

LAB 3

Q.1. Create a superclass Person with attributes name and age, and a method display(). Create a subclass Student that adds an attribute studentID. Write a program to create a Student object and display all its attributes.

Program:

```
package LAB3;
class Person //creating the class with the name Person
{
    String name;//declaring the variables
    int age;

    public Person(String name, int age) //creating the arguments
    constructor
    {
        this.name = name;
        this.age = age;
    }

    public void display() //creating the display method
    {
        System.out.println("Name: " + name);
        System.out.println("Age: " + age);
    }
}

class Student extends Person //creating the another class Student
which inherit to Person class
{
    String studentID;

    public Student(String name, int age, String studentID)
    //creating the constructor
    {
        super(name, age);
        this.studentID = studentID;
    }

    public void display() //creating the display method
    {
        super.display();
        System.out.println("Student ID: " + studentID);
    }
}
```

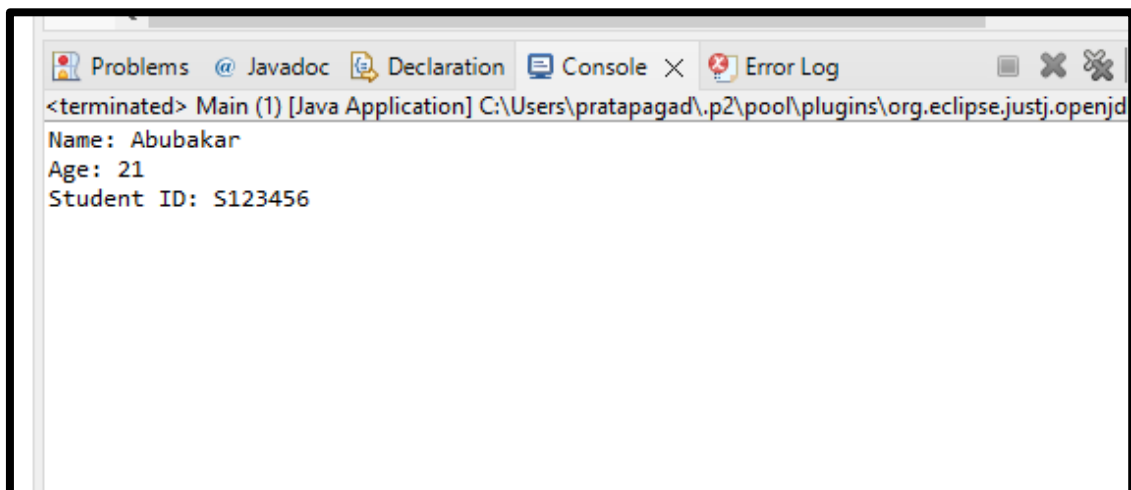
Student's ID: AF0402412

Trainer's Name: Manali Ma'm

Student's Name: Aniket Sharma

```
}  
  
public class Main //it is a main class  
{  
    public static void main(String[] args)  
    {  
        // Creating a Student object and displaying its attributes  
        Student student1 = new Student("Abubakar", 21, "S123456");  
        student1.display();//calling the methods  
    }  
}
```

Output:



Q.2. Create a superclass Calculator with a method add(int a, int b). Create a subclass AdvancedCalculator that overloads the add method to handle three integers.

Program:

```
package LAB3;
import java.util.Scanner;
class Calculator //creating the class Calculator
{
    public int add(int a, int b) //creating the parameterized method
    with 2 arguments
    {
        return a + b;
    }
}

class AdvancedCalculator extends Calculator //creating the class
AdvanceCalculator
{
    public int add(int a, int b, int c) //creating the parameterized
    method with 3 arguments
    {
        return a + b + c;
    }
}

public class Calculator_Demo //creating the main class
{
    public static void main(String[] args)
    {
        Scanner obj=new Scanner(System.in);//creating the Scanner
        object
        int n1,n2; //declaring the variables
        int m1,m2,m3;
        System.out.println("Enter the no. for 2 arguments methods: ");
        n1=obj.nextInt();//taking the numbers as an input
        n2=obj.nextInt();
        System.out.println("Enter the no. for 3 arguments methods: ");
        m1=obj.nextInt();
        m2=obj.nextInt();
        m3=obj.nextInt();
        // Using the Calculator class
        Calculator calc = new Calculator();
        System.out.println("Addition of two numbers: " +
        calc.add(n1, n2));

        // Using the AdvancedCalculator class
```

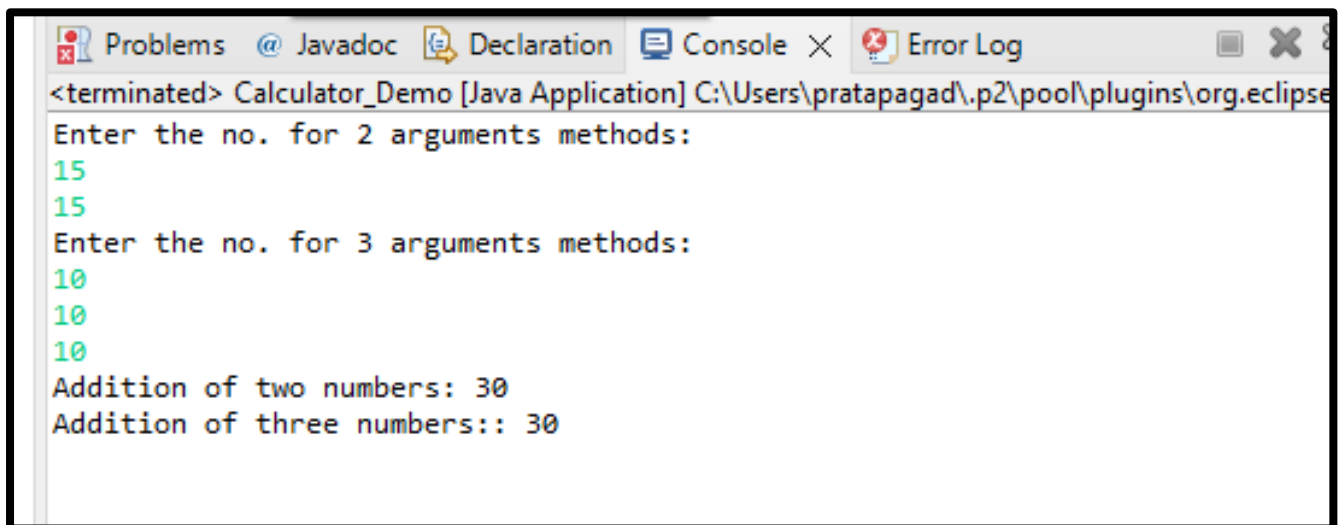
Student's ID: AF0402412

Trainer's Name: Manali Ma'm

Student's Name: Aniket Sharma

```
AdvancedCalculator advCalc = new AdvancedCalculator();
System.out.println("Addition of three numbers:: " +
advCalc.add(m1,m2,m3));
    }
}
```

Output:

A screenshot of the Eclipse IDE's Console window. The title bar shows tabs for 'Problems', 'Javadoc', 'Declaration', 'Console', and 'Error Log'. The console text shows the program's execution: it prompts for two arguments, receives '15' and '15', prompts for three arguments, receives '10', '10', and '10', and finally prints 'Addition of two numbers: 30' and 'Addition of three numbers:: 30'.

```
<terminated> Calculator_Demo [Java Application] C:\Users\pratapagad\.p2\pool\plugins\org.eclipse
Enter the no. for 2 arguments methods:
15
15
Enter the no. for 3 arguments methods:
10
10
10
Addition of two numbers: 30
Addition of three numbers:: 30
```

Q.3. Create a superclass Vehicle with a method move(). Create subclasses Car and Bike that inherit from Vehicle. Write a program to create objects of Car and Bike and call the move() method on each.

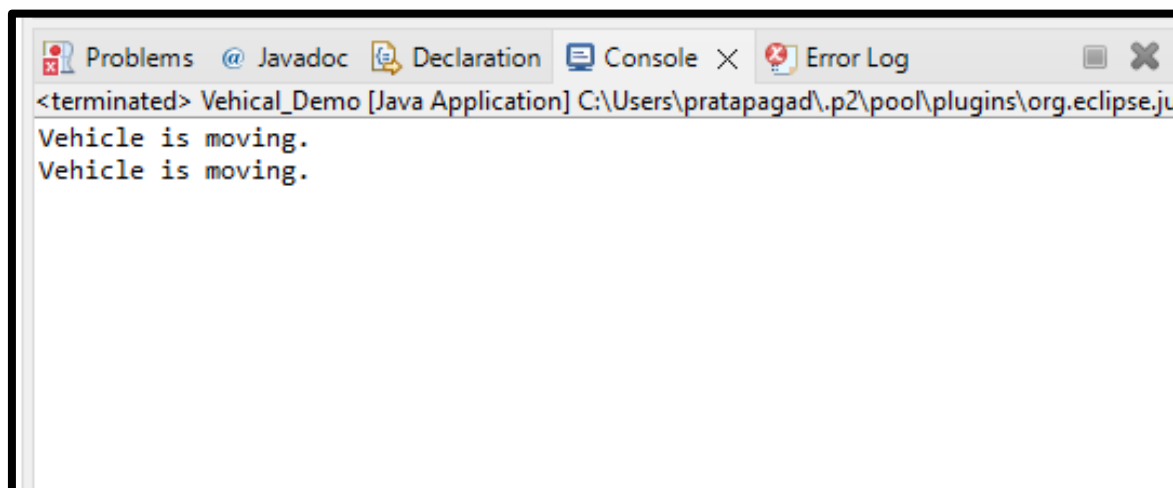
Program:

```
package LAB3;
class Vehicle //Superclass/Parent Vehicle
{
    public void move() //creating the move method
    {
        System.out.println("Vehicle is moving."); //printing the statement
    }
}
class Car extends Vehicle //Subclass Car which inherits to Parent class
{
}

class Bike extends Vehicle //Subclass Bike which inherits to Parent class
{
}

public class Vehical_Demo //Main class
{
    public static void main(String[] args)
    {
        // Creating objects of Car and Bike
        Car car = new Car();
        Bike bike = new Bike();
        // Calling move() method on Car and Bike objects
        car.move();
        bike.move();
    }
}
```

Output:

The screenshot shows the Eclipse IDE's console window. The title bar includes tabs for 'Problems', 'Javadoc', 'Declaration', 'Console', and 'Error Log'. The console output shows the execution of the 'Vehical_Demo' Java application. The first line of output is '<terminated> Vehical_Demo [Java Application] C:\Users\pratapagad\.p2\pool\plugins\org.eclipse.ju'. Below this, the program's output is displayed: 'Vehicle is moving.' followed by 'Vehicle is moving.' on the next line. The text is in a monospaced font, typical of code editors.

Q.4. Create an class Employee with an abstract method calculatePay(). Create subclasses SalariedEmployee and HourlyEmployee that implement the calculatePay() method. Write a program to create objects of both subclasses and call the calculatePay() method.

Program:

```
package LAB3;
abstract class Employee //Abstract superclass Employee
{
    public abstract double calculatePay();// Abstract method
    calculatePay
}
class SalariedEmployee extends Employee //Subclass SalariedEmployee which
inherits to Employee class
{
    private double salary;//declaring the variable
    public SalariedEmployee(double salary) // creating the Constructor
    {
        this.salary = salary;
    }
    // Implementation of calculatePay method
    @Override
    public double calculatePay()
    {
        // For salaried employee, pay is the salary
        return salary;
    }
}
//Subclass HourlyEmployee
class HourlyEmployee extends Employee //Subclass SalariedEmployee which
inherits to Employee class
{
    private double hourlyRate;
    private int hoursWorked;

    // Constructor
    public HourlyEmployee(double hourlyRate, int hoursWorked)
    {
        this.hourlyRate = hourlyRate;
        this.hoursWorked = hoursWorked;
    }

    // Implementation of calculatePay method
    @Override
    public double calculatePay()
    {
        // For hourly employee, pay is calculated by multiplying
        hourly rate with hours worked
        return hourlyRate * hoursWorked;
    }
}
```

Student's ID: AF0402412

Trainer's Name: Manali Ma'm

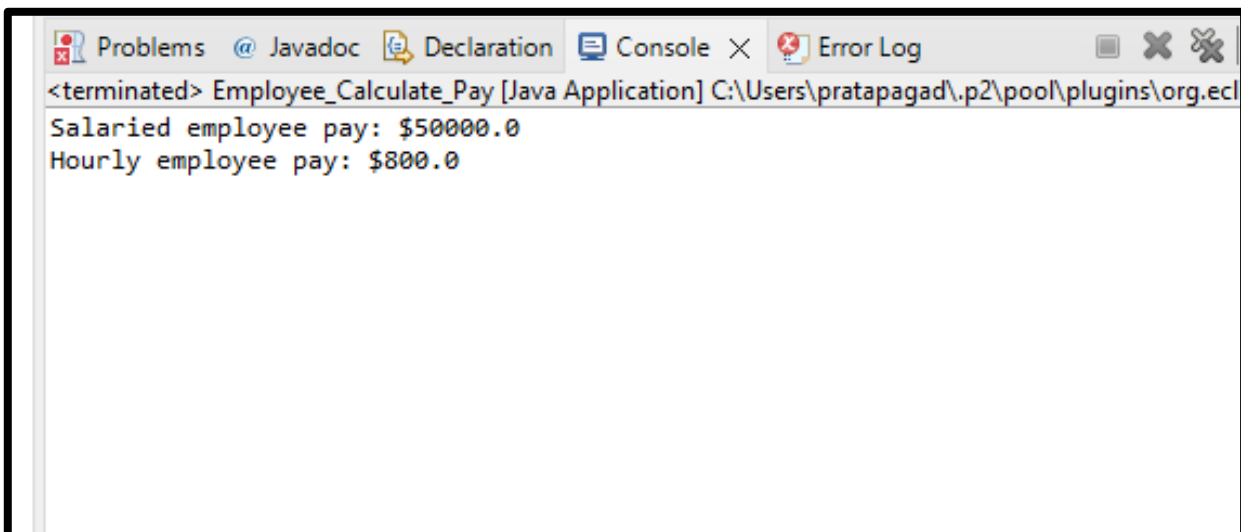
Student's Name: Aniket Sharma

```
}

//Main class
public class Employee_Calculate_Pay
{
    public static void main(String[] args)
    {
        // Creating objects of SalariedEmployee and HourlyEmployee
        SalariedEmployee salariedEmployee = new
SalariedEmployee(50000); // Salary of $50,000 per year
        HourlyEmployee hourlyEmployee = new HourlyEmployee(20, 40); //
Hourly rate of $20, worked 40 hours

        // Calling calculatePay() method on both objects and printing
the result
        System.out.println("Salaried employee pay: $" +
salariedEmployee.calculatePay());
        System.out.println("Hourly employee pay: $" +
hourlyEmployee.calculatePay());
    }
}
```

Output:

A screenshot of an IDE's console window. The title bar shows tabs for 'Problems', '@ Javadoc', 'Declaration', 'Console', and 'Error Log'. The console output shows the program has terminated and displays the calculated pay for two employee types: 'Salaried employee pay: \$50000.0' and 'Hourly employee pay: \$800.0'.

```
<terminated> Employee_Calculate_Pay [Java Application] C:\Users\pratapagad\.p2\pool\plugins\org.ec
Salaried employee pay: $50000.0
Hourly employee pay: $800.0
```

Q.5. Create an class Document with an method void open(). Implement subclasses WordDocument, PDFDocument, and SpreadsheetDocument that extend Document and provide implementations for open(). Write a main class to demonstrate opening different types of documents.(implement compile time- polymorphism).

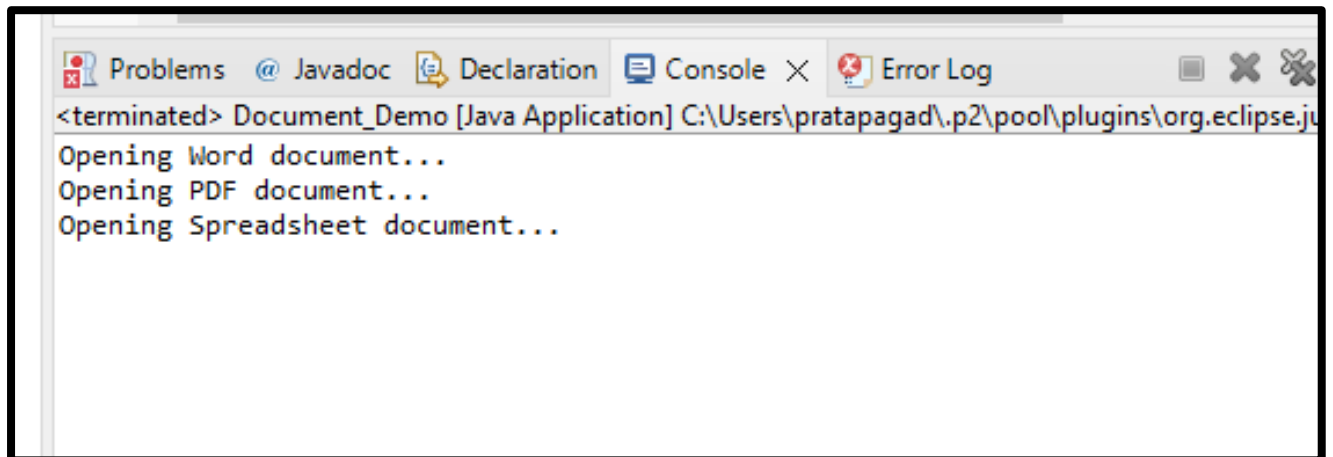
Program:

```
package LAB3;
class Document //Superclass Document
{
    // Method to open the document
    public void open()
    {
        System.out.println("Opening the document...");
    }
}
class WordDocument extends Document //Subclass WordDocument which inherits
to Document class
{
    // Overriding the open() method for Word documents
    @Override
    public void open()
    {
        System.out.println("Opening Word document...");
    }
}
class PDFDocument extends Document //Subclass PDFDocument which inherits
to Document class
{
    // Overriding the open() method for PDF documents
    @Override
    public void open()
    {
        System.out.println("Opening PDF document...");
    }
}
class SpreadsheetDocument extends Document //Subclass SpreadsheetDocument
which inherits to Document class
{
    // Overriding the open() method for Spreadsheet documents
    @Override
    public void open()
    {
        System.out.println("Opening Spreadsheet document...");
    }
}
public class Document_Demo //Main class to demonstrate compile-time
polymorphism
{
```



```
public static void main(String[] args)
{
    // Creating objects of different document types
    Document doc1 = new WordDocument();
    Document doc2 = new PDFDocument();
    Document doc3 = new SpreadsheetDocument();

    // Demonstrating compile-time polymorphism by calling the
    open() method on each object
    doc1.open(); // Calls open() method of WordDocument
    doc2.open(); // Calls open() method of PDFDocument
    doc3.open(); // Calls open() method of SpreadsheetDocument
}
}
```

Output:

Q.6. Create a class Calculator with overloaded methods add() that take different numbers and types of parameters: int add(int a, int b) double add(double a, double b) int add(int a, int b, int c) Write a main class to demonstrate the usage of these methods.

Program:

```
package LAB3;
public class Calculator_Overload //creating the class
"Calculator_Overload"
{
    public int add(int a, int b) //declaring the 2 argument method
    {
        return a + b;
    }

    public double add(double a, double b) //declaring the 2 argument
method
    {
        return a + b;
    }

    public int add(int a, int b, int c) //declaring the 3 argument method
    {
        return a + b + c;
    }

    // You can add more overloaded add() methods for other types if needed

    public static void main(String[] args)
    {
        Calculator_Overload calculator = new
Calculator_Overload();//creating the class object

        // Using the overloaded add() methods
        int sum1 = calculator.add(5, 10);//declaring the variables
        double sum2 = calculator.add(3.5, 4.5);
        int sum3 = calculator.add(2, 3, 4);

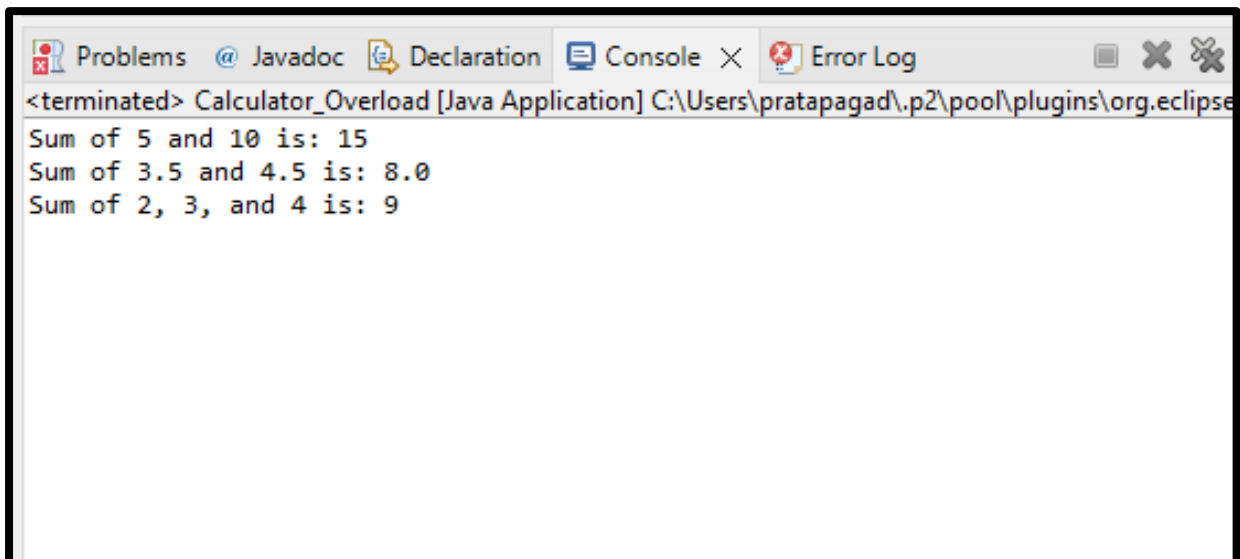
        System.out.println("Sum of 5 and 10 is: " + sum1);//printing the
statement
        System.out.println("Sum of 3.5 and 4.5 is: " + sum2);
        System.out.println("Sum of 2, 3, and 4 is: " + sum3);
    }
}
```

Student's ID: AF0402412

Trainer's Name: Manali Ma'm

Student's Name: Aniket Sharma

Output:



The screenshot shows the Eclipse IDE's Console window. The title bar includes tabs for 'Problems', 'Javadoc', 'Declaration', 'Console', and 'Error Log'. The console text shows the program has terminated and displays three lines of output: 'Sum of 5 and 10 is: 15', 'Sum of 3.5 and 4.5 is: 8.0', and 'Sum of 2, 3, and 4 is: 9'.

```
<terminated> Calculator_Overload [Java Application] C:\Users\pratapagad\.p2\pool\plugins\org.eclipse
Sum of 5 and 10 is: 15
Sum of 3.5 and 4.5 is: 8.0
Sum of 2, 3, and 4 is: 9
```

Q.7. Create a JavaBean class Person with properties firstName, lastName, age, and email. Implement the required no-argument constructor, getter and setter methods for each property. Write a main class to create an instance of Person, set its properties, and print them out.

Program:

```
package demo;

public class Person {
    private String firstName;
    private String lastName;
    private int age;
    private String email;

    public Person() {
        // Required no-argument constructor
    }

    // Getter and setter methods for firstName
    public String getFirstName() {
        return firstName;
    }

    public void setFirstName(String firstName) {
        this.firstName = firstName;
    }

    // Getter and setter methods for lastName
    public String getLastName() {
        return lastName;
    }

    public void setLastName(String lastName) {
        this.lastName = lastName;
    }

    // Getter and setter methods for age
    public int getAge() {
        return age;
    }

    public void setAge(int age) {
        this.age = age;
    }

    // Getter and setter methods for email
    public String getEmail() {
        return email;
    }
}
```

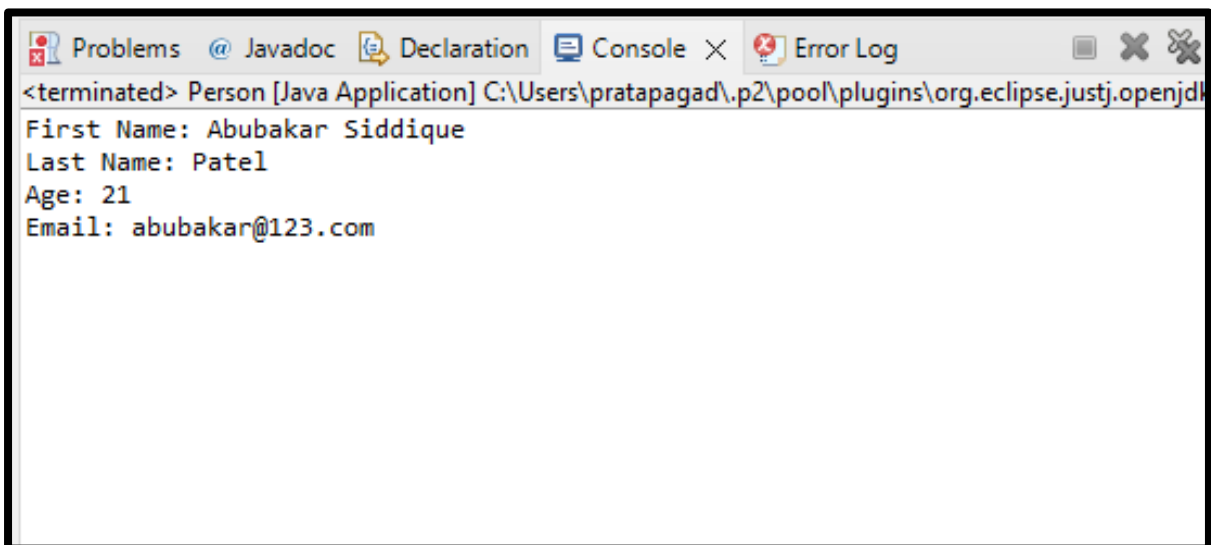
```
public void setEmail(String email) {
    this.email = email;
}

public static void main(String[] args) {
    // Create an instance of Person
    Person person = new Person();

    // Set properties
    person.setFirstName("Abubakar Siddique");
    person.setLastName("Patel");
    person.setAge(21);
    person.setEmail("abubakar@123.com");

    // Print out the properties
    System.out.println("First Name: " + person.getFirstName());
    System.out.println("Last Name: " + person.getLastName());
    System.out.println("Age: " + person.getAge());
    System.out.println("Email: " + person.getEmail());
}
}
```

Output:

A screenshot of the Eclipse IDE's console window. The window has a title bar with icons for Problems, Javadoc, Declaration, Console, and Error Log. The console text shows the output of a Java application: "<terminated> Person [Java Application] C:\Users\pratapagad\.p2\pool\plugins\org.eclipse.justj.openjdk...". Below this, the program's output is displayed on four lines: "First Name: Abubakar Siddique", "Last Name: Patel", "Age: 21", and "Email: abubakar@123.com".

```
<terminated> Person [Java Application] C:\Users\pratapagad\.p2\pool\plugins\org.eclipse.justj.openjdk
First Name: Abubakar Siddique
Last Name: Patel
Age: 21
Email: abubakar@123.com
```

Q.8. Create a JavaBean class Car with properties make, model, year, and color. Implement the required no-argument constructor, getter and setter methods for each property. Write a main class to create an instance of Car, set its properties, and print the car details.

Program:

```
package LAB3;
public class Car_demo {
    private String make;
    private String model;
    private int year;
    private String color;

    public Car() {
        // Required no-argument constructor
    }

    // Getter and setter methods for make
    public String getMake() {
        return make;
    }

    public void setMake(String make) {
        this.make = make;
    }

    // Getter and setter methods for model
    public String getModel() {
        return model;
    }

    public void setModel(String model) {
        this.model = model;
    }

    // Getter and setter methods for year
    public int getYear() {
        return year;
    }

    public void setYear(int year) {
        this.year = year;
    }

    // Getter and setter methods for color
    public String getColor() {
        return color;
    }

    public void setColor(String color) {
```

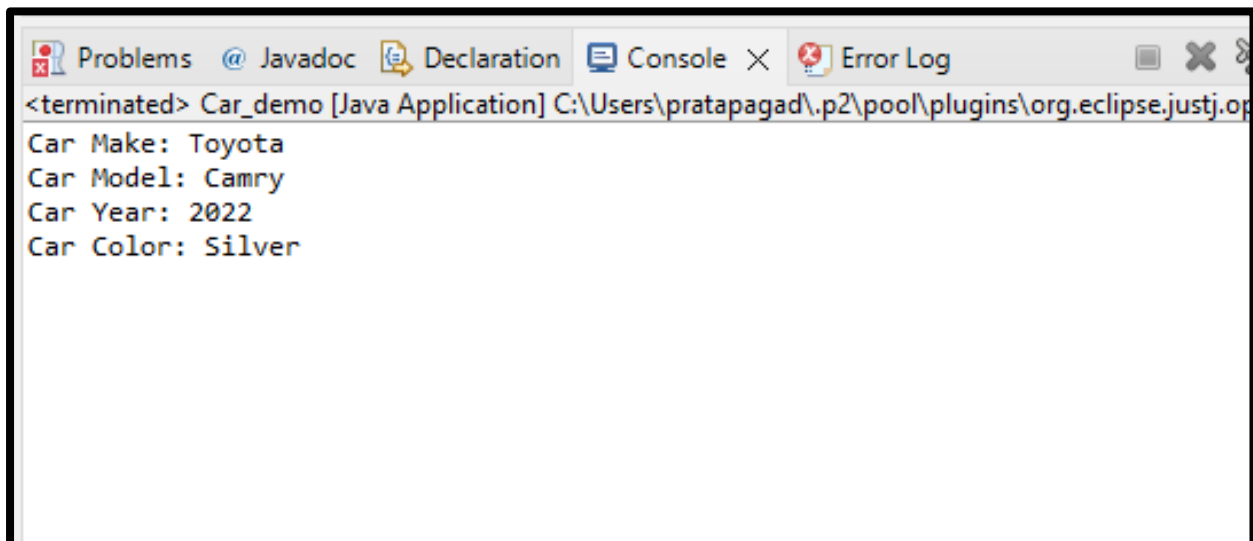
```
        this.color = color;
    }

    public static void main(String[] args) {
        // Create an instance of Car
        Car_demo car = new Car_demo();

        // Set properties
        car.setMake("Toyota");
        car.setModel("Camry");
        car.setYear(2022);
        car.setColor("Silver");

        // Print out the car details
        System.out.println("Car Make: " + car.getMake());
        System.out.println("Car Model: " + car.getModel());
        System.out.println("Car Year: " + car.getYear());
        System.out.println("Car Color: " + car.getColor());
    }
}
```

Output:

A screenshot of the Eclipse IDE's Console window. The window has a title bar with icons for Problems, Javadoc, Declaration, Console, and Error Log. The Console tab is active, showing the output of a Java application. The output text is: <terminated> Car_demo [Java Application] C:\Users\pratapagad\.p2\pool\plugins\org.eclipse.justj.op, followed by four lines of car details: Car Make: Toyota, Car Model: Camry, Car Year: 2022, and Car Color: Silver.

```
<terminated> Car_demo [Java Application] C:\Users\pratapagad\.p2\pool\plugins\org.eclipse.justj.op
Car Make: Toyota
Car Model: Camry
Car Year: 2022
Car Color: Silver
```