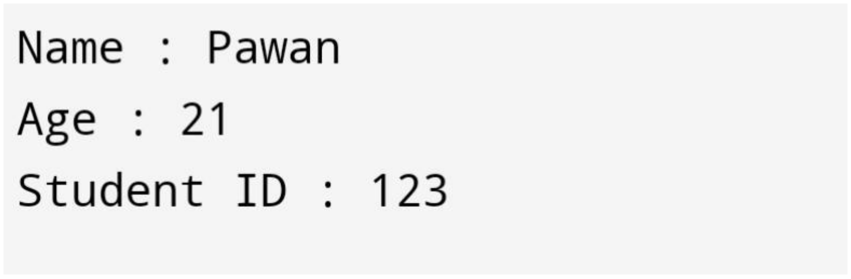
1. Create a superclass Person with attributes name and age, and a method display(). Create a subclass Student that adds an attribute studentID. Write a program to create a Student object and display all its attributes.

Code:-

```
package MyPackage;
class Person { //parent class
  String name;
  Int age;

  Public void display() { //display method prints name and age
    System.out.println("Name : " + name);
    System.out.println("Age : " + age);
  }
}

Class Student extends Person { //child class
  Int studentID;

  Public void display() { //display method override parent class display
method
    Super.display();
    System.out.println("Student ID : " + studentID);
  }
}

Public class PersonDemo
{
  Public static void main (String[] args)
  {
    Student s=new Student(); //creating the object of student class
    s.name = "Pawan";
    s.age = 21;
    s.studentID = 123;

    s.display(); //calling the display method
  }
}
```

Output:-



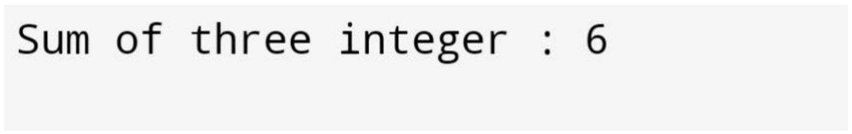
```
Name : Pawan
Age : 21
Student ID : 123
```

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.

Code:-

```
package MyPackage;
//super class
Class Calculator {
  Public int add(int a, int b) {
    Return a + b;
  }
}

//sub-class extends super class
Class AdvancedCalculator extends Calculator {
  //override the add method of super class
  Public int add(int a, int b, int c) {
    Return a + b + c;
  }
}

Public class CalculatorDemo
{
  Public static void main (String[] args)
  { //creating the object of subclass
    AdvancedCalculator advCalc = new AdvancedCalculator();

    //printing the sum of three integers using add method of subclass
    System.out.println("Sum of three integer : " + advCalc.add(1, 2, 3));
  }
}
```

Output:-



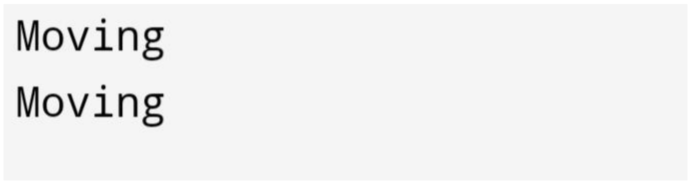
```
Sum of three integer : 6
```

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.

Code:-

```
package MyPackage;
//super class
Class Vehicle {
  Public void move() {
    System.out.println("Moving");
  }
}

//sub class extends super class
Class Car extends Vehicle {
}

//sub class extends super class
Class Bike extends Vehicle {

}

Public class VehicleDemo
{
  Public static void main (String[] args)
  { //creating the object of subclass car
    Car car = new Car();

    //creating the object of subclass car
    Bike bike = new Bike();

    Car.move(); //calling the move method of class car
    Bike.move(); //calling the move method of class bike
  }
}
```

Output:-



```
Moving
Moving
```

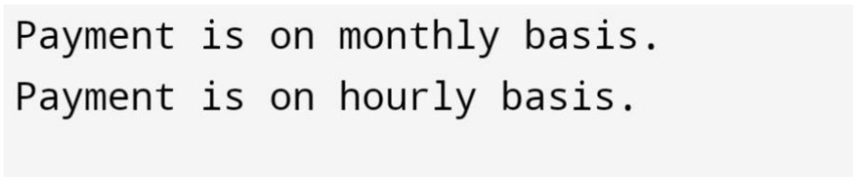
4.Create an class Employee with an abstract method calculatePay(). Create subclasses SalariedEmployee and HourlyEmployee that implement the calculatePay() method. Write a program to create objects of both subclasses and call the calculatePay() method.

Code:-

```
package MyPackage;
//abstract class
Abstract class Employee {
  Abstract void calculatePay();
}

//subclass extends abstract class
Class SalariedEmployee extends Employee {
  Void calculatePay() {
    System.out.println("Payment is on monthly basis.");
  }
}

//subclass extends abstract class
Class HourlyEmployee extends Employee {
  Void calculatePay() {
    System.out.println("Payment is on hourly basis.");
  }
}
```

```
Public class EmployeeDemo
{
  Public static void main(String[] arg)
  { //creating the object of subclasses
    SalariedEmployee se = new SalariedEmployee();
    HourlyEmployee he = new HourlyEmployee();

    Se.calculatePay(); //calling the calculatePay method of 1st subclass
    He.calculatePay(); //calling the calculatePay method of 2nd subclass
  }
}
```

Output:-



```
Payment is on monthly basis.
Payment is on hourly basis.
```

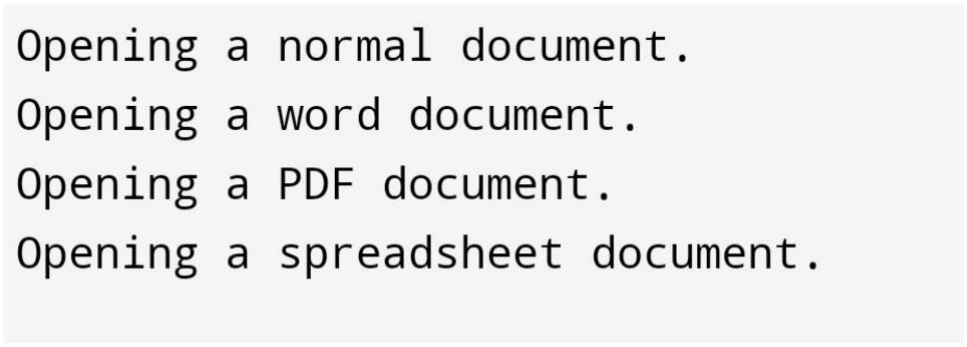
5.Create an class Document with an method void open(). Implement subclasses WordDocument, PDFDocument, and SpreadsheetDocument that extend Document and provide implementations for open(). Write a main class to demonstrate opening different types of documents.(implement complile time- polymorphism).

Code:-

```
package MyPackage;
//super class
Class Document
{
  Public void open() {
    System.out.println("Opening a normal document.");
  }
}

//subclass extends super class
Class WordDocument extends Document
{
  Public void open() {
    System.out.println("Opening a word document.");
  }
}

//subclass extends super class
Class PDFDocument extends Document
{
  Public void open() {
    System.out.println("Opening a PDF document.");
  }
}

//subclass extends super class
```

```
Class SpreadSheetDocument extends Document
{
  Public void open() {
    System.out.println("Opening a spreadsheet document.");
  }
}

Public class DocumentDemo
{
  Public static void main(String[] args)
  { // creating the object of super class
    Document d = new Document();

    //creating the object of subclasses
    WordDocument wd = new WordDocument();
    PDFDocument pd = new PDFDocument();
    SpreadSheetDocument sd = new SpreadSheetDocument();

    //calling the open method of super class
    d.open();

    //calling the open method of subclasses
    Wd.open();
    Pd.open();
    Sd.open();
  }
}
```

Output:-



```
Opening a normal document.
Opening a word document.
Opening a PDF document.
Opening a spreadsheet document.
```

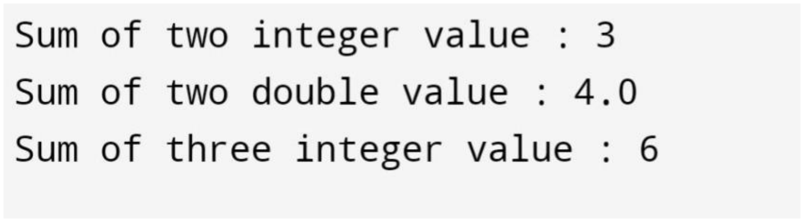
6.Create a class Calculator with overloaded methods add() that take different numbers and types of parameters: int add(int a, int b), double add(double a, double b), int add(int a, int b, int c) Write a main class to demonstrate the usage of these methods.

Code:-

```
package MyPackage;
//creating a class calculator
Class Calculator
{
  //creating a n add method and overloading add method with different
numbers and types of parameters
```

```
  Public int add(int a, int b) {
    Return a + b;
  }

  Public double add(double a, double b) {
    Return a + b;
  }

  Public int add(int a, int b, int c) {
    Return a + b + c;
  }
}

Public class CalculatorDemo
{
  Public static void main(String[] args)
  {
    //creating the object of calculator class
    Calculator c = new Calculator();

    System.out.println("Sum of two integer value : " + c.add(1, 2));
    System.out.println("Sum of two double value : " + c.add(1.7, 2.3));
    System.out.println("Sum of three integer value : " + c.add(1, 2, 3));
  }
}
```

Output:-



```
Sum of two integer value : 3
Sum of two double value : 4.0
Sum of three integer value : 6
```

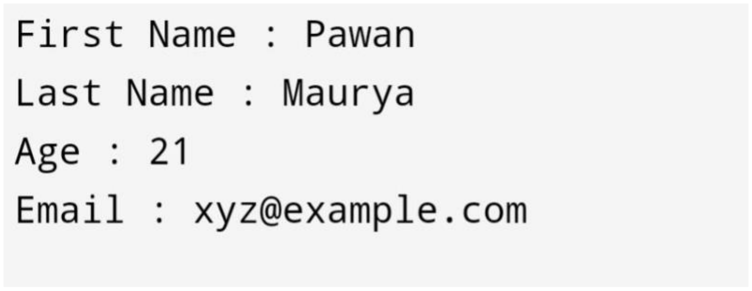
7.Create a JavaBean class Person with properties firstName, lastName, age, and email. Implement the required no-argument constructor, getter and setter methods for each property. Write a main class to create an instance of Person, set its properties, and print them out.

Code:-

```
package MyPackage;
//creating a javabean class person which implements Serializable
Class Person implements java.io.Serializable
{
  String firstName;
  String lastName;
  Int age;
  String email;

  //no-argument constructor
  Public Person() {

  }

  //getter method for firstName
  Public String getFirstName() {
    Return firstName;
  }

  //setter method for firstName
  Public void setFirstName(String firstName) {
    This.firstName=firstName;
  }

  //getter method for lastName
  Public String getLastName() {
    Return lastName;
  }

  //setter method for lastName
  Public void setLastName(String lastName) {
    This.lastName=lastName;
  }

  //getter method for age
  Public int getAge() {
    Return age;
  }

  //setter method for age
  Public void setAge(int age) {
    This.age=age;
  }

  //getter method for email
  Public String getEmail() {
    Return email;
  }

  //setter method for email
  Public void setEmail(String email) {
    This.email=email;
  }
}

Public class PersonDemo
{
  Public static void main(String[] args)
  {
    //creating the object of person class
    Person p = new Person();

    //setting the properties of person class
    p.setFirstName("Pawan");
    p.setLastName("Maurya");
    p.setAge(21);
    p.setEmail(xyz@example.com);

    //getting the properties of person class
```

```
        System.out.println(“First Name : “ + p.getFirstName());
        System.out.println(“Last Name : “ + p.getLastName());
        System.out.println(“Age : “ + p.getAge());
        System.out.println(“Email : “ + p.getEmail());
      }
    }
```

Output:-



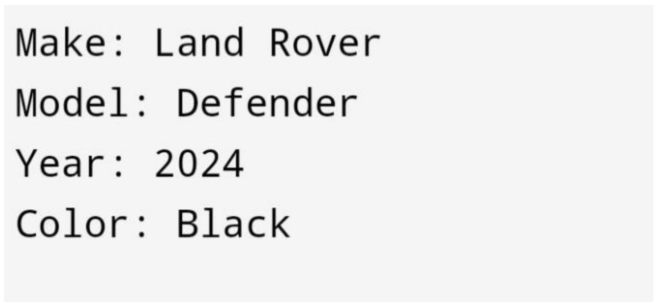
```
First Name : Pawan
Last Name : Maurya
Age : 21
Email : xyz@example.com
```

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.

Code:-

```
package MyPackage;
//creating a javabean class car which implements Serializable
Class Car implements java.io.Serializable
{
  String make;
  String model;
  Int year;
  String color;

  Public Car() {
    // No-argument constructor
  }

  //getter method for make
  Public String getMake() {
    Return make;
  }

  //setter method for make
  Public void setMake(String make) {
    This.make = make;
  }

  //getter method for model
  Public String getModel() {
    Return model;
  }

  //setter method for model
  Public void setModel(String model) {
    This.model = model;
  }

  //getter method for year
  Public int getYear() {
    Return year;
  }

  //setter method for year
  Public void setYear(int year) {
    This.year = year;
  }

  //getter method for color
  Public String getColor() {
    Return color;
  }

  //setter method for color
  Public void setColor(String color) {
    This.color = color;
  }
}

Public class CarDemo
{
  Public static void main(String[] args)
  {
    // Create an object of class Car
    Car c = new Car();

    // Set the properties of the car
    c.setMake(“Land Rover”);
    c.setModel(“Defender”);
    c.setYear(2024);
    c.setColor(“Black”);

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

Output:-



```
Make: Land Rover
Model: Defender
Year: 2024
Color: Black
```