IPS ACADEMY INDORE

INSTITUTE OF ENGINEERING & SCIENCE COMPUTER SCIENCE & ENGINEERING DEPARTMENT – DATA SCIENCE

Lab Manual

(Jan-July 2024)

Object Oriented Programming & Methodology

Name : **Bhakti Panchal** Year : **2nd**

Semester : **4th** Enrollment no : **0808DS221030**

IPS ACADEMY, INSTITUTE OF ENGINEERING AND SCIENCE

CONTENTS

- Vision & Mission of the Institute
- Vision & Mission of the Department
- PEOs
- POs
- COs
- Content beyond Syllabus.
- Laboratory Regulations and Safety Rules
- Index
- Experiments

IPS ACADEMY, INSTITUTE OF ENGINEERING AND SCIENCE

Vision of the Institute

To be the fountainhead of novel ideas & innovations in science & technology & persist to be a foundation of pride for all Indians.

Mission of the Institute

- To provide value based broad Engineering, Technology and Science where education in students is urged to develop their professional skills.
- To inculcate dedication, hard work, sincerity, integrity and ethics in building up overall professional personality of our student and faculty.
- To inculcate a spirit of entrepreneurship and innovation in passing out students.
- To instigate sponsored research and provide consultancy services in technical, educational and industrial areas.

IPS ACADEMY, INSTITUTE OF ENGINEERING AND SCIENCE

Vision of the Department

Attaining global recognition in computer science and engineering education, research and training to meet the growing needs of the industry and society.

Mission of the Department

Provide quality undergraduate and postgraduate education, in both the theoretical and applied foundations of computer science, and train students to effectively apply this education to solve real-world problems, thus amplifying their potential for lifelong high-quality careers.

IPS ACADEMY, INSTITUTE OF ENGINEERING AND SCIENCE

Program Education Objectives (PEOs)

PEO 1:

To prepare students for successful careers in software industry that meet the needs of Indian and multinational companies.

PEO 2:

To provide students with solid foundation in mathematical, scientific and engineering fundamentals to solve engineering problems and required also to pursue higher studies.

PEO 3:

To develop the ability to work with the core competence of computer science & engineering i.e. software engineering, hardware structure & networking concepts so that one can find feasible solution to real world problems.

PEO 4:

To inseminate in students effective communication skills, team work, multidisciplinary approach, and an ability to relate engineering issues to broader social context.

PEO 5:

To motivate students perseverance for lifelong learning and to introduce them to professional ethics and codes of professional practice.

Program Outcomes (POs)

PO1. Engineering knowledge: Apply the knowledge of mathematics, science, engineering fundamentals, and an engineering specialization to the solution of complex engineering problems.

PO2. Problem analysis: Identify, formulate, research literature, and analyze complex engineering problems reaching substantiated conclusions using first principles of mathematics, natural sciences, and engineering sciences.

PO3.Design/development of solutions: Design solutions for complex engineering problems and design system components or processes that meet the specified needs with appropriate consideration for the public health and safety, and the cultural, societal, and environmental considerations.

PO4. Conduct investigations of complex problems: Use research-based knowledge and research methods including design of experiments, analysis and interpretation of data, and synthesis of the information to provide valid conclusions.

PO5. Modern tool usage: Create, select, and apply appropriate techniques, resources, and modern engineering and IT tools including prediction and modeling to complex engineering activities with an understanding of the limitations.

PO6. The engineer and society: Apply reasoning informed by the contextual knowledge to assess societal, health, safety, legal and cultural issues and the consequent responsibilities relevant to the professional engineering practice.

PO7. Environment and sustainability: Understand the impact of the professional engineering solutions in societal and environmental contexts, and demonstrate the knowledge of, and need for sustainable development.

PO8. Ethics: Apply ethical principles and commit to professional ethics and responsibilities and norms of the engineering practice.

PO9. Individual and team work: Function effectively as an individual, and as a member or leader in diverse teams, and in multidisciplinary settings.

PO10. Communication: Communicate effectively on complex engineering activities with the engineering community and with society at large, such as, being able to comprehend and write effective reports and design documentation, make effective presentations, and give and receive clear instructions.

PO11. Project management and finance: Demonstrate knowledge and understanding of the engineering and management principles and apply these to one's own work, as a member and leader in a team, to manage projects and in multidisciplinary environments.

PO12. Life-long learning: Recognize the need for, and have the preparation and ability to engage in independent and life-long learning in the broadest context of technological change

Program Specific Outcomes (PSOs)

PSO1. Academic competence:
(i) Understand fundamental concepts in statistics, mathematics and Computer Science.
(ii) Demonstrate an understanding of various analysis tools and software Used in data science.
PSO2. Personal and Professional Competence:
(i) Apply laboratory-oriented problem solving and be capable in Data Visualization and Interpretation.
(ii) Solve case studies by applying various technologies, comparing Results and analyzing inferences.
(iii) Develop problem solving approach and present output with effective Presentation and communication skills
PSO3. Research Competence:
(i) Design and develop tools and algorithms.
(ii) Contribute in existing open sources platforms
(iii) Construct use case based models for various domains for greater perspective.

.

Department of Computer Science & Engineering – Data Science

IPS ACADEMY, INSTITUTE OF ENGINEERING AND SCIENCE

Course Outcome of OOPM(LC-DS06)

CO1. Understand object-oriented programming principles and apply them in solving Problems.

CO2.Develop skill in data abstraction and message passing

CO3.Understand fundamentals of relationship amongst objects

CO4.Learn about the need of exception and errors

CO5.Understand the concept of threads

IPS ACADEMY, INSTITUTE OF ENGINEERING AND SCIENCE

Laboratory Regulations and Safety Rules

1. Without Prior permission do not enter into the Laboratory.
2. While entering into the LAB students should wear their ID cards.
3. The Students should come with proper uniform.
4. Student should not use mobile phone inside the laboratory.
5. Students should sign in the LOGIN REGISTER before entering into the laboratory.
6. Students should come with observation and record note book to the laboratory.
7. Do not change any computer setting.
8. Students should maintain silence inside the laboratory.
9. After completing the laboratory exercise, make sure to SHUTDOWN the system properly.

IPS ACADEMY, INSTITUTE OF ENGINEERING AND SCIENCE

INDEX

S.NO.	Experiment Name	Date	Grade	Signature
1.	Write a program to show Concept of classes in java.(CO1)			
2.	Write a java program showing the concept of constructor in java. (CO2)			
3.	Write a program to show concept of array in java.CO2)			
4.	Write a program to show inheritance in java.(CO3)			
5.	Write a java program show Polymorphism in java. (CO2)			
6.	Write a program to show interfacing between two class.(CO3)			
7.	Write a program to show Exception Handling in java. (CO4)			
8.	Write a program to add a Class to a Package.(CO4)			
9.	Write a program to show Life Cycle of a Thread.(CO5)			

IPS ACADEMY, INSTITUTE OF ENGINEERING AND SCIENCE

Experiment No. 1

Write a program to show concept of classes in java.

Objective: We have created a class with two method one for initializing variables and In another class we have make the main method and call out the method of other class.

```
class Shape {
   String name;
   int vertex;
   int area;
}
public class Manual_Programs {
   public static void main(String[] args) {
      System.out.println("calulating the area of shape");
```

```
        Shape quadrilateral = new Shape();
        quadrilateral.name = "square";
        quadrilateral.vertex = 4;
        int side = 8;
        quadrilateral.area = side*side ;
        System.out.println("quadrilateral.name);
        System.out.println("quadrilateral.vertex);
        System.out.println("quadrilateral.area);
        Shape quadri = new Shape();
        quadri.name = "rectangle";
        quadri.vertex = 4;
        int length = 7;
```



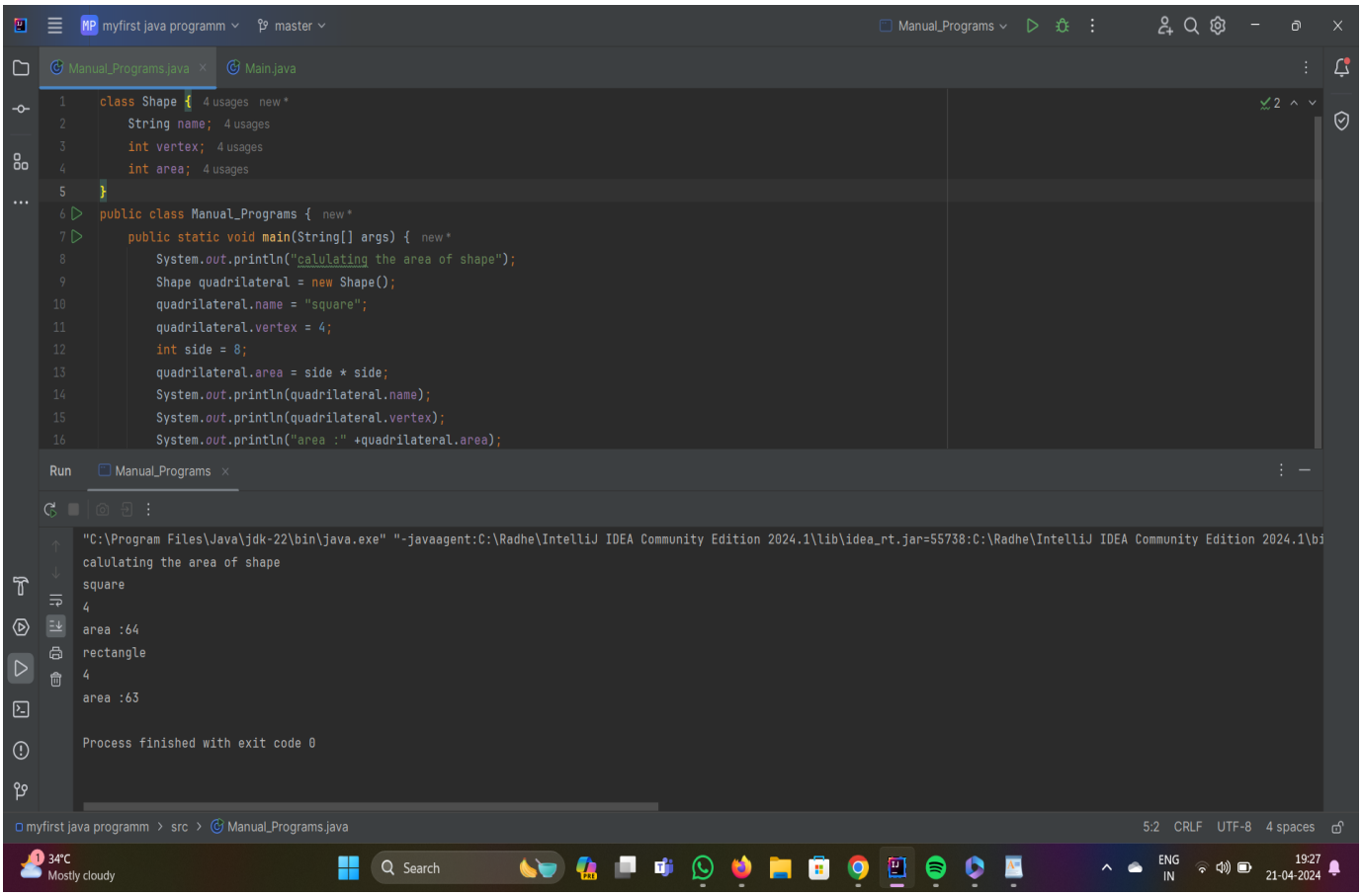
```
        int breadth = 9;
        quadri.area = length * breadth;
        System.out.println(quadri.name + "\n" + quadri.vertex + "\n" + "area :"+quadri.area);
    }
}
```

OUTPUT :

```
Square
4
area : 64
Rectangle
4
area : 63
```

Department of Computer Science & Engineering – Data Science

IPS ACADEMY, INSTITUTE OF ENGINEERING AND SCIENCE


myfirst java programm
master
Manual_Programs
Manual_Programs.java
Main.java
class Shape { 4 usages new *
String name; 4 usages
int vertex; 4 usages
int area; 4 usages
}
public class Manual_Programs { new *
public static void main(String[] args) { new *
System.out.println("calulating the area of shape");
Shape quadrilateral = new Shape();
quadrilateral.name = "square";
quadrilateral.vertex = 4;
int side = 8;
quadrilateral.area = side * side;
System.out.println(quadrilateral.name);
System.out.println(quadrilateral.vertex);
System.out.println("area :" +quadrilateral.area);
Run
Manual_Programs
"C:\Program Files\Java\jdk-22\bin\java.exe" "-javaagent:C:\Radhe\IntelliJ IDEA Community Edition 2024.1\lib\idea_rt.jar=55738:C:\Radhe\IntelliJ IDEA Community Edition 2024.1\bi
calulating the area of shape
square
4
area :64
rectangle
4
area :63
Process finished with exit code 0
myfirst java programm > src > Manual_Programs.java
5:2 CRLF UTF-8 4 spaces
34°C
Mostly cloudy
Search
ENG
IN
19:27
21-04-2024

Viva Questions

Q.1 what do you mean by classes?

class is a blueprint or a template for creating objects. It serves as a model that defines the attributes (variables) and behaviors (methods) that objects of that class will have.

Here are some key points about classes in Java:

Attributes (Variables): A class defines the attributes or variables that represent the state of the objects created from that class. These variables hold the data associated with the objects.

Methods (Functions): A class also defines methods, which are functions that represent the behavior or actions that objects of that class can perform. Methods can manipulate the object's state, perform operations, or provide information about the object.

A class describes a group of objects with similar properties, common behavior and common relationship with each other. It is a user-defined data type which has some data members and member functions.

Q.2 what do you mean by objects?

Ans- An object in Java is a basic unit of Object-Oriented Programming and represents real-life entities. Objects are the instances of a class that are created to use the attributes and methods of a class. A typical Java program creates many objects, which as you know, interact by invoking methods. An object consists of :

1. **State**: It is represented by attributes of an object. It also reflects the properties of an object.
2. **Behavior**: It is represented by the methods of an object. It also reflects the response of an object with other objects.
3. **Identity**: It gives a unique name to an object and enables one object to interact with other objects.

Example of an object: dog

Q.3 How object is represented in memory?

Ans-An object is basic runtime entity in an object oriented system. Programming problems are analyzed in terms of objects. Objects take up space in the memory and have an associated address. These are also known as class variables.

In Java, all objects are dynamically allocated on Heap. This is different from C++ where objects can be allocated memory either on Stack or on Heap. In JAVA , when we allocate the object using new(), the object is allocated on Heap, otherwise on Stack if not global or static. In Java, when we only declare a variable of a class type, only a reference is created (memory is not allocated for the object). To allocate memory to an object, we must use new(). So the object is always allocated memory on the heap (See this for more details).

There are two ways to create an object of string in java:

- **By string literal**
- **By new keyword**

Experiment No. 2

Write a java program showing the concept of constructor in java.

Objective : In these program we have created two constructor(Parameterized and Non Parameterized of a class

```
class college_students {
    private String name;
    private int id;

    //creating a parameterized constructor
    public college_students() {
        id = 1030;
        name = "Bhakti Panchal";
    }

    //creating a non parameterized constructor
    public college_students(String myname, int myid) {
        id = myid;
        name = myname;
    }

    public String getname() {
        return name;
    }
```

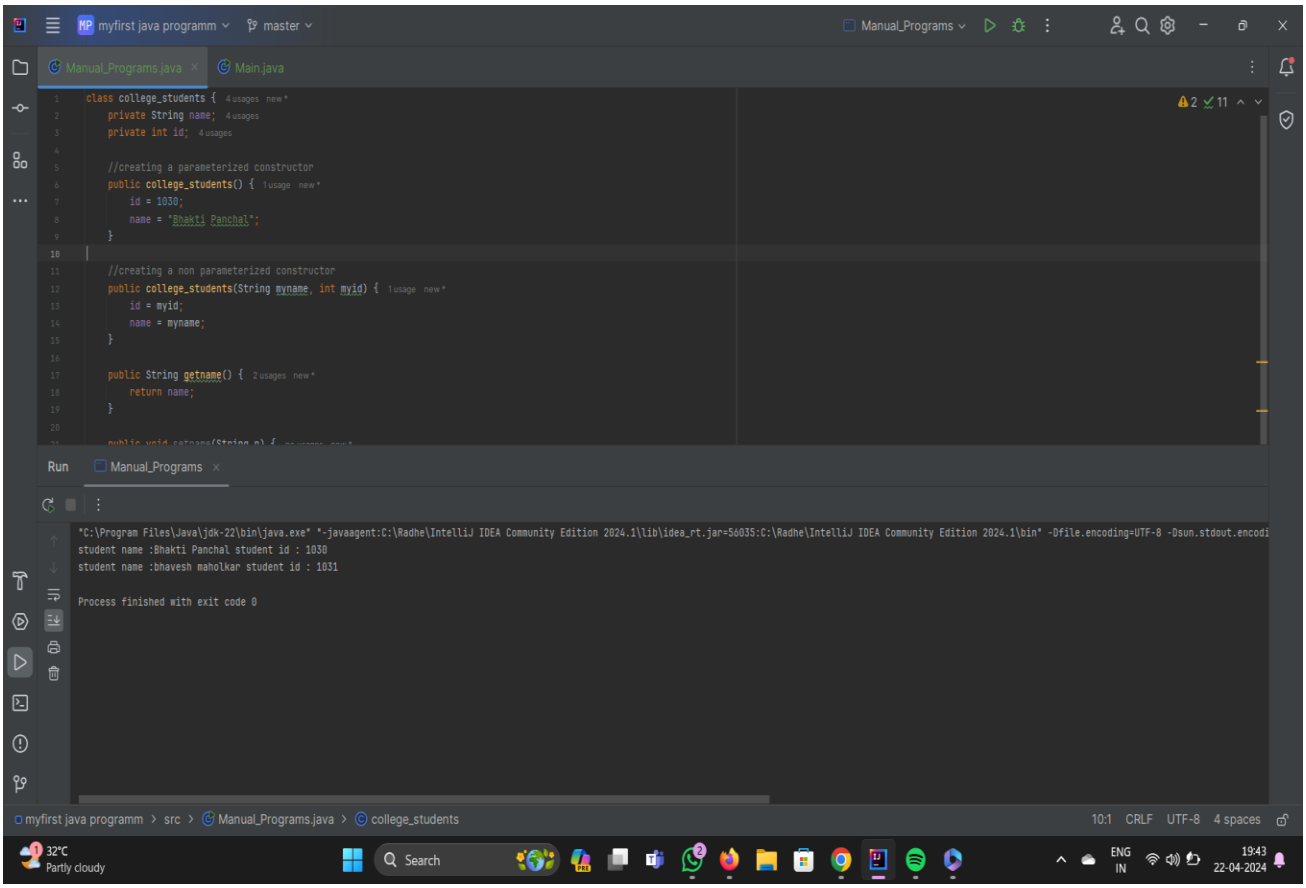
```
  public void setname(String n) {
      this.name = n;
   }

   public void setid(int k) {
      this.id = k;
   }

   public int getid() {
      return id;
   }
}

public class Manual_Programs {
   public static void main(String[] args) {
      //creating object using both constructor
      college_students student1 = new college_students(); //parameterized construcator
      college_students student2 = new college_students("Bhavesh Maholkar", 1031);//non parameterized constructor

      //getting data from objects
      System.out.println("student name :" + student1.getname() + " student id : " + student1.getid());
      System.out.println("student name :" + student2.getname() + " student id : " + student2.getid());
   }
}
```

OUTPUT :

student name :Bhakti Panchal student id : 1030

student name :Bhavesh Maholkar student id : 1031

IPS ACADEMY, INSTITUTE OF ENGINEERING AND SCIENCE



IPS ACADEMY, INSTITUTE OF ENGINEERING AND SCIENCE

Viva Questions

Q.1 Explain the concept of constructor and destructor?

Ans- a constructor is a special type of method that is automatically called when an object of a class is created. The primary purpose of a constructor is to initialize the newly created object. Constructors have the same name as the class and do not have a return type, not even void.Here are some key points about constructors in Java:

Initialization: Constructors are used to initialize the state of an object. This includes initializing instance variables, setting up the object's initial state, and performing any necessary setup tasks.

Automatic Invocation: Constructors are automatically called when an object of a class is created using the new keyword. This means that you don't need to explicitly call a constructor; it's invoked automatically during object creation.

Name: Constructors have the same name as the class they belong to. This distinguishes them from other methods in the class.

No Return Type: Constructors do not have a return type, not even void. This is because their primary purpose is initialization rather than computation or returning a value.

Overloading: Like other methods, constructors can be overloaded, which means you can have multiple constructors with different parameter lists in the same class. This allows for object creation with different initializations.

Default Constructor: If you don't provide any constructors in your class, Java automatically provides a default constructor. This constructor takes no arguments and initializes instance variables to their default values (0 for numeric types, false for boolean, null for reference types).

Access Modifiers: Constructors can have access modifiers such as public, private, protected, or package-private (no explicit modifier). The choice of access modifier determines from where the constructor can be accessed.

Constructor is used to initialize the variables and those variables are then used for processing. If variables are not initialized then it contains some garbage value. Constructor has no return type and name same as class name.

Destructor is used to destroy or used to free up the memory occupied by our object. It is followed by a tilde sign (~), as constructors it has no return type and has same name as class name.

Q.2 Explain types of constructor?

Ans.Thereare 3 basic type of constructor:-

In Java, constructors can be categorized into several types based on their characteristics and usage:

1. **Default Constructor**: If you do not explicitly provide any constructors in your class, Java automatically provides a default constructor. This constructor takes no arguments and initializes instance variables to their default values (0 for numeric types, `false` for boolean, `null` for reference types).

Example:

```java
public MyClass() {
    // Default constructor
}
```

2. **Parameterized Constructor:** A constructor that accepts one or more parameters. It is used to initialize the object with specific values provided during object creation.

Example:

```java
public MyClass(int value) {
    // Parameterized constructor
    this.value = value;
}
```

3. **Copy Constructor** A constructor that accepts an object of the same class as a parameter and initializes a new object with the same state as the passed object. It is used to create a new object with the same state as an existing object.

Example:


```
public MyClass(MyClass obj) {
    // Copy constructor
    this.value = obj.value;
}
```

4. **Constructor Overloading**: In Java, you can have multiple constructors in a class with different parameter lists. This is called constructor overloading. It allows objects to be initialized in different ways based on the parameters provided during object creation.

Example:

```java
public MyClass() {
    // Non-parameterized constructor
    this.value = 0; // Default value
}

public MyClass(int value) {
    // Parameterized constructor
    this.value = value;
}
```

5. **Private Constructor:** A constructor that is declared with the `private` access modifier. It is used to prevent the instantiation of objects of the class from outside the class itself. Private

constructors are often used in classes that contain only static methods and fields and are not meant to be instantiated.

Example:

```
public class Singleton {
    private static Singleton instance;
```



```
    // Private constructor
    private Singleton() {
        // Initialization code
    }

    public static Singleton getInstance() {
        if (instance == null) {
            instance = new Singleton();
        }
        return instance;
    }
}
```

These are the common types of constructors in Java. Each type serves a specific purpose and can be used based on the requirements of the class and the application.

Experiment 3

Write a program to show concept of array in java.

Objective: In these program we have declared an array and find the sum of it's element. Display method is used to display the array.

```
class Student {
    private String name;
    private int marks;

    // Constructor
    public Student(String name, int marks) {
        this.name = name;
        this.marks = marks;
    }

    // Getter methods
    public String getName() {
```

```
        return name;
    }

    public int getMarks() {
        return marks;}
```



```
}
public class Manual_Programs {
    public static void main(String[] args) {
        // Creating an array of 10 students
        Student[] students = new Student[10];
        {
            for(int i=0;i<students.length;i++){
                System.out.println("Total number of students :"+students.length);
            }
        }
        // Initializing students with names and marks
        students[0] = new Student("Aarya", 85);
        students[1] = new Student("Abhish", 78);
        students[2] = new Student("Bhavesh", 92);
        students[3] = new Student("Deveshi", 80);
        students[4] = new Student("Dimple", 88);
```

```
        students[5] = new Student("Harsh", 95);
        students[6] = new Student("Jayash", 70);
        students[7] = new Student("Kunal", 83);
        students[8] = new Student("Mrinal", 91);
        students[9] = new Student("Naina", 75);
```

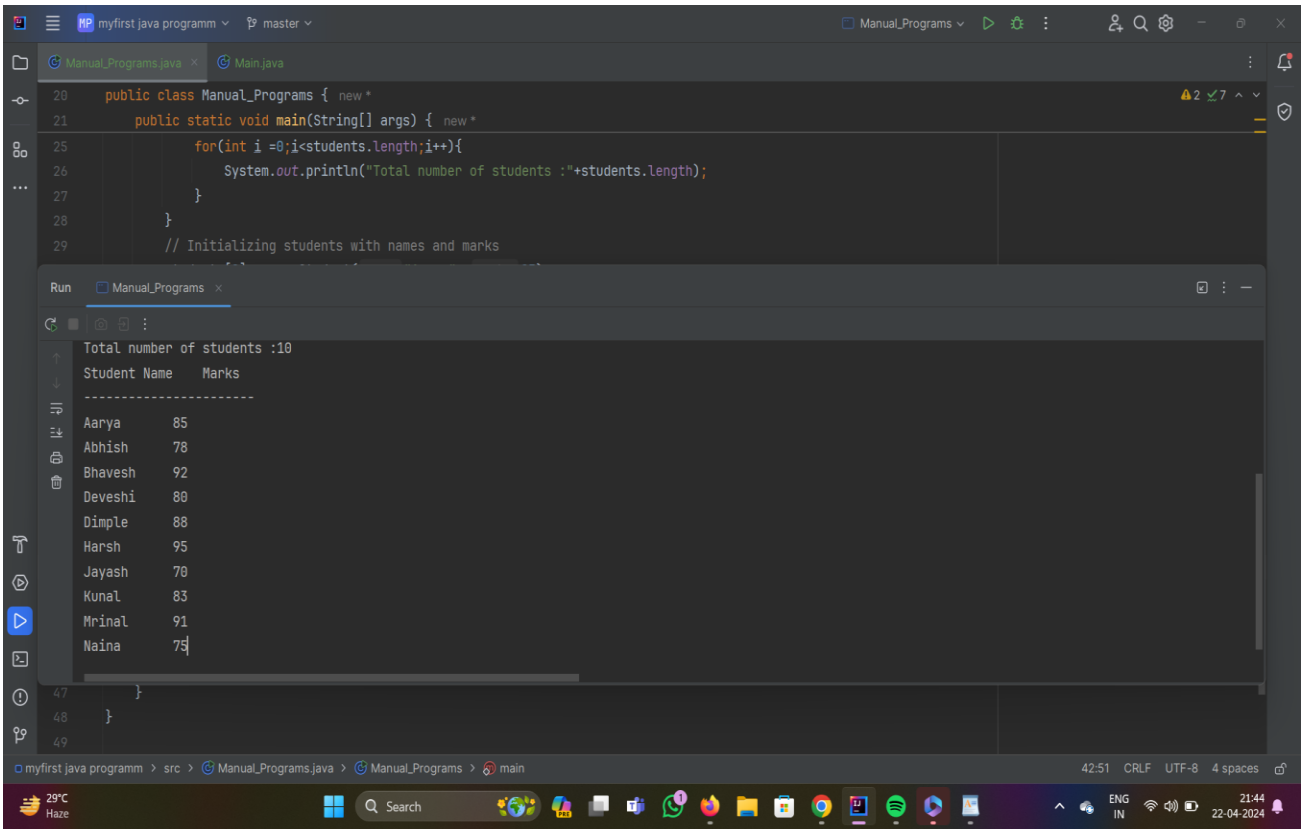


```
        // Printing student names and marks
        System.out.println("Student Name\tMarks");
        System.out.println("----------------------");
        for (Student student : students) {
            System.out.println(student.getName() + "\t\t" + student.getMarks());
        }
    }
}
```

OUTPUT:

```
Total number of students :10
Student Name  Marks
----------------------
Aarya          85
Abhish         78
Bhavesh        92
Deveshi        80
```

Dimple	88
Harsh	95
Jayash	70
Kunal	83
Mrinal	91
Naina	75



Department of Computer Science & Engineering – Data Science

IPS ACADEMY, INSTITUTE OF ENGINEERING AND SCIENCE

Viva Questions

Q.What is an array?

Ans:- In Java, an array is a data structure that stores a fixed-size sequential collection of elements of the same type. Each element in an array is accessed by an index. The index starts from 0 and goes up to one less than the size of the array. Arrays in Java are objects, and they can be of any data type, including primitive types (like int, char, double, etc.) and reference types (like objects of a class).Here are some key points about arrays in Java:

Fixed Size: Once an array is created, its size cannot be changed. You specify the size of the array when you create it, and it remains fixed throughout its lifetime.

Indexed Access: Elements in an array are accessed using an index, which is an integer value starting from 0 to length - 1, where length is the size of the array.

Homogeneous Elements: All elements in an array must be of the same type. This means you cannot mix different types of elements in the same array.

Declared and Initialized: You must declare an array before you can use it. Declaration specifies the type of elements the array will hold and allocates memory for the array. Initialization assigns initial values to the elements of the array.

Reference Type: In Java, arrays are reference types, which means when you create an array, you're actually creating an object on the heap. Therefore, you need to use the new keyword to create an array.

Length Property: Arrays have a length property that indicates the number of elements in the array. It is a final variable and cannot be changed.

An array is a data structure that stores a collection of elements of the same data type in contiguous memory locations. Each element in an array is identified by its index or position within the array, which starts at 0. Arrays are commonly used in programming to store and manipulate data.

Q.2 What is the property in java use to know the length of array?

Ans:- In Java, you can use the length property to determine the length of an array. This property provides the number of elements in the array. Here's how you can use it: java

```
int[] myArray = {1, 2, 3, 4, 5};

int length = myArray.length;

System.out.println("The length of the array is: " + length);
```


This will output:

The length of the array is: 5

length property of arrays in java help in knowing the length of array.

Q.3 What are basic use of array ?

Ans:- Arrays are fundamental data structures in programming languages like Java, C, Python, and many others. They serve several basic purposes:

Grouping Data: Arrays allow you to store multiple values of the same type under a single name. For example, you can create an array of integers to store a list of numbers.

Accessing Elements: You can access individual elements of an array using their index. In most programming languages, arrays are zero-indexed, meaning the first element is at index 0, the second at index 1, and so on.

Iteration: Arrays are often used in loops to perform operations on each element sequentially. For example, you might iterate over an array to calculate the sum of its elements or to find the maximum value.

Passing Parameters: Arrays are frequently used to pass a collection of values as a single parameter to functions or methods. This is particularly useful when you need to work with a variable number of values or when you want to modify the values in the array directly within the function.

Storing Collections: Arrays are used to represent collections of items, such as a list of students in a class, scores in a game, or temperatures over time. They provide a convenient way to manage and manipulate these collections.

Overall, arrays are essential for managing and manipulating collections of data in programming, offering efficiency and flexibility in organizing and accessing elements.

Arrays are used to store similar type of data. We can also store multiple dimension matrix using array.

Experiment 4

Write a program to show inheritance in java.

Program1: Single level Inheritance

Objective: In these program we have created a Animal class which is extended by class Dog. This program help in understanding the single level inheritance.

```
// Parent class
class Office {
   protected String officeName;
   protected String officeLocation;

   // Constructor
   public Office(String name, String location) {
      officeName = name;
```

```
        officeLocation = location;
    }

    // Method to display office details
    public void displayOfficeDetails() {
        System.out.println("Office Name: " + officeName);
        System.out.println("Office Location: " + officeLocation);
    }
}

// Child class inheriting from Office
class Department extends Office {
    private String departmentName;
```



```
    // Constructor
    public Department(String name, String location, String deptName) {
        super(name, location); // Call the constructor of the parent class
        departmentName = deptName;
    }

    // Method to display department details
    public void displayDepartmentDetails() {
        System.out.println("Department Name: " + departmentName);
```

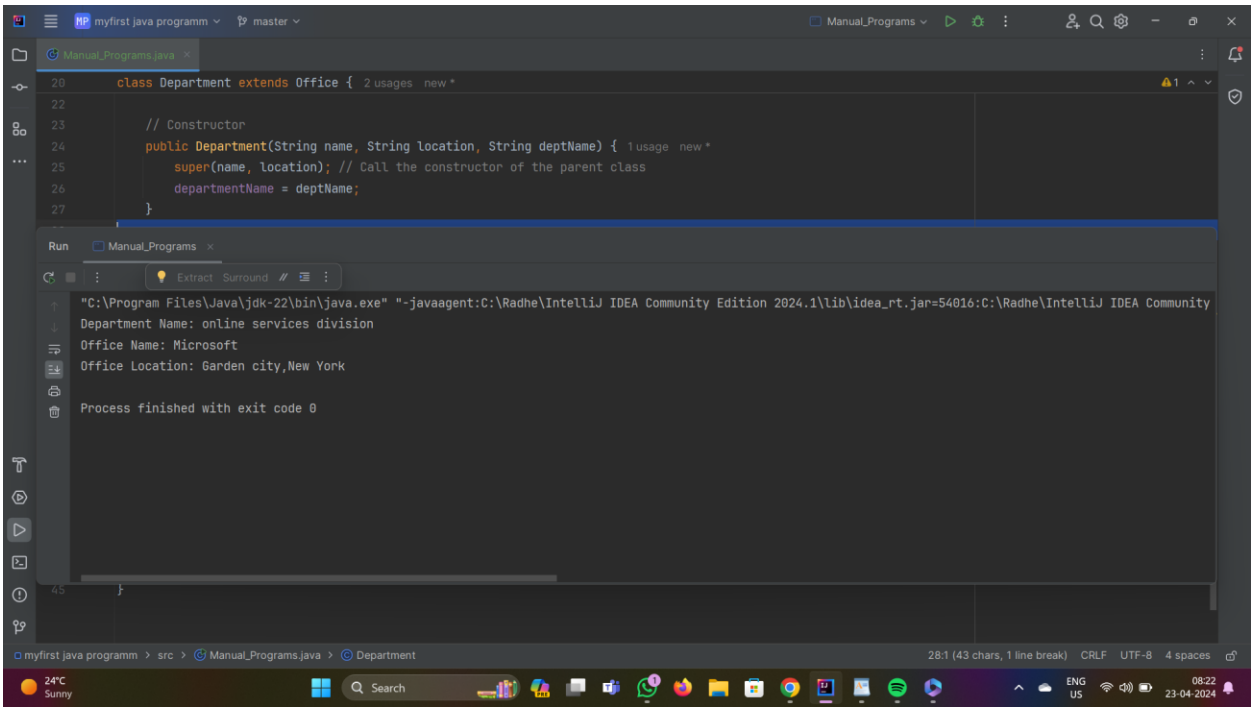
```
        displayOfficeDetails(); // Call the method of the parent class to display office details
    }
}

// Main class
public class Manual_Programs {
    public static void main(String[] args) {
        // Create an instance of the child class
        Department department = new Department("Microsoft", "Garden city,New York",
"online services division");

        // Call method to display department details
        department.displayDepartmentDetails();
    }
}
```


OUTPUT :

```
Department Name: online services division
Office Name: Microsoft
Office Location: Garden city,New York
```

Department of Computer Science & Engineering – Data Science

IPS ACADEMY, INSTITUTE OF ENGINEERING AND SCIENCE


myfirst java programm
master
Manual_Programs
Manual_Programs.java
class Department extends Office { 2 usages new *
// Constructor
public Department(String name, String location, String deptName) { 1 usage new *
super(name, location); // Call the constructor of the parent class
departmentName = deptName;
}
Run
Manual_Programs
"C:\Program Files\Java\jdk-22\bin\java.exe" "-javaagent:C:\Radhe\IntelliJ IDEA Community Edition 2024.1\lib\idea_rt.jar=54016:C:\Radhe\IntelliJ IDEA Community
Department Name: online services division
Office Name: Microsoft
Office Location: Garden city,New York
Process finished with exit code 0
}
myfirst java programm > src > Manual_Programs.java > Department
28:1 (43 chars, 1 line break) CRLF UTF-8 4 spaces
24°C
Sunny
Search
ENG
US
08:22
23-04-2024

Program2: Multilevel Inheritance

Objective: In these program we have class A which is extended by its child class B. This class B also extended by its child class C.

```
// Parent class
class Bird {
   protected String speciesName;
   protected String habitat;

   // Constructor
   public Bird(String speciesName, String habitat) {
      this.speciesName = speciesName;
      this.habitat = habitat;
   }

   // Method to display bird details
   public void displayBirdDetails() {
      System.out.println("Species: " + speciesName);
      System.out.println("Habitat: " + habitat);
   }
}
 // Subclass inheriting from Bird
class BirdSpecies extends Bird {
   protected String color;
     // Constructor
   public BirdSpecies(String speciesName, String habitat, String color) {
      super(speciesName, habitat); // Call the constructor of the parent class
      this.color = color;}
```

```
    // Method to display bird species details
    public void displayBirdSpeciesDetails() {
        System.out.println("Color: " + color);
        displayBirdDetails(); // Call the method of the parent class to display bird details
    }
}

// Subclass inheriting from BirdSpecies
class Parrot extends BirdSpecies {
    private String language;

    // Constructor
    public Parrot(String speciesName, String habitat, String color, String language) {
        super(speciesName, habitat, color); // Call the constructor of the parent class
        this.language = language;
    }

    // Method to display parrot details
    public void displayParrotDetails() {
        System.out.println("Language: " + language);
        displayBirdSpeciesDetails(); // Call the method of the parent class to display bird species
details
    }
}// Main class
public class Manual_Programs{
    public static void main(String[] args) {// Create an instance of the Parrot class
```

```
        Parrot parrot = new Parrot("African Grey Parrot", "Rainforest", "Reddish green", "Babble");


    // Call method to display parrot details
    parrot.displayParrotDetails();
  }
  }
```

OUTPUT :

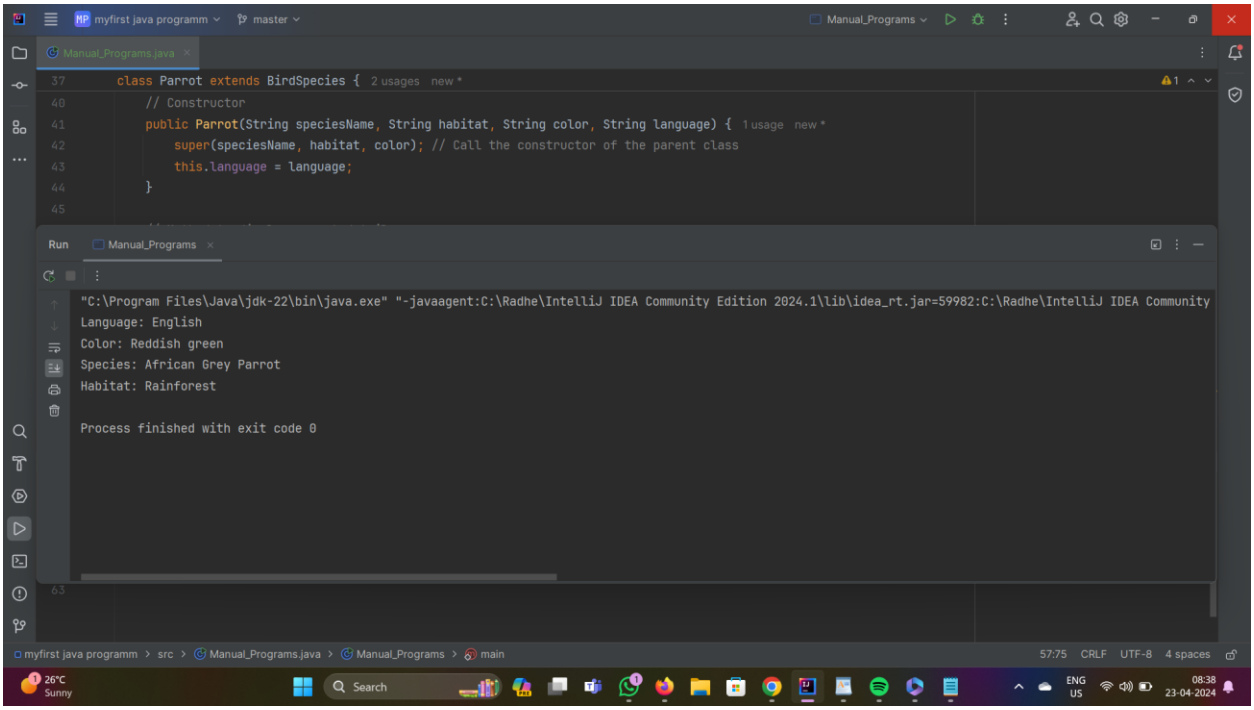
Language: Babble

Color: Reddish green

Species: African Grey Parrot

Habitat: Rainforest

IPS ACADEMY, INSTITUTE OF ENGINEERING AND SCIENCE



IPS ACADEMY, INSTITUTE OF ENGINEERING AND SCIENCE

Program 3: Hierarchal Inheritance

Objective: In these program we created a parent class Animal which is extended by its two child class Dog and Cat to understand the hierarchal Inheritance.

```
// Parent class
class Institute {
   protected String name;
   protected String location;

   // Constructor
   public Institute(String name, String location) {
      this.name = name;
      this.location = location;
   }

   // Method to display institute details
   public void displayInstituteDetails() {
      System.out.println("Name: " + name);
      System.out.println("Location: " + location);
   }
}
// Subclass inheriting from Institute
class University extends Institute {
   private String accreditation;
 // Constructor
```

```
    public University(String name, String location, String accreditation) {
        super(name, location); // Call the constructor of the parent class
        this.accreditation = accreditation;
```



```
}
    // Method to display university details
    public void displayUniversityDetails() {
        System.out.println("Accreditation: " + accreditation);
        displayInstituteDetails(); // Call the method of the parent class to display institute details
    }
}

// Subclass inheriting from Institute
class School extends Institute {
    private int gradeLevels;

    // Constructor
    public School(String name, String location, int gradeLevels) {
        super(name, location); // Call the constructor of the parent class
        this.gradeLevels = gradeLevels;
    }

    // Method to display school details
    public void displaySchoolDetails() {
        System.out.println("Grade Levels: " + gradeLevels);
```

```
        displayInstituteDetails(); // Call the method of the parent class to display institute details
    }
}
 // Main class
public class Manual_Programs{
```



```
public static void main(String[] args) {
        // Create an instance of the University class
        University university = new University("Stanford University", "California", "NAAC");

        // Create an instance of the School class
        School school = new School("Sant Joseph School", "Mumbai", 12);

        // Call method to display university details
        university.displayUniversityDetails();

        System.out.println(); // Add a blank line for separation

        // Call method to display school details
        school.displaySchoolDetails();
    }
}
```

OUTPUT:

Accreditation: NAAC

Name: Stanford University

Location: California

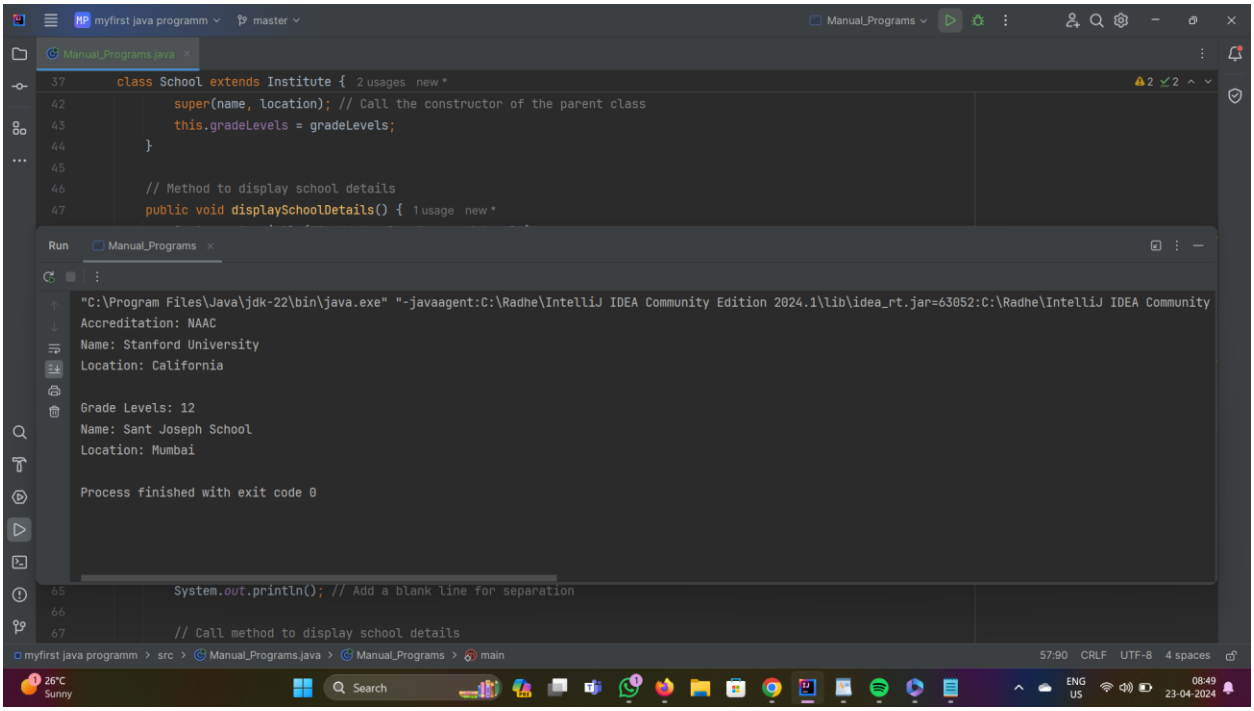
Grade Levels: 12

Name: Sant Joseph School

Location: Mumbai

Department of Computer Science & Engineering – Data Science

IPS ACADEMY, INSTITUTE OF ENGINEERING AND SCIENCE



Department of Computer Science & Engineering – Data Science

IPS ACADEMY, INSTITUTE OF ENGINEERING AND SCIENCE

Viva Questions

Q.1 What is inheritance in Java?

Ans- Inheritance in Java is a mechanism by which a class (subclass or child class) can inherit properties and behavior (methods) from another class (superclass or parent class). This facilitates code reuse and establishes a hierarchical relationship between classes.

Key points about inheritance in Java:

Superclass and Subclass: Inheritance involves two types of classes - the superclass (or parent class) and the subclass (or child class). The subclass inherits properties and methods from the superclass.

Syntax: In Java, inheritance is implemented using the extends keyword. When defining a subclass, you specify the superclass that it extends. For example:

java

```
class Subclass extends Superclass {
   // subclass members
}
```

Access Modifiers: Inherited members (fields and methods) can have different access modifiers in the subclass compared to the superclass. However, they cannot have a more restrictive access modifier. For example, if a method is public in the superclass, it can't be private in the subclass.

Single Inheritance: Java supports single inheritance, meaning a subclass can only extend one superclass. However, Java supports multiple levels of inheritance, allowing subclasses to further extend other subclasses.

Overriding Methods: Subclasses can override methods from the superclass by providing a new implementation. This allows customization of behavior specific to the subclass. To override a method, the method signature in the subclass must match the method signature in the superclass.Constructor Inheritance: Constructors are not inherited in Java, but they are invoked implicitly or explicitly in the subclass. If a constructor is not explicitly defined in the subclass, the compiler automatically inserts a call to the superclass constructor using super().

Object Class: Every class in Java directly or indirectly extends the Object class, which is the root of the class hierarchy in Java. This provides common methods like toString(), equals(), and hashCode() to all objects in Java.

Inheritance is a powerful feature in Java that promotes code reuse, modularity, and abstraction, enabling developers to create more maintainable and scalable software.Inheritance is a mechanism in Java by which a class (subclass or derived class) can inherit properties (fields and methods) from another class (superclass or base class)..

Q.2 What are the types of inheritance supported in Java?

Ans- there are several types of inheritance that are supported:

Single Inheritance: In single inheritance, a subclass inherits from only one superclass. Java supports single inheritance, meaning a class can have only one direct superclass.

Multilevel Inheritance: In multilevel inheritance, a subclass inherits from a superclass, and then another subclass inherits from the subclass, creating a chain of inheritance. Java supports multilevel inheritance, allowing classes to be organized into multiple levels of inheritance hierarchy.

Hierarchical Inheritance: In hierarchical inheritance, multiple subclasses inherit from a single superclass. Each subclass shares the properties and methods of the superclass, creating a

hierarchical structure. Java supports hierarchical inheritance, enabling classes to form a tree-like structure.

Multiple Inheritance (Through Interfaces): While Java does not support multiple inheritance of classes (i.e., inheriting from multiple superclasses), it supports a form of multiple inheritance through interfaces. A class can implement multiple interfaces, thereby inheriting abstract methods from each interface. This allows for a limited form of multiple inheritance in Java.

Hybrid Inheritance: Hybrid inheritance is a combination of multiple types of inheritance, such as single inheritance, multilevel inheritance, and hierarchical inheritance. Java supports hybrid inheritance through the combination of single inheritance, multilevel inheritance, and interfaces.

While Java does not support direct multiple inheritance of classes, it provides a flexible inheritance mechanism through interfaces, enabling developers to achieve the benefits of multiple inheritance while avoiding the complexities associated with it.

Experiment 5

Write a java program showing the concept of Ploymorphism.

Program1:Method Overloading

Objective: In these program we have ceated more than one show method with different parameter list to understand the method overloading.

```
class Automobiles {
  // Method overloading - same method name with different parameters
  public void start() {
    System.out.println("Starting the engine...");
  }

  public void start(String ignitionType) {
    System.out.println("Starting the engine using " + ignitionType + " ignition...");
  }
```

```
    public void start(int ignitionCode) {
        System.out.println("Starting the engine with ignition code: " + ignitionCode);
    }
}

public class Manual_Programs{
    public static void main(String[] args) {
        // Create an instance of Automobiles
        Automobiles car = new Automobiles();
 // Call different overloaded start methods
        car.start(); // Calls start() with no parameters
```



```
    car.start("key"); // Calls start(String ignitionType)
        car.start(1234); // Calls start(int ignitionCode)
    }
}
```

OUTPUT :

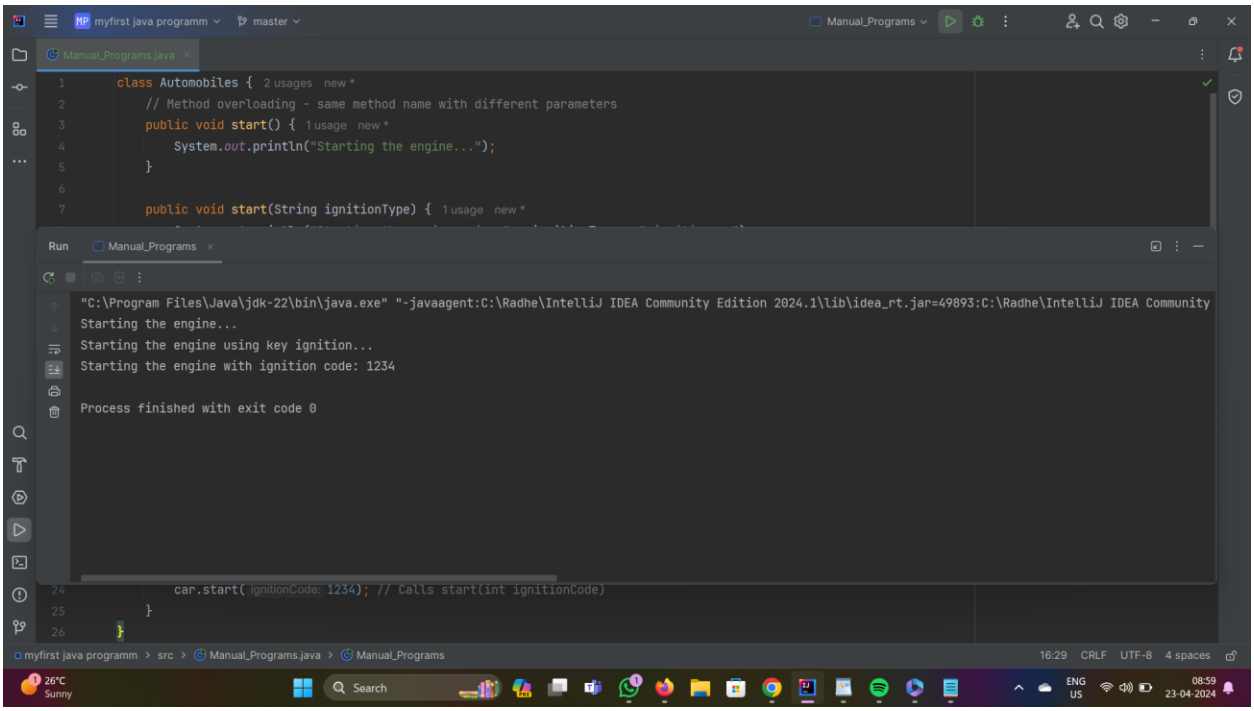
Starting the engine...

Starting the engine using key ignition...

Starting the engine with ignition code: 1234

Department of Computer Science & Engineering – Data Science

IPS ACADEMY, INSTITUTE OF ENGINEERING AND SCIENCE



Program2: method overriding

Objective: In these program we have created same method with same parameters but in different classes. This is method overriding.

```
// Parent class
class Vehicle {
    // Method to start the vehicle
    public void start() {
        System.out.println("Vehicle started.");
    }

    // Method to stop the vehicle
    public void stop() {
        System.out.println("Vehicle stopped.");
    }
}

// Subclass inheriting from Vehicle
class Car extends Vehicle {
    // Method overriding - overriding the start method from the superclass
    @Override
    public void start() {
        System.out.println("Car started.");
    }
}
 // Subclass inheriting from Vehicle
class Motorcycle extends Vehicle {
```

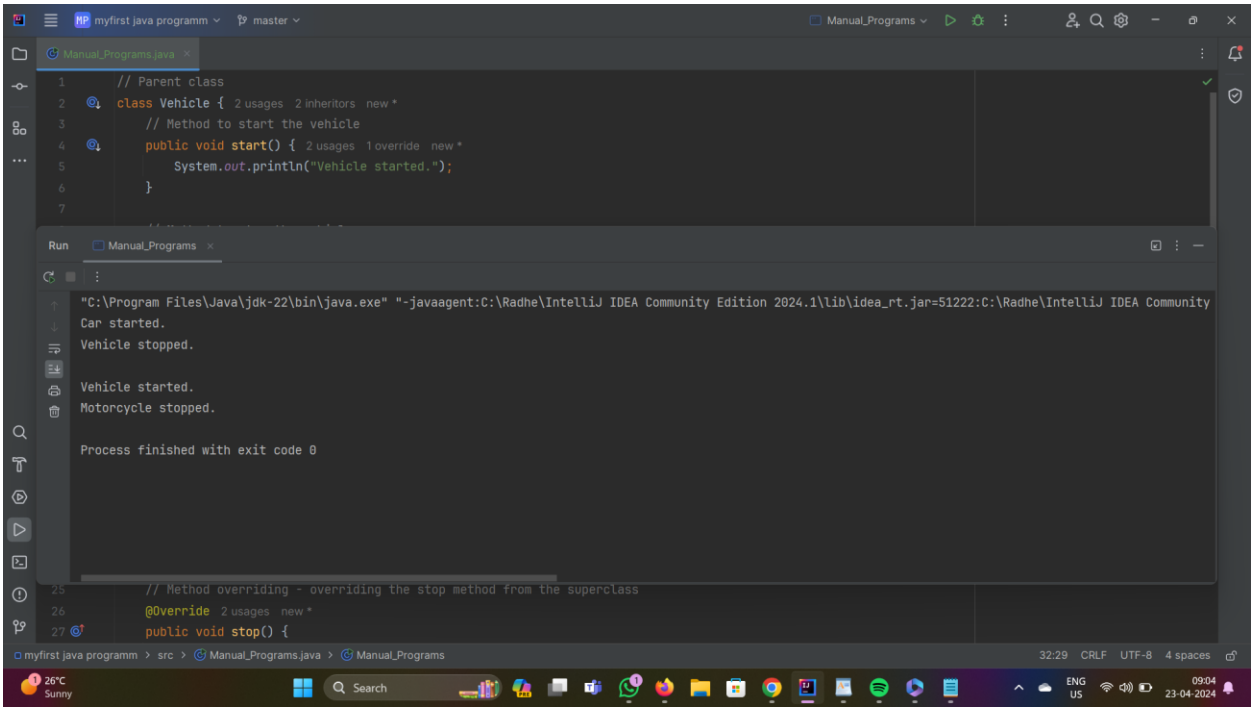
```
  // Method overriding - overriding the stop method from the superclass
   @Override
   public void stop() {
      System.out.println("Motorcycle stopped.");
   }
}

public class Manual_Programs{
   public static void main(String[] args) {
      // Create instances of Car and Motorcycle
      Car myCar = new Car();
      Motorcycle myMotorcycle = new Motorcycle();

      // Call the start and stop methods of both classes
      myCar.start(); // Calls the overridden start method in Car class
      myCar.stop();  // Calls the stop method in Vehicle class
      System.out.println(); // Add a blank line for separation
      myMotorcycle.start(); // Calls the start method in Vehicle class
      myMotorcycle.stop();  // Calls the overridden stop method in Motorcycle class
   }
}
```

OUTPUT :

```
Car started.
Vehicle stopped.
```

Vehicle started.

Motorcycle stopped.



Viva Questions:

Q.1 What is polymorphism in Java?

Ans- Polymorphism in Java refers to the ability of objects to take on multiple forms. In other words, it allows a single interface to be used for different data types or objects. There are two main types of polymorphism in Java:

- Compile-Time Polymorphism (Static Binding or Method Overloading): Compile-time polymorphism occurs when the method to be invoked is determined at compile time. It is achieved through method overloading. In method overloading, multiple methods with the same name are defined within the same class, but with different method signatures (different parameters or parameter types). The appropriate method is selected based on the number and types of arguments passed to it at compile time.
- Run-Time Polymorphism (Dynamic Binding or Method Overriding): Run-time polymorphism occurs when the method to be invoked is determined at runtime. It is achieved through method overriding. In method overriding, a subclass provides a specific implementation of a method that is already defined in its superclass. When a method is called on an object of the subclass, the JVM determines which implementation of the method to execute based on the actual type of the object at runtime.
- Polymorphism enables code reusability, flexibility, and extensibility in Java programming. It allows for more generic and flexible code by treating objects of different classes uniformly through a common interface or superclass. This is a key principle of object-oriented programming and helps in writing cleaner, more modular, and more maintainable code.

Polymorphism is a core concept in object-oriented programming where an object can take on multiple forms. In Java, polymorphism is achieved through method overriding and method overloading.

Q.2 Differentiate between compile-time polymorphism and runtime polymorphism.

Ans- Compile-time polymorphism (static polymorphism): Occurs when the compiler determines which method to call based on the method signature during compile-time. Method overloading is an example of compile-time polymorphism.

Runtime polymorphism (dynamic polymorphism): Occurs when the method to be invoked is determined at runtime based on the actual object being referred to. Method overriding is an example of runtime polymorphism.

Experiment 6

Write a program to show interfaces between two class.

Objective: In these program we have created a interface A which is extended by another interface B. Wit help of classes we have implemented this interfaces.

```
// Interface defining mathematical operations
interface MathOperations {
   int add(int a, int b);
   int subtract(int a, int b);
   int multiply(int a, int b);
   double divide(int a, int b);
}

// Class implementing the MathOperations interface
class Calculator implements MathOperations {
   @Override
   public int add(int a, int b) {
      return a + b;
   }
```

```
    @Override
    public int subtract(int a, int b) {
        return a - b;
    }
```



```
    @Override
    public int multiply(int a, int b) {
        return a * b;
    }

    @Override
    public double divide(int a, int b) {
        if (b == 0) {
            throw new ArithmeticException("Cannot divide by zero");
        }
        return (double) a / b;
    }
}

public class Manual_Programs{
    public static void main(String[] args) {
        // Create an instance of the Calculator class
        Calculator calculator = new Calculator();
```

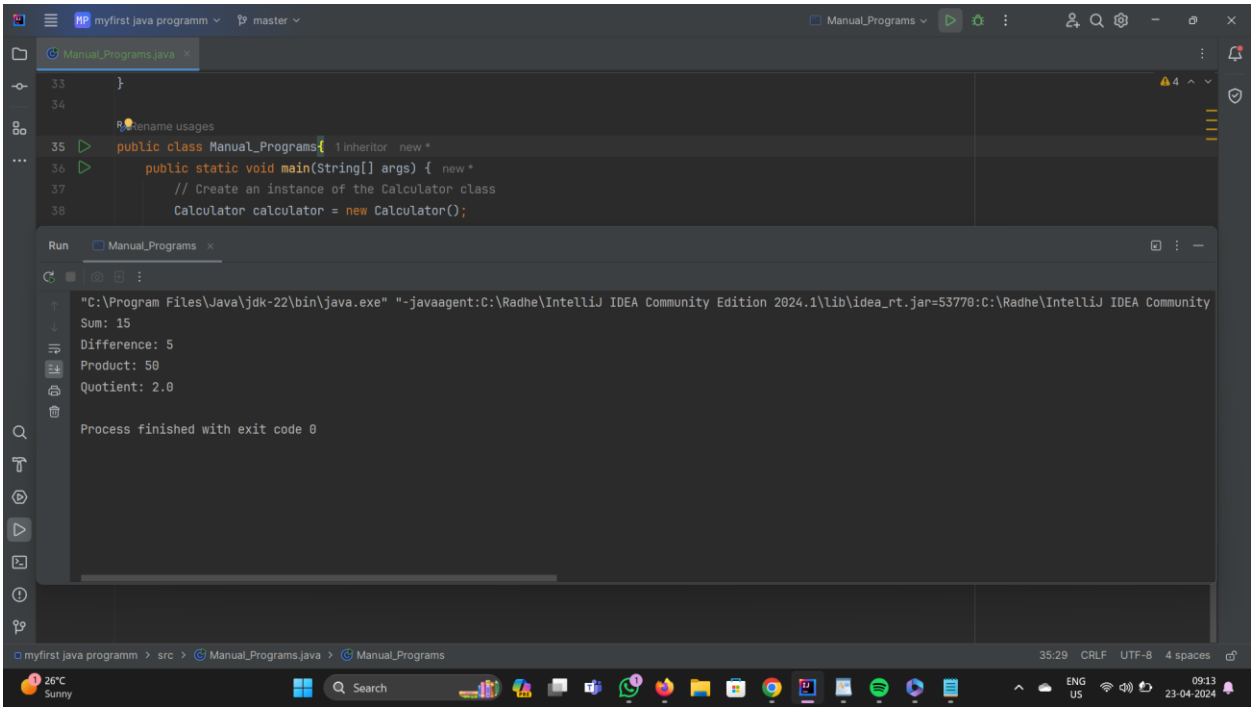
```
        // Perform mathematical operations using the interface methods
        int sum = calculator.add(10, 5);
        int difference = calculator.subtract(10, 5);
        int product = calculator.multiply(10, 5);
        double quotient = calculator.divide(10, 5);
    // Display the results
```



```
System.out.println("Sum: " + sum);
        System.out.println("Difference: " + difference);
        System.out.println("Product: " + product);
        System.out.println("Quotient: " + quotient);
    }
}
```

OUTPUT :

```
Sum: 15
Difference: 5
Product: 50
Quotient: 2.0
```


myfirst java programm
master
Manual_Programs
Manual_Programs.java
33 }
34
Rename usages
35 public class Manual_Programs{ 1 inheritor new *
36 public static void main(String[] args) { new *
37 // Create an instance of the Calculator class
38 Calculator calculator = new Calculator();
Run
Manual_Programs
"C:\Program Files\Java\jdk-22\bin\java.exe" "-javaagent:C:\Radhe\IntelliJ IDEA Community Edition 2024.1\lib\idea_rt.jar=53770:C:\Radhe\IntelliJ IDEA Community
Sum: 15
Difference: 5
Product: 50
Quotient: 2.0
Process finished with exit code 0
myfirst java programm > src > Manual_Programs.java > Manual_Programs
35:29 CRLF UTF-8 4 spaces
26°C
Sunny
Search
ENG
US
09:13
23-04-2024

Viva Question:

Q.1 What is an interface in Java?

Ans:-An interface in Java is a reference type that contains abstract methods as well as constants (static final variables). It provides a contract for classes to implement, specifying the methods that must be implemented by any class that implements the interface.

Q.2What is the difference between an interface and a class in Java?

Ans:-A class can contain both concrete methods and abstract methods, whereas an interface can only contain abstract methods (methods without implementation) and constants.

A class can be instantiated to create objects, whereas an interface cannot be instantiated.

A class can extend only one class (single inheritance), but it can implement multiple interfaces (multiple inheritance).

IPS ACADEMY, INSTITUTE OF ENGINEERING AND SCIENCE

Experiment 7

Write a program to show Exception Handling in java.

Objective: In these program we are learning to handle the exception with help of try, catch , finally , throw and throws keyword.

```
class MathsOperation {
   // Method to perform division operation
   public static double divide(int a, int b) throws ArithmeticException {
      if (b == 0) {
         throw new ArithmeticException("Cannot divide by zero");
      }
      return (double) a / b;
   }
}
```

```
public class Manual_Programs{
   public static void main(String[] args) {
      int dividend = 10;
      int[] divisors = {2, 0, 5};
         for (int divisor : divisors) {
         try {
            double result = MathsOperation.divide(dividend, divisor);
            System.out.println(dividend + " / " + divisor + " = " + result);
         } catch (ArithmeticException e) {
            System.out.println("Exception caught: " + e.getMessage());
         } finally {
            System.out.println("Inside finally block");}
```



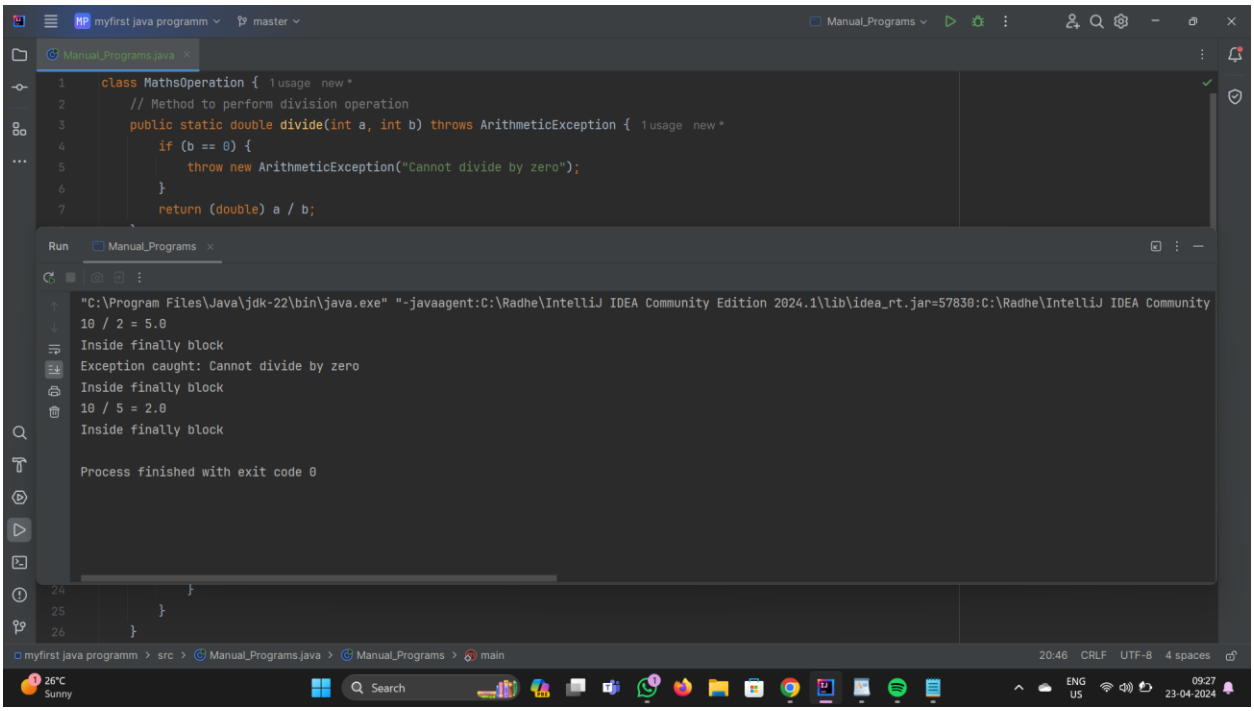
```
      }
   }
}
```

OUTPUT:

```
Inside finally block
Exception caught: Cannot divide by zero
Inside finally block
10 / 5 = 2.0
Inside finally block
```

Department of Computer Science & Engineering – Data Science

IPS ACADEMY, INSTITUTE OF ENGINEERING AND SCIENCE


myfirst java programm
master
Manual_Programs
Manual_Programs.java
class MathsOperation { 1 usage new *
// Method to perform division operation
public static double divide(int a, int b) throws ArithmeticException { 1 usage new *
if (b == 0) {
throw new ArithmeticException("Cannot divide by zero");
}
return (double) a / b;
Run
Manual_Programs
"C:\Program Files\Java\jdk-22\bin\java.exe" "-javaagent:C:\Radhe\IntelliJ IDEA Community Edition 2024.1\lib\idea_rt.jar=57830:C:\Radhe\IntelliJ IDEA Community
10 / 2 = 5.0
Inside finally block
Exception caught: Cannot divide by zero
Inside finally block
10 / 5 = 2.0
Inside finally block
Process finished with exit code 0
myfirst java programm > src > Manual_Programs.java > Manual_Programs > main
20:46 CRLF UTF-8 4 spaces
26°C
Sunny
Search
ENG
US
09:27
23-04-2024

Viva Questions

Q.1 what do you mean by an Exception?

Ans- An exception is a problem that arises during the execution of a program. A C++ exception is a response to an exceptional circumstance that arises while a program is running, such as an attempt to divide by zero.

Exceptions provide a way to transfer control from one part of a program to another. C++ exception handling is built upon three keywords: try, catch, and throw.

Q.2 Explain blocks contained in an Exception handler?

Ans-

Try block:

Statements which will raise errors at runtime are written in this block.

Syntax-

try(Expression);

Catch block: Runtime errors are caught by the catch block; It catches the Exception thrown by the throw statement in try block.

Syntax-

catch()

{

}

Throw block: If an Exception occurs in try block then it is thrown by this statement.

Syntax:

Throw(object)

{----------:

IPS ACADEMY, INSTITUTE OF ENGINEERING AND SCIENCE

Experiment 8

Write a program to add a Class to to a Package.

Objective: In these program we have created a package pack with some Methods.

package pack

package Myprograms;

```
public class Manual_Programs {
   public void displayDetails(String name, int age) {
      System.out.println("Name: " + name);
      System.out.println("Age: " + age);
   }

   // Method to calculate grade based on marks
   public char calculateGrade(int marks) {
      if (marks >= 90) {
         return 'A';
      } else if (marks >= 80) {
         return 'B';
      } else if (marks >= 70) {
         return 'C';
      } else if (marks >= 60) {
         return 'D';
      } else {
         return 'F';}
```

```
}
import Myprograms.Manual_Programs;
public class students{
    public static void main(String[] args) {
        // Create an instance of the Student class
        Student student = new Student();

        // Call the displayDetails method
        student.displayDetails("Alice", 22);

        // Call the calculateGrade method
        int marks = 75;
        char grade = student.calculateGrade(marks);
        System.out.println("Marks: " + marks);
        System.out.println("Grade: " + grade);
    }
}
```

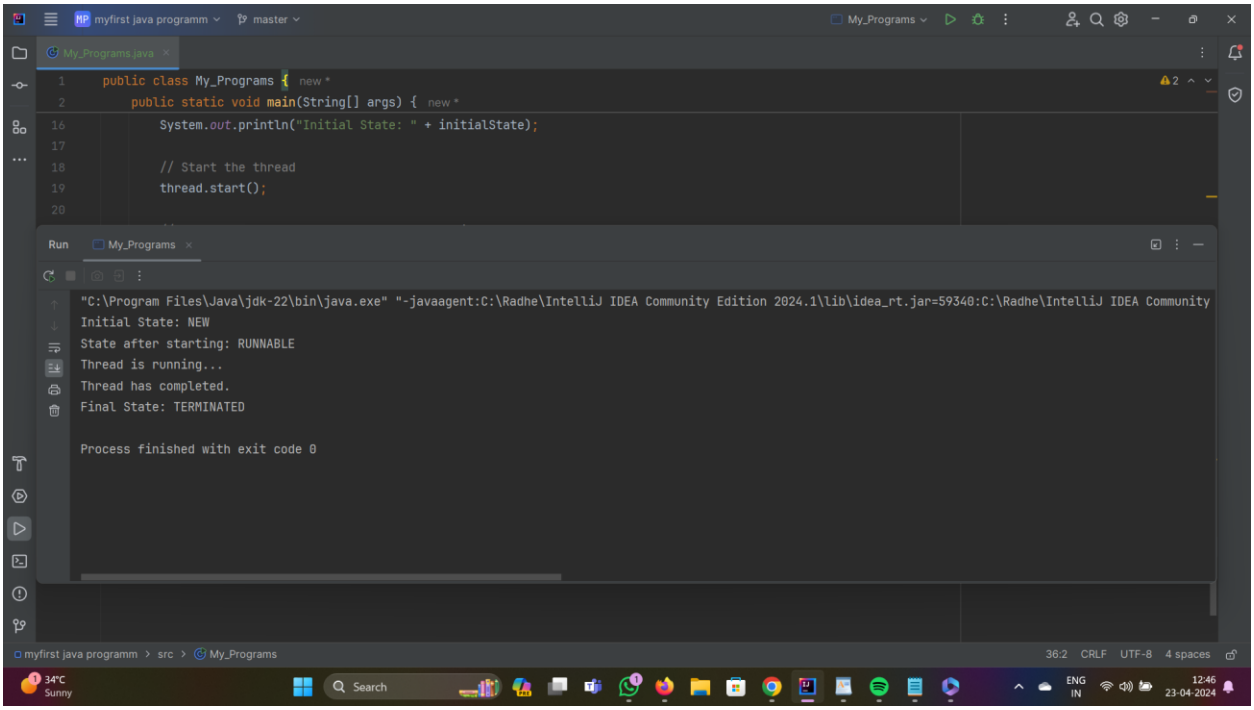
OUTPUT :

Name: Alice

Age: 22

Marks: 75

Grade: C


myfirst java programm
master
My_Programs
My_Programs.java
public class My_Programs { new *
public static void main(String[] args) { new *
System.out.println("Initial State: " + initialState);
// Start the thread
thread.start();
Run
My_Programs
"C:\Program Files\Java\jdk-22\bin\java.exe" "-javaagent:C:\Radhe\IntelliJ IDEA Community Edition 2024.1\lib\idea_rt.jar=59340:C:\Radhe\IntelliJ IDEA Community
Initial State: NEW
State after starting: RUNNABLE
Thread is running...
Thread has completed.
Final State: TERMINATED
Process finished with exit code 0
myfirst java programm > src > My_Programs
36:2 CRLF UTF-8 4 spaces
34°C
Sunny
Search
ENG
IN
12:46
23-04-2024

IPS ACADEMY, INSTITUTE OF ENGINEERING AND SCIENCE

Viva Questions

Q.1 What is a package in Java?

Ans- A package in Java is a mechanism to organize classes and interfaces into namespaces, making it easier to manage and maintain large codebases. It helps prevent naming conflicts and provides access protection.

Q.2 What is the purpose of using packages in Java?

Ans-The main purposes of using packages in Java are:

Organizing classes and interfaces into meaningful groups.

Preventing naming conflicts by providing a namespace for classes.

Controlling access to classes and interfaces using access modifiers (e.g., public, private, protected).

IPS ACADEMY, INSTITUTE OF ENGINEERING AND SCIENCE

Experiment 9

Write a program to show Life Cycle of a Thread

Objective: In these program we will understand the life cycle of thread.WE have extended the thread clas to implement the thread.

```
public class My_Programs {
   public static void main(String[] args) {
      // Create a new thread
      Thread thread = new Thread(() -> {
         System.out.println("Thread is running...");
         try {
            Thread.sleep(2000); // Simulate some task being performed by the thread
         } catch (InterruptedException e) {
            e.printStackTrace();
         }
         System.out.println("Thread has completed.");
      });

      // Get the initial state of the thread
      Thread.State initialState = thread.getState();
```

```
        System.out.println("Initial State: " + initialState);
        // Start the thread
        thread.start();
        // Get the state of the thread after starting
        Thread.State runningState = thread.getState();
        System.out.println("State after starting: " + runningState);
```



```
        try {
            // Wait for the thread to complete
            thread.join();
        } catch (InterruptedException e) {
            e.printStackTrace();
        }

        // Get the final state of the thread after completion
        Thread.State finalState = thread.getState();
        System.out.println("Final State: " + finalState);
    }
}
```

OUTPUT :

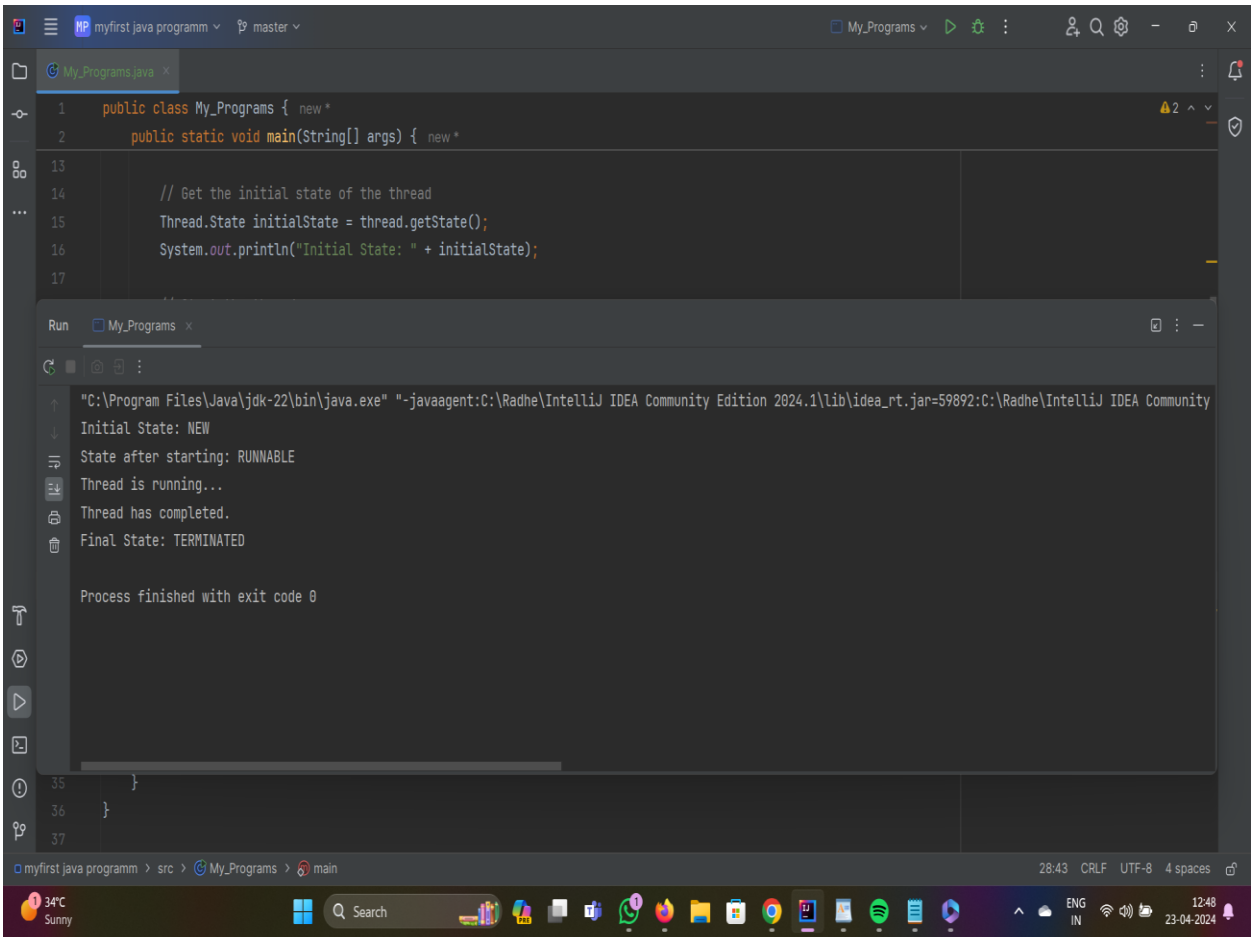
```
Initial State: NEW
Thread is running...
State after starting: RUNNABLE
```

Thread has completed.

Final State: TERMINATED

Department of Computer Science & Engineering – Data Science

IPS ACADEMY, INSTITUTE OF ENGINEERING AND SCIENCE


myfirst java programm
master
My_Programs
My_Programs.java
public class My_Programs { new *
public static void main(String[] args) { new *
// Get the initial state of the thread
Thread.State initialState = thread.getState();
System.out.println("Initial State: " + initialState);
Run
My_Programs
"C:\Program Files\Java\jdk-22\bin\java.exe" "-javaagent:C:\Radhe\IntelliJ IDEA Community Edition 2024.1\lib\idea_rt.jar=59892:C:\Radhe\IntelliJ IDEA Community
Initial State: NEW
State after starting: RUNNABLE
Thread is running...
Thread has completed.
Final State: TERMINATED
Process finished with exit code 0
}
}
myfirst java programm › src › My_Programs › main
28:43 CRLF UTF-8 4 spaces
34°C
Sunny
Search
ENG
IN
12:48
23-04-2024

Viva Question:

Q.1 What is Thread.

Ans:-The thread is a program execution that uses process resources for accomplishing the task. All threads within a single program are logically contained within a process. The kernel allocates a stack and a thread control block (TCB) to each thread. The operating system saves only the stack pointer and CPU state at the time of switching between the threads of the same process.

Q.2 What is Multithreading and Multitasking.

Ans:- Multitasking:

Multitasking refers to the ability of a computer system to execute multiple tasks or processes concurrently. These tasks can be running simultaneously, or they can be rapidly switched between by the operating system, giving the appearance of simultaneous execution.

Multitasking allows a computer to serve multiple users or applications concurrently, improving efficiency and resource utilization.

There are two main types of multitasking: preemptive multitasking, where the operating system controls task switching, and cooperative multitasking, where tasks voluntarily yield control to other tasks.

Multithreading:

Multithreading is a specific technique within multitasking where a single process or program is divided into multiple execution threads. Each thread represents a separate flow of control within the process, capable of executing independently.

Multithreading allows for concurrent execution of different parts of a program, enabling tasks to be performed in parallel and making more efficient use of available resources, such as CPU cores.

Threads within the same process share memory space and resources, allowing them to communicate and synchronize with each other easily.

Java and many other programming languages provide built-in support for multithreading, making it relatively easy to develop concurrent applications.

IPS ACADEMY

16 Collages, 71 Courses, 51 Acre Campus

ISO 9001: 2008 Certified

Knowledge Village Rajendra Nagar

A. B. Road Indore 452012(M.P.) India

Ph: 0731-4014601-604 Mo: 07746000161

E Mail: ies.ipsacademy.org &

Website: www.ipsacademy.org