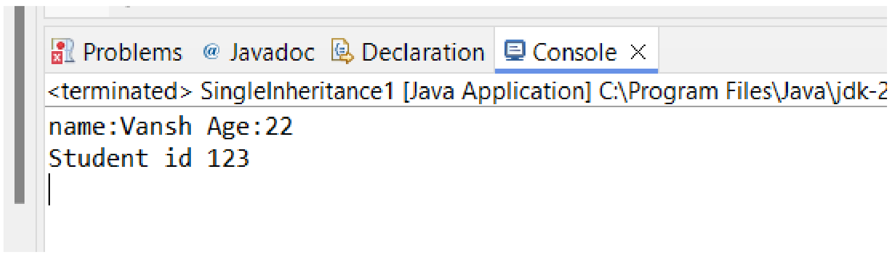
LAB_3

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 Assignment;

class Person     //parent class
{
	String name="Vansh";
	int age=22;
	public void superclass()  //method
	{
		System.out.println("name:" + name+" " +"Age:" + age);
	}

}
class Student extends Person   //student class
{
	int studentId=123;
	public void childclass()  //method
	{
		System.out.println("Student id "+ studentId);
	}
}
public class SingleInheritance1 //main class
{

	public static void main(String args[])  //main method
	{
		Student id=new Student(); //object declaration
		id.superclass();
		id.childclass();
	}

}
```

OUTPUT:-



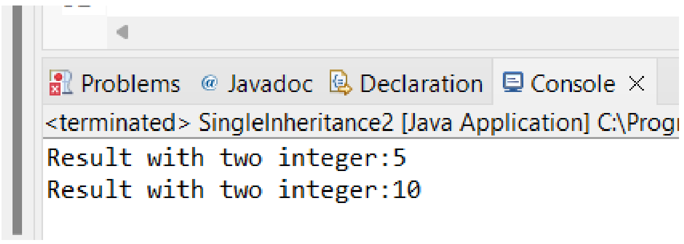
```
<terminated> SingleInheritance1 [Java Application] C:\Program Files\Java\jdk-2
name:Vansh Age:22
Student id 123
```

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 Assignment;
class Calculator
{
	public int add(int a,int b)
	{
		return a+b;
	}
}
class AdvancedCalculator extends Calculator
{
	@Override
	public int add(int a,int b)
	{
		return a+b;
	}
	public int add(int a,int b,int c) {
		return a+b+c;
	}
}
public class SingleInheritance2 {

	public static void main(String[] args) {
		int a=2;
		int b=3;
		int c=5;
		AdvancedCalculator av=new AdvancedCalculator();
		System.out.println("Result with two integer:"+ av.add(a, b));
		System.out.println("Result with two integer:"+ av.add(a, b,c));
	}

}
```

OUTPUT:-



```
<terminated> SingleInheritance2 [Java Application] C:\Progı
Result with two integer:5
Result with two integer:10
```

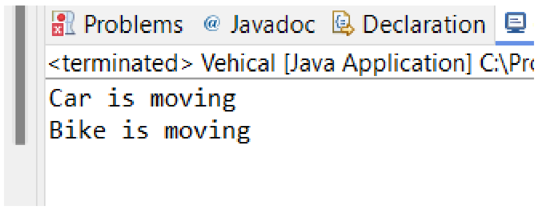
---

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 lab_3;

class Vehicle {  //superclass vehical
	public void move() {
        System.out.println("Vehicle is moving");
    }
}
class Car extends Vehicle { //Subclass Bike extends Vehicle
    public void move() {
        System.out.println("Car is moving");
    }
}
class Bike extends Vehicle { //Subclass Bike extends Vehicle
    public void move() {
        System.out.println("Bike is moving");
    }
}
public class Vehical{
    public static void main(String[] args) {

        Vehicle car = new Car();
        Vehicle bike = new Bike();
        car.move();
        bike.move();
    }
}
```

OUTPUT:-



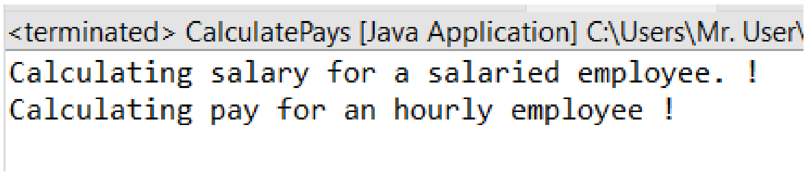
```
<terminated> Vehical [Java Application] C:\Pr
Car is moving
Bike is moving
```

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 EDemo;
//Abstract superclass Employee
abstract class Employees {
    public abstract void calculatePay(); // Abstract method far calculate and pay
}
class SalariedEmployee extends Employees {
    public void calculatePay() {
        System.out.println("Calculating salary for a salaried employee. !");
    }
}
//Subclass HourlyEmployee
class HourlyEmployee extends Employees {
    public void calculatePay() {
        System.out.println("Calculating pay for an hourly employee !");
    }
}
public class CalculatePays {
    public static void main(String[] args) {
        Employees salariedEmp = new SalariedEmployee();
        Employees hourlyEmp = new HourlyEmployee();
        salariedEmp.calculatePay(); //calling methods
        hourlyEmp.calculatePay();
    }
}
```

OUTPUT:-



```
<terminated> CalculatePays [Java Application] C:\Users\Mr. User\
Calculating salary for a salaried employee. !
Calculating pay for an hourly employee !
```

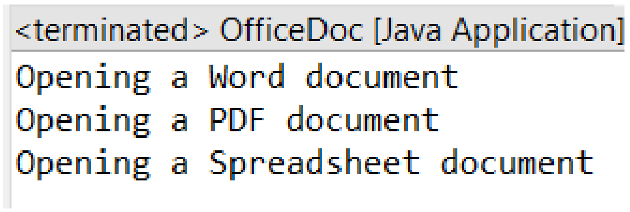
---

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 Hellow;
class Document {
        // Method to open document (to be overridden by subclasses)
        public void open() {
                System.out.println("Opening a generic document");
        }
}
//Sub claases
class WordDocument extends Document {
        public void open() {
                System.out.println("Opening a Word document");
        }
}
class PDFDocument extends Document {
        public void open() {
                System.out.println("Opening a PDF document");
        }
}
class SpreadsheetDocument extends Document {
        public void open() {
                System.out.println("Opening a Spreadsheet document");
        }
}
public class OfficeDoc {
        public static void main(String[] args) {
                Document doc1 = new WordDocument();
                Document doc2 = new PDFDocument();
                Document doc3 = new SpreadsheetDocument();
                //calling the method from classes
                doc1.open();
                doc2.open();
                doc3.open();
        }
}
```

OUTPUT:-



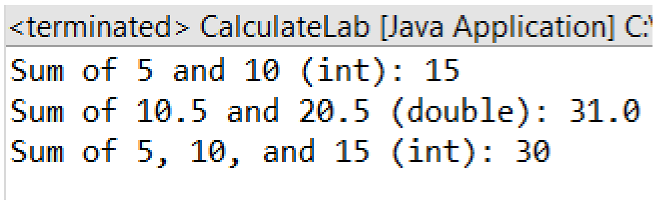
```
<terminated> OfficeDoc [Java Application]
Opening a Word document
Opening a PDF document
Opening a Spreadsheet document
```

---

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 Hellow;

//creating a Class with overloaded add methods
class Calculat {
        //Method to add two integers
        public int add(int a, int b) {
                return a + b;
        }
        //Method for add two doubles
        public double add(double a, double b) {
                return a + b;
        }
        //Method for add three integers
        public int add(int a, int b, int c) {
                return a + b + c;
        }
}
public class CalculateLab {
        public static void main(String[] args) {
                Calculat calc = new Calculat();
                //Demonstrate adding two integers
                int sum1 = calc.add(5, 10);
                System.out.println("Sum of 5 and 10 (int): " + sum1);
                double sum2 = calc.add(10.5, 20.5);
                System.out.println("Sum of 10.5 and 20.5 (double): " + sum2);
                int sum3 = calc.add(5,10,15);
                System.out.println("Sum of 5, 10, and 15 (int): " + sum3);
        }
}
```

OUTPUT:-


```
<terminated> CalculateLab [Java Application] C:'
Sum of 5 and 10 (int): 15
Sum of 10.5 and 20.5 (double): 31.0
Sum of 5, 10, and 15 (int): 30
```

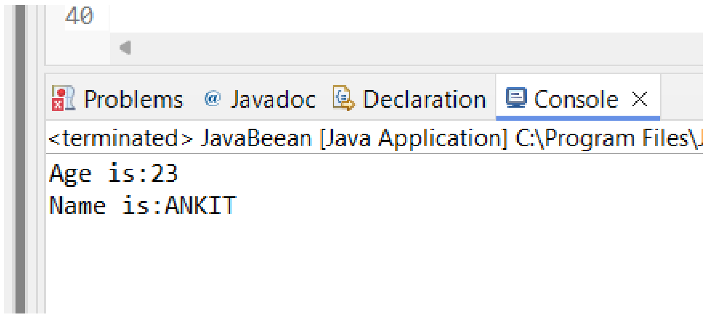
7. Create a JavaBean class Person with properties firstName, lastName, age, and email. Implement the required no-argument constructor, getter and setter methods for each property. Write a main class to create an instance of Person, set its properties, and print them out.

CODE:-

```
package WorksOfClass;
class demo implements java.io.Serializable
{
	private int age;
	private String name;

	public demo() //no argument
	{

	}
	public int getAge() //getter method
	{
		return age;
	}
	public void setAge(int age)
	{
		this.age=age;
	}
	public String getName() //getter method
	{
		return name;
	}
	public void setName( String studname)
	{
		this.name=studname;
	}
}
public class JavaBeean {

	public static void main(String[] args) {
		demo jd=new demo();
		jd.setAge(23);
		System.out.println("Age is:"+jd.getAge());

		jd.setName("ANKIT");
		System.out.println("Name is:"+jd.getName());
	}
```

}

OUTPUT:-



Problems Javadoc Declaration Console

```
<terminated> JavaBeean [Java Application] C:\Program Files\J
Age is:23
Name is:ANKIT
```

---

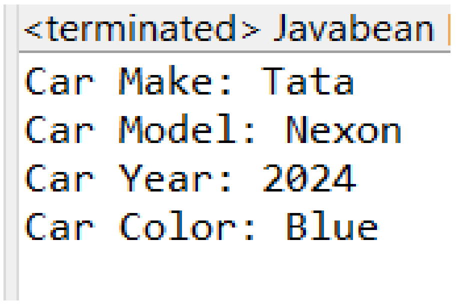
8. Create a JavaBean class Car with properties make, model, year, and color. Implement the required no-argument constructor, getter and setter methods for each property. Write a main class to create an instance of Car, set its properties, and print the car details.

CODE:-

```
package EDemo;

import java.io.Serializable;
class Cars implements Serializable {
	private String make;
	private String model;
	private int year;
	private String color;
	public Cars() {}
	public String getMake() {
		return make;
	}
	// Setter for make
	public void setMake(String make) {
		this.make = make;
	}
	// Getter for model
	public String getModel() {
		return model;
	}
	// Setter for model
	public void setModel(String model) {
		this.model = model;
	}
	// Getter for year
	public int getYear() {
		return year;
	}
	// Setter for year
	public void setYear(int year) {
		this.year = year;
	}
	// Getter for color
	public String getColor() {
		return color;
	}
```

```
        // Setter for color
        public void setColor(String color) {
                this.color = color;
        }
}
public class Javabean { // main class
        public static void main(String[] args) {
                // Create an object of Car
                Cars car = new Cars();
                // Seting thepropeerties of car
                car.setMake("Tata");
                car.setModel("Nexon");
                car.setYear(2024);
                car.setColor("Blue");
                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:-



```
<terminated> Javabean
Car Make: Tata
Car Model: Nexon
Car Year: 2024
Car Color: Blue
```
