**Department of Computing**

**CS 212: Object Oriented Programming**

**Lab 06: Inheritance**

**Lab Engineer: Ms. Bareera Anam**

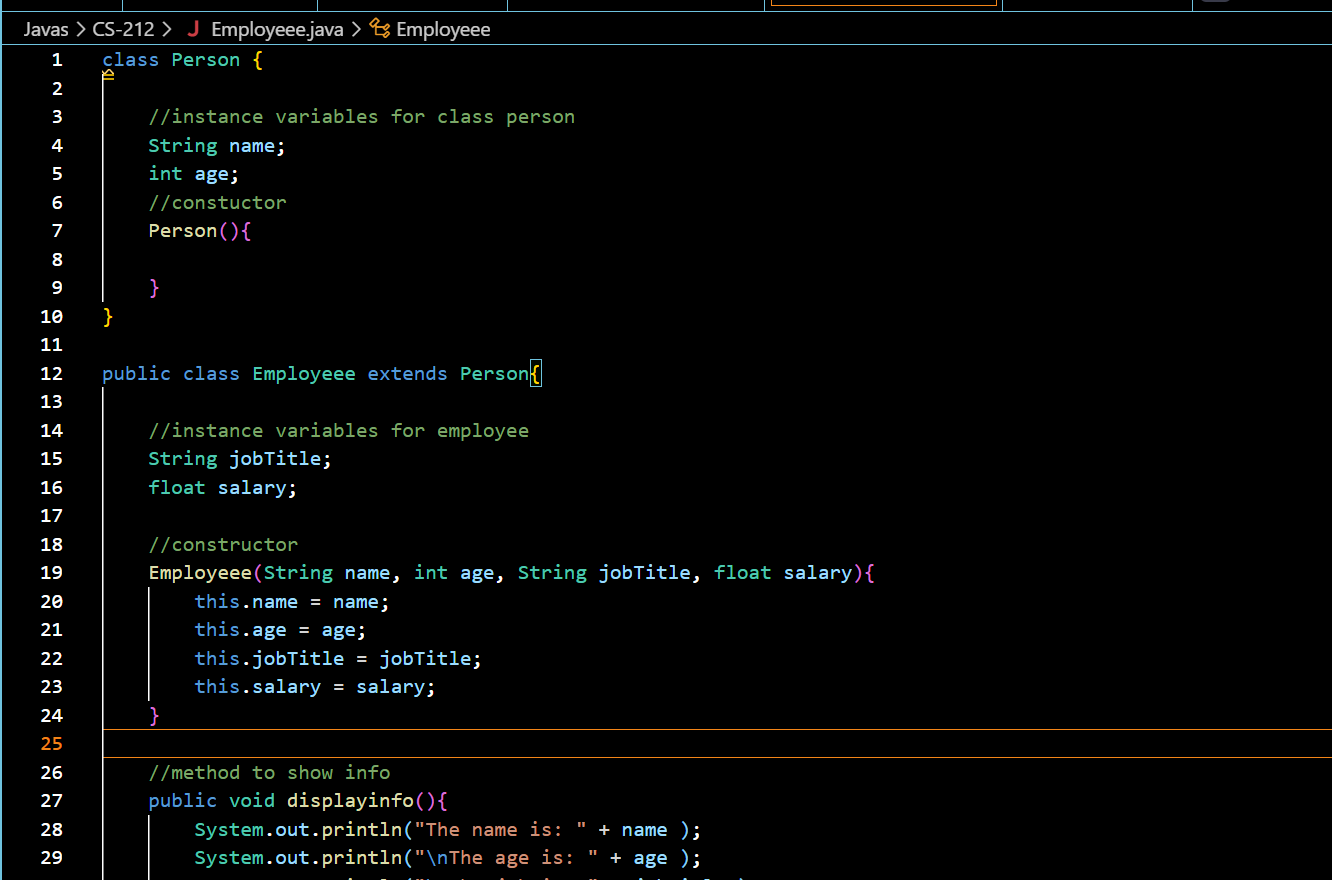
**Class: BSDS -02A**

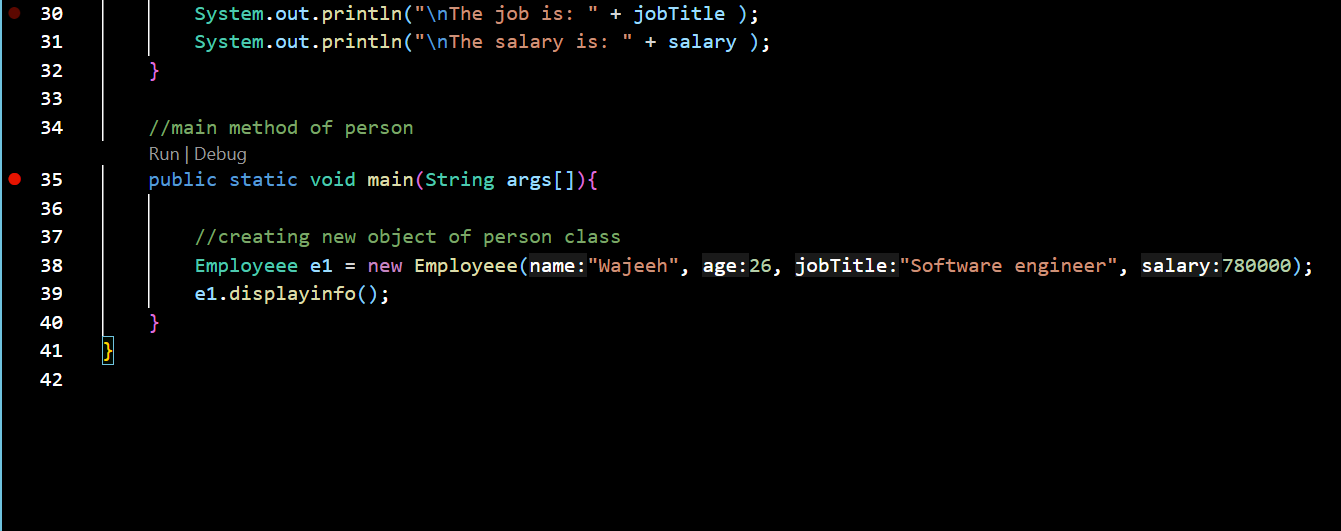
**AILYA ZAINAB**

**( 5 2 3 5 0 6 )**

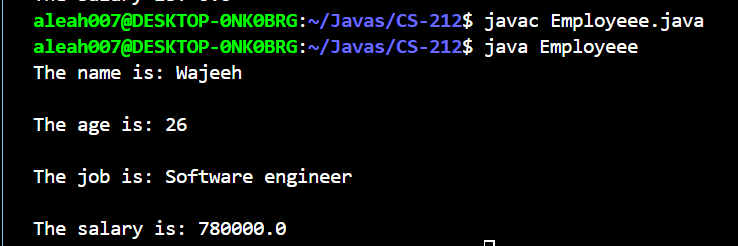
**TASK#01**

**INPUT:**





**OUTPUT:**



CODE:

class Person {

//instance variables for class person

String name;

int age;

//constuctor

Person(){

}

}

public class Employeee extends Person{

//instance variables for employee

String jobTitle;

float salary;

//constructor

Employeee(String name, int age, String jobTitle, float salary){

this.name = name;

this.age = age;

this.jobTitle = jobTitle;

this.salary = salary;

}

//method to show info

public void displayinfo(){

System.out.println("The name is: " + name );

System.out.println("\nThe age is: " + age );

System.out.println("\nThe job is: " + jobTitle );

System.out.println("\nThe salary is: " + salary );

}

//main method of person

public static void main(String args[]){

//creating new object of person class

Employeee e1 = new Employeee("Wajeeh", 26, "Software engineer", 780000);

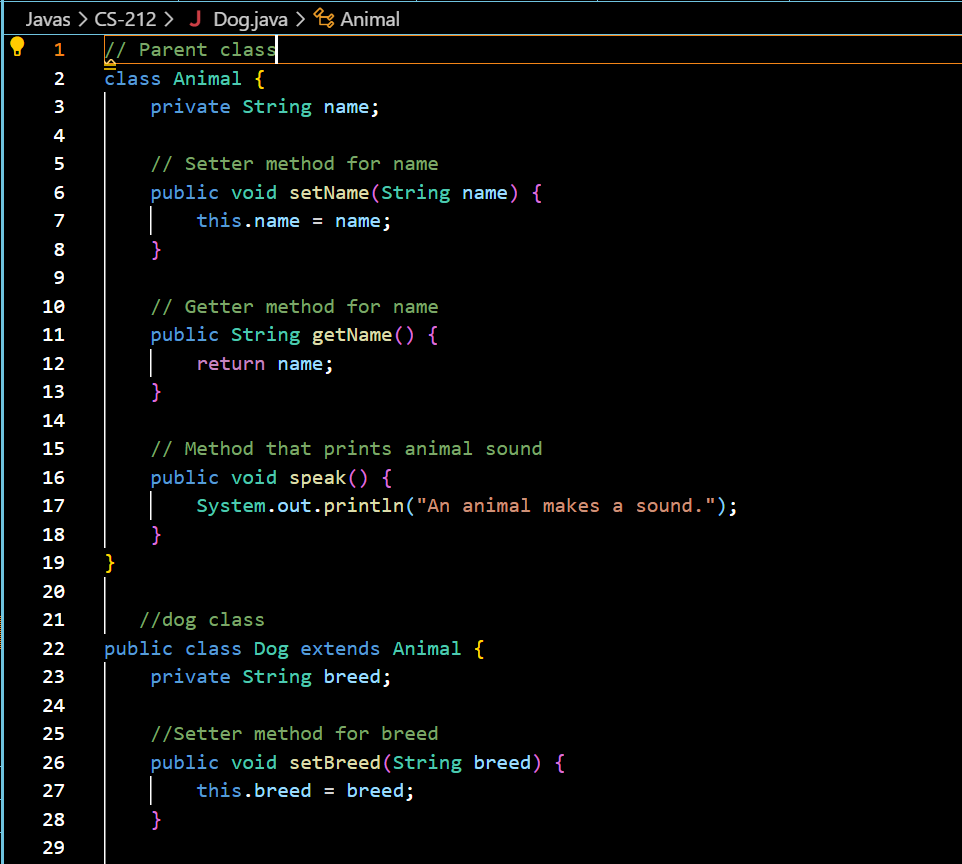
e1.displayinfo();

}

}

**TASK#02**

**INPUT:**





CODE:

// Parent class

class Animal {

private String name;

// Setter method for name

public void setName(String name) {

this.name = name;

}

// Getter method for name

public String getName() {

return name;

}

// Method that prints animal sound

public void speak() {

System.out.println("An animal makes a sound.");

}

}

//dog class

public class Dog extends Animal {

private String breed;

//Setter method for breed

public void setBreed(String breed) {

this.breed = breed;

}

// Getter method for breed

public String getBreed() {

return breed;

}

public void speak() {

System.out.println("A dog barks!");

}

//main method

public static void main(String[] args) {

// Creating Dog object

Dog myDog = new Dog();

// Setting name

myDog.setName("Buddy");

// Setting breed

myDog.setBreed("Golden Retriever");

System.out.println("Dog's Name: " + myDog.getName());

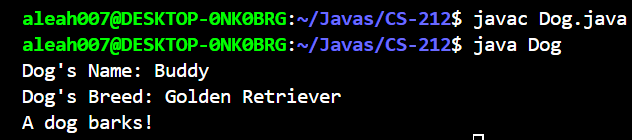
System.out.println("Dog's Breed: " + myDog.getBreed());

myDog.speak();

}

}

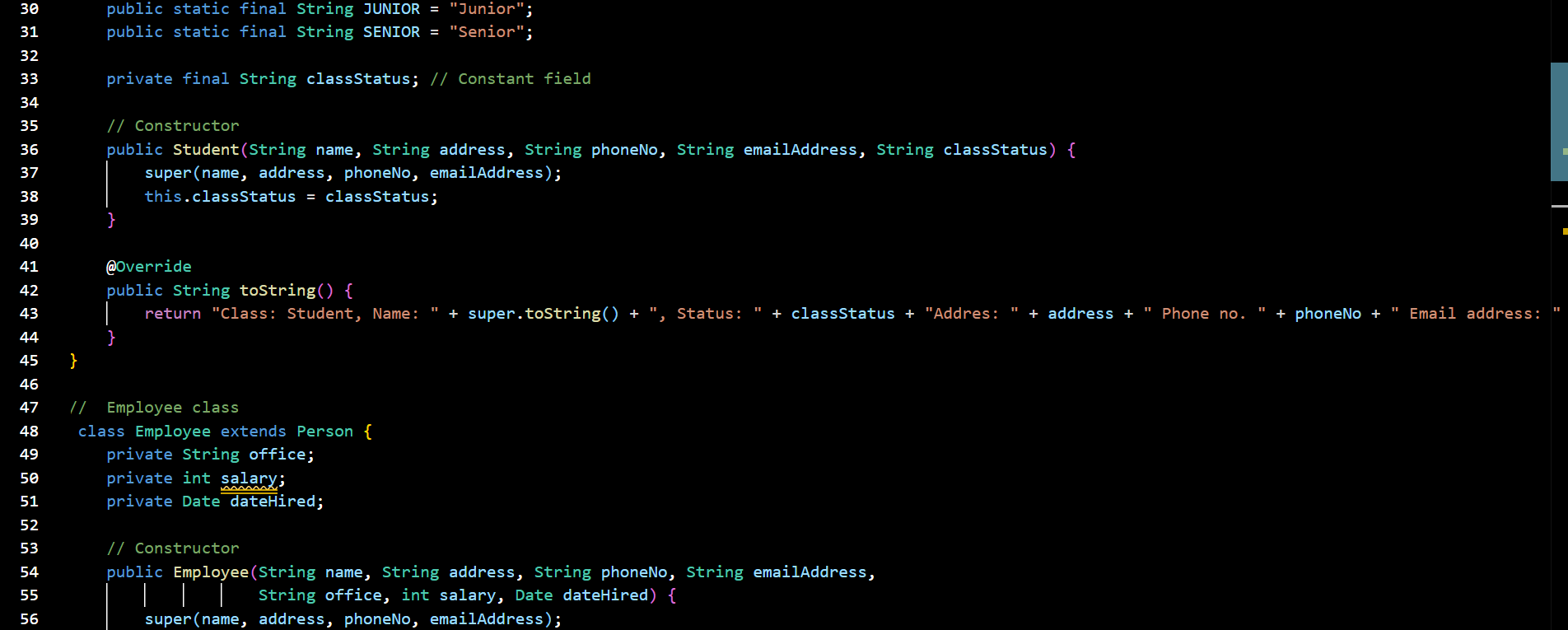
**Output:**

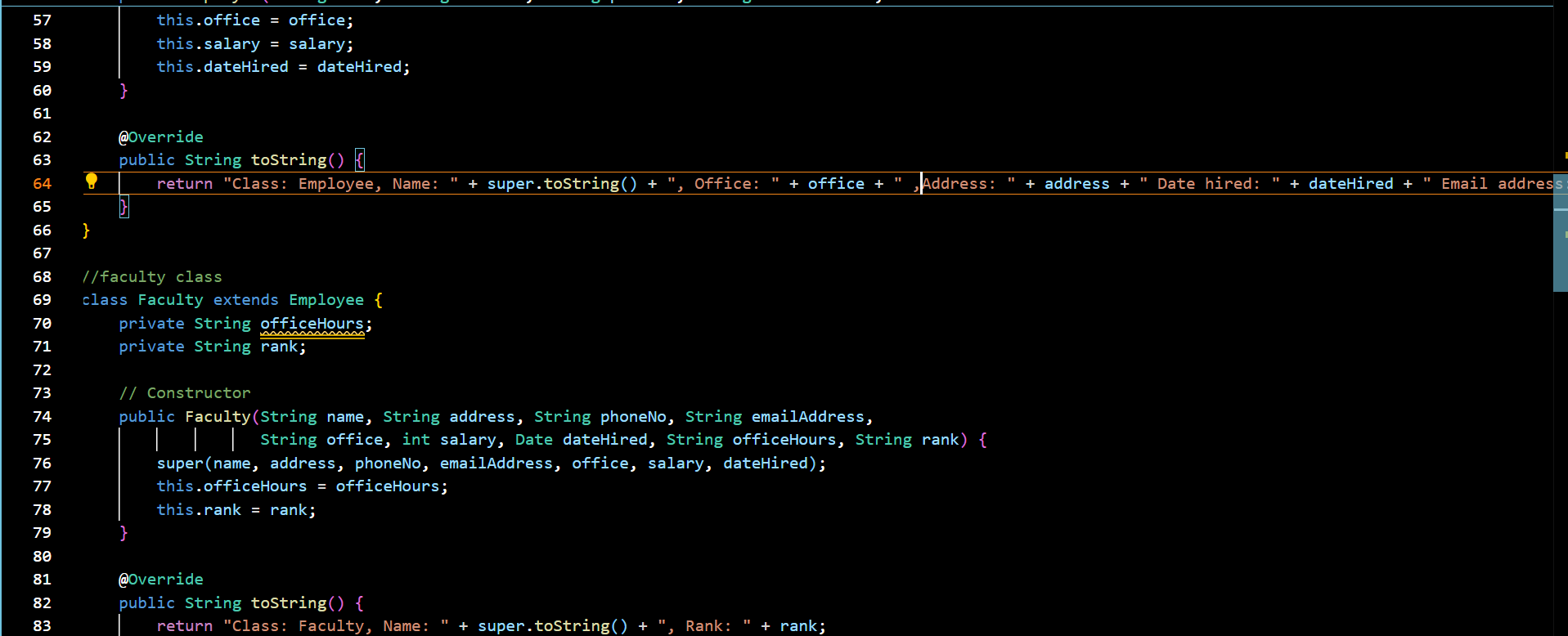


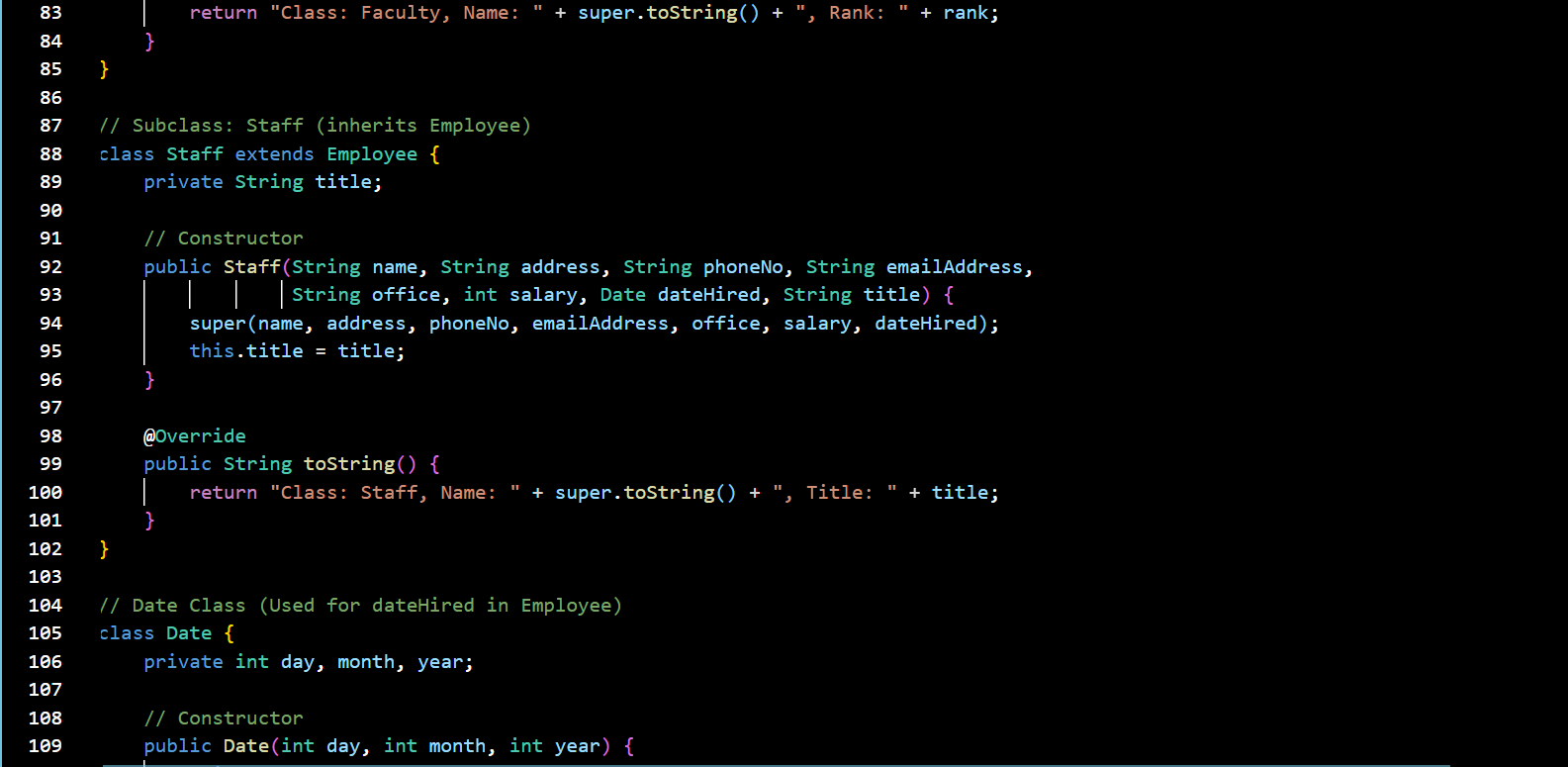
**TASK#03**

**INPUT:**

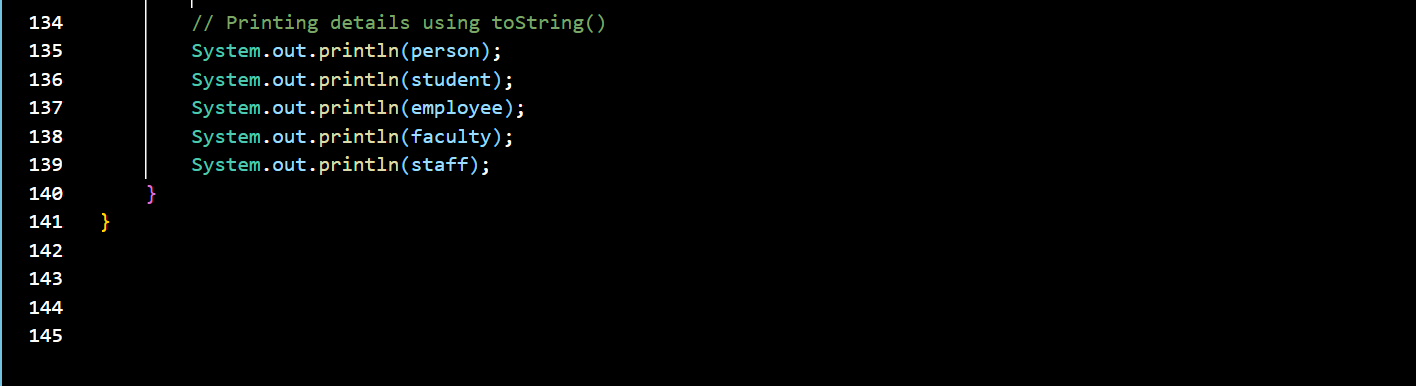












**CODE:**

//base class

class Person {

protected String name;

protected String address;

protected String phoneNo;

protected String emailAddress;

// Constructor

public Person(String name, String address, String phoneNo, String emailAddress) {

this.name = name;

this.address = address;

this.phoneNo = phoneNo;

this.emailAddress = emailAddress;

}

// toString method to override adress and display info

@Override

public String toString() {

return "Class: Person, Name: " + name + " Adress: " + address + "Phone no. " + phoneNo + "EmailAddress: " + emailAddress;

}

}

// student class

class Student extends Person {

// Defining class status constants

public static final String FRESHMAN = "Freshman";

public static final String SOPHOMORE = "Sophomore";

public static final String JUNIOR = "Junior";

public static final String SENIOR = "Senior";

private final String classStatus; // Constant field

// Constructor

public Student(String name, String address, String phoneNo, String emailAddress, String classStatus) {

super(name, address, phoneNo, emailAddress);

this.classStatus = classStatus;

}

@Override

public String toString() {

return "Class: Student, Name: " + super.toString() + ", Status: " + classStatus + "Addres: " + address + " Phone no. " + phoneNo + " Email address: " + emailAddress;

}

}

// Employee class

class Employee extends Person {

private String office;

private int salary;

private Date dateHired;

// Constructor

public Employee(String name, String address, String phoneNo, String emailAddress,

String office, int salary, Date dateHired) {

super(name, address, phoneNo, emailAddress);

this.office = office;

this.salary = salary;

this.dateHired = dateHired;

}

@Override

public String toString() {

return "Class: Employee, Name: " + super.toString() + ", Office: " + office + " ,Address: " + address + " Date hired: " + dateHired + " Email address: " + emailAddress + " Phone no. " + phoneNo;

}

}

//faculty class

class Faculty extends Employee {

private String officeHours;

private String rank;

// Constructor

public Faculty(String name, String address, String phoneNo, String emailAddress,

String office, int salary, Date dateHired, String officeHours, String rank) {

super(name, address, phoneNo, emailAddress, office, salary, dateHired);

this.officeHours = officeHours;

this.rank = rank;

}

@Override

public String toString() {

return "Class: Faculty, Name: " + super.toString() + ", Rank: " + rank;

}

}

// Subclass: Staff (inherits Employee)

class Staff extends Employee {

private String title;

// Constructor

public Staff(String name, String address, String phoneNo, String emailAddress,

String office, int salary, Date dateHired, String title) {

super(name, address, phoneNo, emailAddress, office, salary, dateHired);

this.title = title;

}

@Override

public String toString() {

return "Class: Staff, Name: " + super.toString() + ", Title: " + title;

}

}

// Date Class (Used for dateHired in Employee)

class Date {

private int day, month, year;

// Constructor

public Date(int day, int month, int year) {

this.day = day;

this.month = month;

this.year = year;

}

@Override

public String toString() {

return day + "/" + month + "/" + year;

}

}

// Test Class

public class Test {

public static void main(String[] args) {

// Creating objects of each class

Person person = new Person("Alice", "123 Street", "123456789", "alice@mail.com");

Student student = new Student("Bob", "456 Lane", "987654321", "bob@mail.com", Student.SOPHOMORE);

Employee employee = new Employee("Charlie", "789 Avenue", "567890123", "charlie@mail.com",

"Room 101", 50000, new Date(1, 1, 2020));

Faculty faculty = new Faculty("Dr. Smith", "Campus Block", "432109876", "smith@university.com",

"Room 202", 70000, new Date(15, 8, 2015), "9 AM - 5 PM", "Professor");

Staff staff = new Staff("Mr. Johnson", "Admin Building", "345678901", "johnson@university.com",

"HR Office", 45000, new Date(10, 6, 2018), "HR Manager");

// Printing details using toString()

System.out.println(person);

System.out.println(student);

System.out.println(employee);

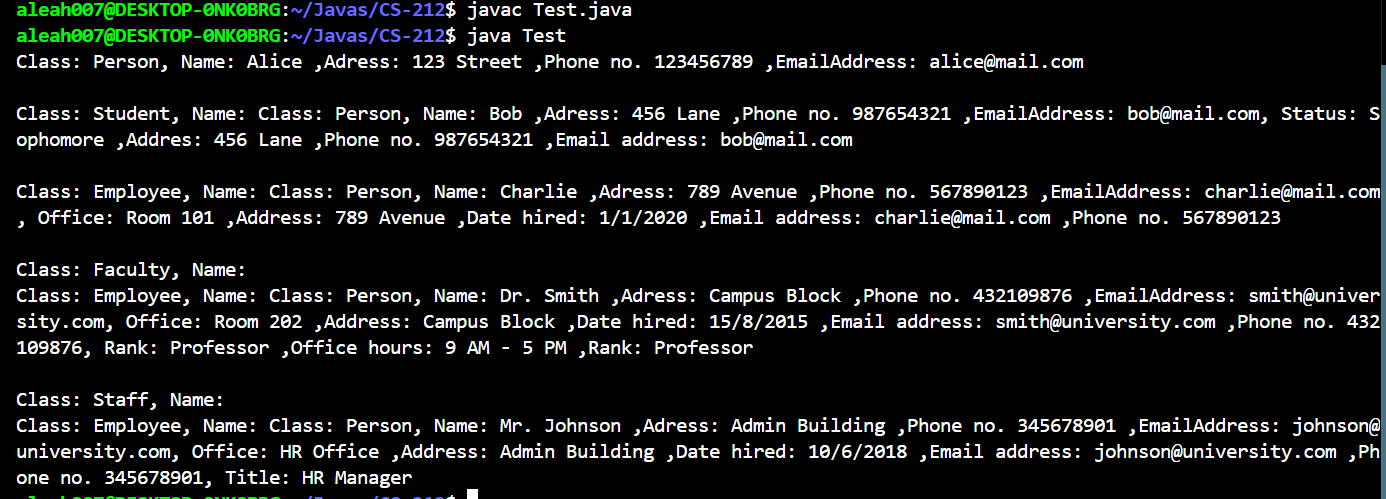
System.out.println(faculty);

System.out.println(staff);

}

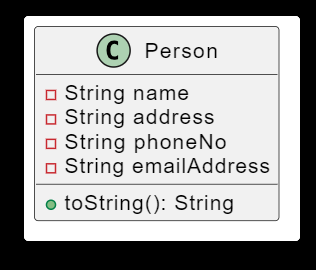
}

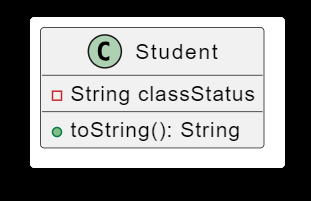
**OUTPUT:**

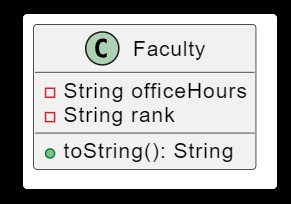
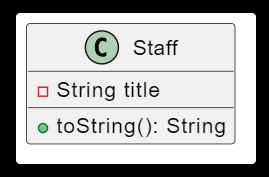


**UML DIAGRAM:**

PARENT CLASS

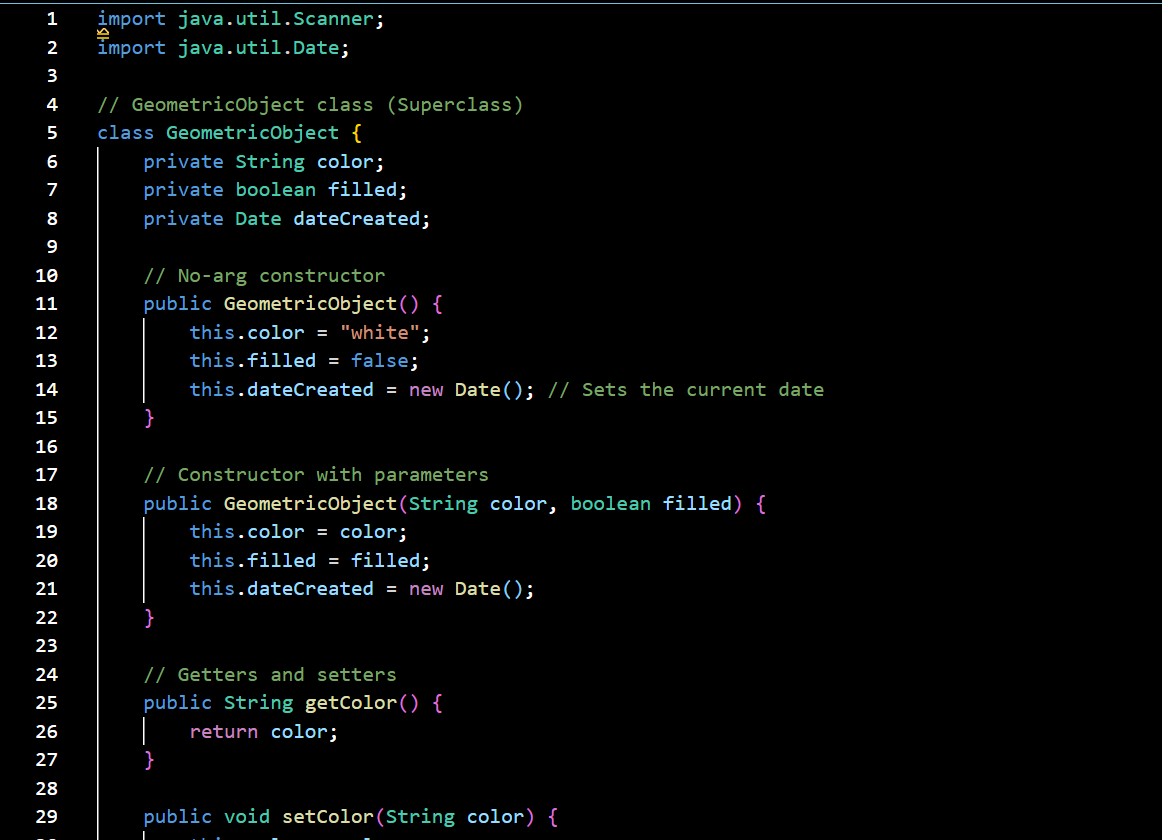


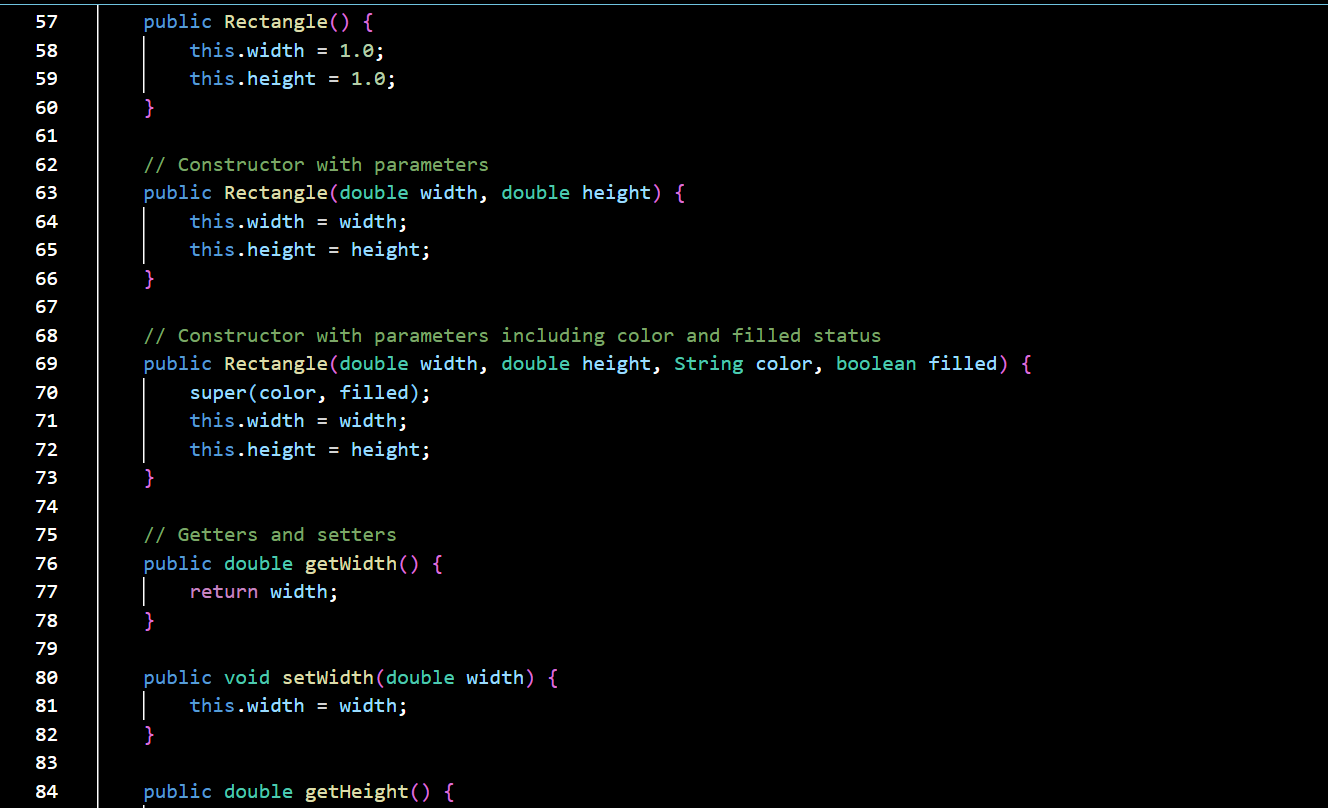
 

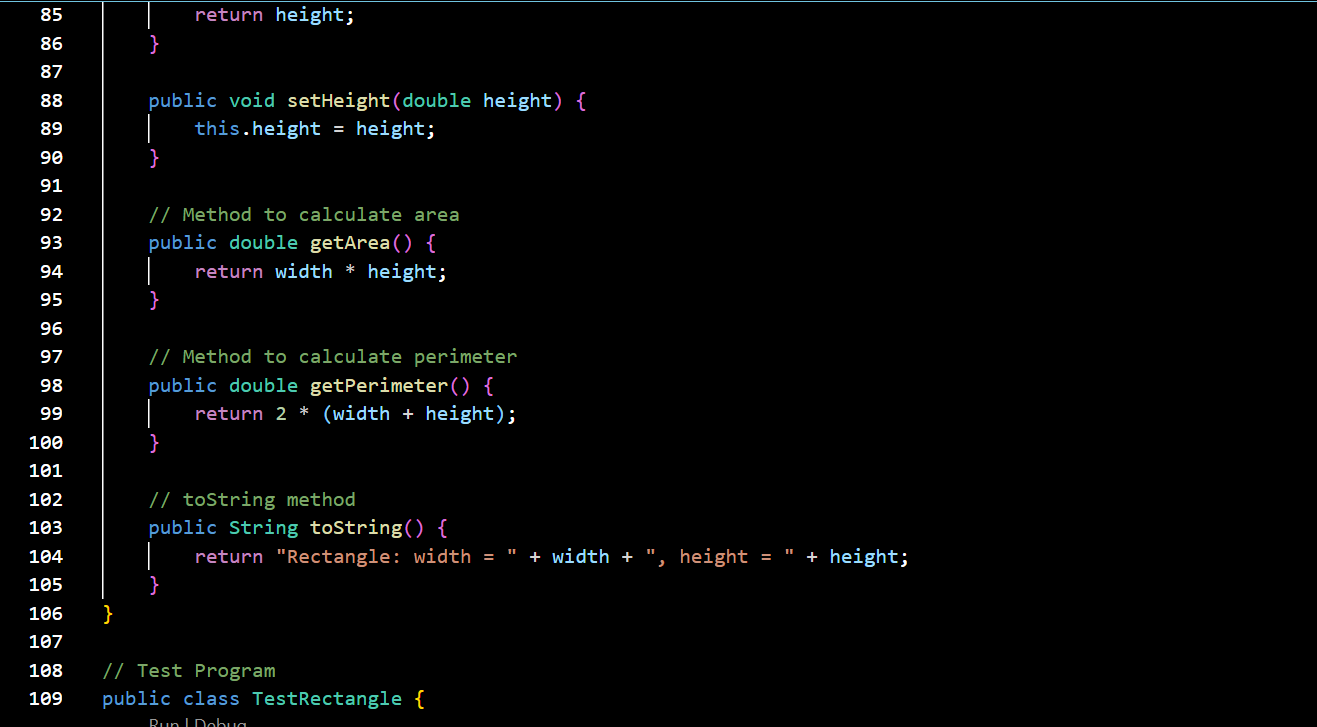
**TASK#04**

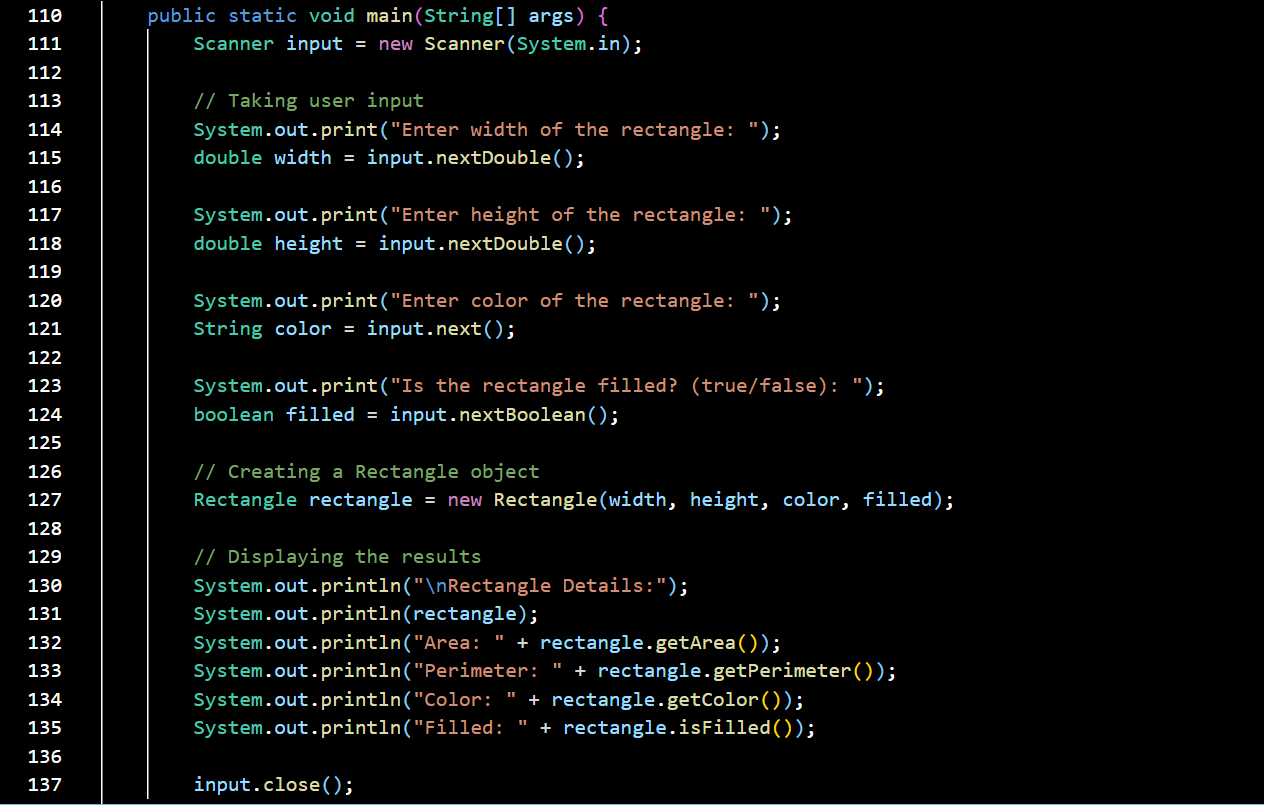
**INPUT:**

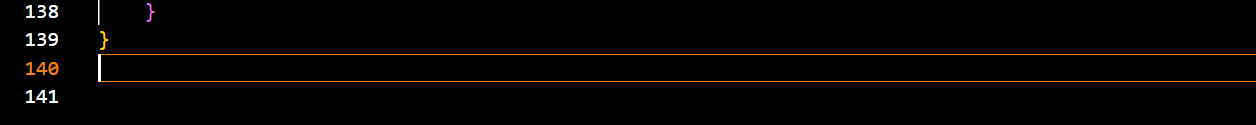












CODE:

import java.util.Scanner;

import java.util.Date;

// GeometricObject class (Superclass)

class GeometricObject {

private String color;

private boolean filled;

private Date dateCreated;

// No-arg constructor

public GeometricObject() {

this.color = "white";

this.filled = false;

this.dateCreated = new Date(); // Sets the current date

}

// Constructor with parameters

public GeometricObject(String color, boolean filled) {

this.color = color;

this.filled = filled;

this.dateCreated = new Date();

}

// Getters and setters

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 Date getDateCreated() {

return dateCreated;

}

// toString method

public String toString() {

return "GeometricObject created on " + dateCreated + "\nColor: " + color + "\nFilled: " + filled;

}

}

// Rectangle class (Subclass of GeometricObject)

class Rectangle extends GeometricObject {

private double width;

private double height;

// No-arg constructor

public Rectangle() {

this.width = 1.0;

this.height = 1.0;

}

// Constructor with parameters

public Rectangle(double width, double height) {

this.width = width;

this.height = height;

}

// Constructor with parameters including color and filled status

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

super(color, filled);

this.width = width;

this.height = height;

}

// Getters and setters

public double getWidth() {

return width;

}

public void setWidth(double width) {

this.width = width;

}

public double getHeight() {

return height;

}

public void setHeight(double height) {

this.height = height;

}

// Method to calculate area

public double getArea() {

return width \* height;

}

// Method to calculate perimeter

public double getPerimeter() {

return 2 \* (width + height);

}

// toString method

public String toString() {

return "Rectangle: width = " + width + ", height = " + height;

}

}

// Test Program

public class TestRectangle {

public static void main(String[] args) {

Scanner input = new Scanner(System.in);

// Taking user input

System.out.print("Enter width of the rectangle: ");

double width = input.nextDouble();

System.out.print("Enter height of the rectangle: ");

double height = input.nextDouble();

System.out.print("Enter color of the rectangle: ");

String color = input.next();

System.out.print("Is the rectangle filled? (true/false): ");

boolean filled = input.nextBoolean();

// Creating a Rectangle object

Rectangle rectangle = new Rectangle(width, height, color, filled);

// Displaying the results

System.out.println("\nRectangle Details:");

System.out.println(rectangle);

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

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

System.out.println("Color: " + rectangle.getColor());

System.out.println("Filled: " + rectangle.isFilled());

input.close();

}

}

**OUTPUT:**

