRate*) {

        super(firstName, lastName, CNIC);

        if (grossSales >= 0)

            this.grossSales = grossSales;

        else

            System.out.println("Error: Gross sales cannot be negative.");

        if (commissionRate >= 0)

            this.commissionRate = commissionRate;

        else

            System.out.println("Error: Commission rate cannot be negative.");

    }

    public double getGrossSales() {

        return grossSales;

    }

    public void setGrossSales(double *grossSales*) {

        if (grossSales >= 0)

            this.grossSales = grossSales;

        else

            System.out.println("Error: Gross sales cannot be negative.");

    }

    public double getCommissionRate() {

        return commissionRate;

    }

    public void setCommissionRate(double *commissionRate*) {

        if (commissionRate >= 0)

            this.commissionRate = commissionRate;

        else

            System.out.println("Error: Commission rate cannot be negative.");

    }

    @Override

    public String toString() {

        return "\nCommission employee: " + super.toString();

    }

    @Override

    public double earnings() {

        return grossSales \* commissionRate;

    }

}

class BasePlusCommissionEmployee extends CommissionEmployee {

    private double baseSalary;

    public BasePlusCommissionEmployee(String *firstName*, String *lastName*, String *CNIC*, double *grossSales*, double *commissionRate*, double *baseSalary*) {

        super(firstName, lastName, CNIC, grossSales, commissionRate);

        if (baseSalary >= 0)

            this.baseSalary = baseSalary;

        else

            System.out.println("Error: Base salary cannot be negative.");

    }

    public double getBaseSalary() {

        return baseSalary;

    }

    public void setBaseSalary(double *baseSalary*) {

        if (baseSalary >= 0)

            this.baseSalary = baseSalary;

        else

            System.out.println("Error: Base salary cannot be negative.");

    }

    @Override

    public String toString() {

        return "\nBase plus Commission employee: " + super.toString();

    }

    @Override

    public double earnings() {

        return baseSalary + super.earnings();

    }

}

public class PayRollSystemTest {

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

        Employee firstEmployee = new SalariedEmployee("Muhammad", "Ali", "11111-1111", 800.00);

        Employee secondEmployee = new CommissionEmployee("Tarwan", "Kumar", "222-22-2222", 10000, 0.06);

        Employee thirdEmployee = new BasePlusCommissionEmployee("Fabeeha", "Fatima", "333-33-3333", 5000, 0.04, 300);

        Employee fourthEmployee = new HourlyEmployee("Hasnain", "Ali", "444-44-4444", 16.75, 40);

        System.out.println(firstEmployee);

        System.out.println("Earnings: " + firstEmployee.earnings());

        System.out.println(secondEmployee);

        System.out.println("Earnings: " + secondEmployee.earnings());

        System.out.println(thirdEmployee);

        BasePlusCommissionEmployee currentEmployee = (BasePlusCommissionEmployee) thirdEmployee;

        double oldBaseSalary = currentEmployee.getBaseSalary();

        System.out.println("Old base salary: " + oldBaseSalary);

        currentEmployee.setBaseSalary(1.10 \* oldBaseSalary);

        System.out.println("New base salary with 10% increase is: " + currentEmployee.getBaseSalary());

        System.out.println("Earnings: " + thirdEmployee.earnings());

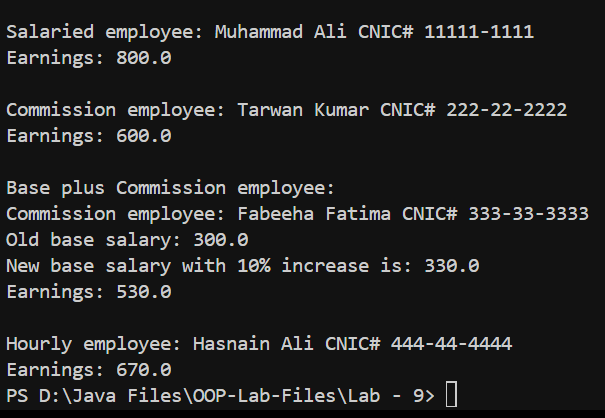
        System.out.println(fourthEmployee);

        System.out.println("Earnings: " + fourthEmployee.earnings());

    }

}

**Output:**



**Exercise 3:**

**Code:**

import java.util.Scanner;

class Point {

    private int x;

    private int y;

    public Point(int *x*, int *y*) {

        this.x = x;

        this.y = y;

    }

    public int getX() {

        return x;

    }

    public int getY() {

        return y;

    }

    public double distanceTo(Point *p*) {

        return Math.sqrt((x - p.getX()) \* (x - p.getX()) + (y - p.getY()) \* (y - p.getY()));

    }

    public String toString() {

        return "(" + x + ", " + y + ")";

    }

}

class Shape {

    protected Point center;

    public Shape(Point *center*) {

        this.center = center;

    }

    public boolean isInside(Point *p*) {

        return false;

    }

}

class Circle extends Shape {

    private int radius;

    public Circle(Point *center*, int *radius*) {

        super(center);

        this.radius = radius;

    }

    public int getRadius() {

        return radius;

    }

    @Override

    public String toString() {

        return "Circle: Center " + center.toString() + ", Radius " + radius;

    }

    @Override

    public boolean isInside(Point *p*) {

        return center.distanceTo(p) <= radius;

    }

}

class Rectangle extends Shape {

    private int length;

    private int width;

    public Rectangle(Point *center*, int *length*, int *width*) {

        super(center);

        this.length = length;

        this.width = width;

    }

    public int getLength() {

        return length;

    }

    public int getWidth() {

        return width;

    }

    @Override

    public String toString() {

        return "Rectangle: Center " + center.toString() + ", Length " + length + ", Width " + width;

    }

}

class ShapesArray {

    private Shape[] shapes;

    private int circleCounter;

    public ShapesArray(int *size*) {

        shapes = new Shape[size];

        circleCounter = 0;

    }

    public void addShape(Shape *shape*) {

        if (shape instanceof Circle) {

            circleCounter++;

        }

        for (int i = 0; i < shapes.length; i++) {

            if (shapes[i] == null) {

                shapes[i] = shape;

                return;

            }

        }

        System.out.println("Shapes array is full, cannot add more shapes.");

    }

    public void displayRectsInfo() {

        for (Shape shape : shapes) {

            if (shape instanceof Rectangle) {

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

            }

        }

    }

    public int getCircleCounter() {

        return circleCounter;

    }

    public double getAvgAreas() {

        double totalArea = 0;

        int count = 0;

        for (Shape shape : shapes) {

            if (shape != null) {

                if (shape instanceof Circle) {

                    totalArea += Math.PI \* Math.pow(((Circle) shape).getRadius(), 2);

                    count++;

                } else if (shape instanceof Rectangle) {

                    totalArea += ((Rectangle) shape).getLength() \* ((Rectangle) shape).getWidth();

                    count++;

                }

            }

        }

        if (count == 0) {

            return 0;

        }

        return totalArea / count;

    }

    public void removeAllRect() {

        for (int i = 0; i < shapes.length; i++) {

            if (shapes[i] instanceof Rectangle) {

                shapes[i] = null;

            }

        }

    }

}

public class TestShape {

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

        Scanner scanner = new Scanner(System.in);

        ShapesArray shapesArray = new ShapesArray(20);

        while (true) {

            System.out.println("Options:");

            System.out.println("1. Add new shape");

            System.out.println("   a. Add rectangle");

            System.out.println("   b. Add circle");

            System.out.println("2. Display all rectangles");

            System.out.println("3. Display the average shape area");

            System.out.println("4. Display the number of circles");

            System.out.println("5. Remove all rectangles");

            System.out.println("6. Exit");

            System.out.print("Enter your choice: ");

            int choice = scanner.nextInt();

            switch (choice) {

                case 1:

                    System.out.println("Enter shape type (a for rectangle, b for circle): ");

                    char shapeType = scanner.next().charAt(0);

                    if (shapeType == 'a') {

                        System.out.println("Enter center coordinates (x y): ");

                        int x = scanner.nextInt();

                        int y = scanner.nextInt();

                        Point center = new Point(x, y);

                        System.out.println("Enter length and width: ");

                        int length = scanner.nextInt();

                        int width = scanner.nextInt();

                        shapesArray.addShape(new Rectangle(center, length, width));

                    } else if (shapeType == 'b') {

                        System.out.println("Enter center coordinates (x y): ");

                        int x = scanner.nextInt();

                        int y = scanner.nextInt();

                        Point center = new Point(x, y);

                        System.out.println("Enter radius: ");

                        int radius = scanner.nextInt();

                        shapesArray.addShape(new Circle(center, radius));

                    } else {

                        System.out.println("Invalid choice.");

                    }

                    break;

                case 2:

                    System.out.println("Displaying all rectangles:");

                    shapesArray.displayRectsInfo();

                    break;

                case 3:

                    System.out.println("Average shape area: " + shapesArray.getAvgAreas());

                    break;

                case 4:

                    System.out.println("Number of circles: " + shapesArray.getCircleCounter());

                    break;

                case 5:

                    shapesArray.removeAllRect();

                    System.out.println("All rectangles removed.");

                    break;

                case 6:

                    System.out.println("Exiting program...");

                    scanner.close();

                    System.exit(0);

                default:

                    System.out.println("Invalid choice. Please enter a number between 1 and 6.");

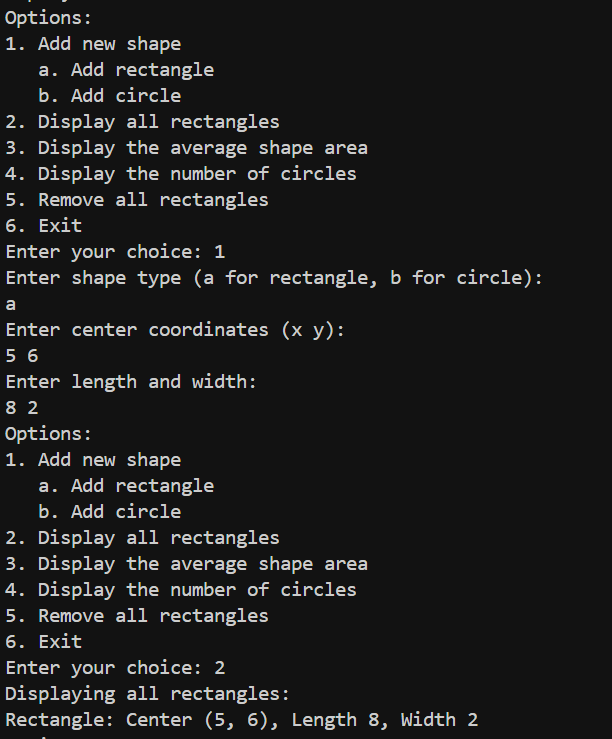
            }

        }

    }

}

**Output:**

****

**Exercise 4:**

**Code:**

class Maths {

    public void display() {

        System.out.println("Hello, I am the display method of class Maths");

    }

}

class Algebra extends Maths {

    @Override

    public void display() {

        System.out.println("Hello, I am the display method of Algebra");

    }

}

public class Exercise4 {

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

*// Upcasting*

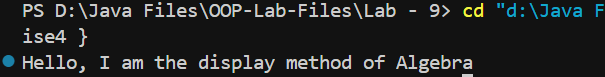
        Maths math = new Algebra();

        math.display();

    }

}

**Output:**

****