Vidyavardhaka College of Engineering

Gokulam III stage, Mysuru – 570 002
Autonomous Institute under Visvesvaraya Technological University (VTU)
Accredited by NBA (2020- 2023) & NAAC with 'A' Grade (2018 - 2023)

COURSE SYLLABUS

SEMESTER-III

Course Name: Advanced Java Programming	**Course Code: BISAJ317**
No. of Lecture hours / week: 00	**CIE Marks: 50**
No. of Tutorial hours / week: 04	**SEE Marks: 50**
Total No. of Lecture+ Tutorial/Practical hours 50	**SEE Duration: 02hr**
L: T: P: 0:0:4	**Credits: 02**

Course Prerequisites: Basic understanding of Java programming with OOPS.

Course Overview: The focus of this course is on design and implementation of advanced java concepts through hands on experience to develop real world applications.

Course Learning Objectives (CLO)
This course will enable students to,

- Familiarize advanced features of Java.
- Develop core java applications using multiple threads, JDBC, JSP, Applets and Servlets.

PART A : INTRODUCTION

Following are Basic core Java programs.

1. Write a Java program that prompts the user for an integer N and generates all the prime numbers up to N.
2. Write a Java program to create a class box with instance variable width, height, depth and create an object using default constructors and parameterized constructors.
3. Write a Java program for adding two numbers using method overloading.
4. Write a Java program to convert an integer 257 to byte using narrowing type conversion and widening type conversion.
5. Write a Java program to sort the string elements in a 1-dimensional array.
6. Write a Java program to sum all the elements of an array using for-each version of for loop.
7. Create a Java class Customer with the following details as variables within it: CustID, Name, Age, Phone, Place. Write a Java program to create n Customers objects and print the CustID, Name, Age, Phone and Place of these objects with suitable headings using "this" keyword.
8. Design a super class called Employee with details as EmpID, Name, Phone, Salary, extend this class by writing two subclasses namely Tester (ProjectID, ProjectName), Developer (ProjectName). Write a Java Program to read and display at least 2 Employee objects of all two categories using Inheritance.

PART B – Advanced Concepts

1. Develop a Java program to create an interface named Shape that contains a method named printArea(). Provide three classes named Rectangle, Triangle and Circle such that each one of the classes extends the interface Shape. Each one of the classes contains only the method printArea() that prints the area of the given shape.
2. Create a class Book which contains four members: name, author, price, num of pages. Include a constructor to set the values for the members. Include methods to set and get the details of the objects. Include a toString() method that could display the complete details of the book. Develop a Java program to create n book objects.
3. Write a program that demonstrates handling of exceptions in inheritance. Create a base class called "Father" and derived class called "Son" which extends the base class. In Father class, implement a constructor which takes the age and throws the exception Wrong Age () i.e when the input age is equal to father's age.
4. Write a program which creates two threads, one thread displaying "Vidyavardhaka College of Engineering" once every ten seconds and another displaying "CSE" once every two seconds.
5. Write a program that sorts an array of strings using compareTo() to determine bubble sort ordering.
6. Develop a java program to create an enum as session and demonstrate the usage of value(), valueOf() and ordinal() methods.
7. Write a Java program to implement the SQL commands using JDBC.
8. Write a JSP program which shows a Sample Order Form.

PART C - Open Ended Experiments

Students shall solve a problem (either given by the staff or students may come up with their own problem) using the design techniques.

1. Develop JSP program which displays the System date and time.
2. Develop a Swings program to display image on the button.
3. Develop Session Handling using servlets.
4. Develop Chat Server using Java.

PART A: INTRODUCTION

1. Write a Java program that prompts the user for an integer N and generates all the prime numbers up to N.

```java
import java.util.Scanner;

public class PrimeNumbers {
    public static void main(String[] args) {
        Scanner scanner = new Scanner(System.in);

        // Prompt user for input
        System.out.print("Enter an integer N: ");
        int N = scanner.nextInt();

        System.out.println("Prime numbers up to " + N + ":");

        // Iterate through numbers from 2 to N
        for (int i = 2; i <= N; i++) {
            boolean isPrime = true;

            // Check if the number i is prime
            for (int j = 2; j <= Math.sqrt(i); j++) {
                if (i % j == 0) {
                    isPrime = false;
                    break;
                }
            }

            // If i is prime, print it
            if (isPrime) {
                System.out.print(i + " ");
            }
        }

        // Close the scanner
        scanner.close();
    }
}
```

OUTPUT:

```
Enter an integer N: 10
Prime numbers up to 10:
2 3 5 7
```

2. Write a Java program to create a class box with instance variable width, height, depth and create an object using default constructors and parameterized constructors.

```java
class Box {
    double width;
    double height;
    double depth;

    // Default constructor
    public Box() {
        width = 1.0;
        height = 1.0;
        depth = 1.0;
    }

    // Parameterized constructor
    public Box(double width, double height, double depth) {
        this.width = width;
        this.height = height;
        this.depth = depth;
    }

    // Method to calculate and return the volume of the box
    public double calculateVolume() {
        return width * height * depth;
    }
}

public class BoxDemo {
    public static void main(String[] args) {
        // Creating objects using default constructor
        Box box1 = new Box();
        System.out.println("Box 1 - Default Constructor");
        System.out.println("Width: " + box1.width);
        System.out.println("Height: " + box1.height);
        System.out.println("Depth: " + box1.depth);
        System.out.println("Volume: " + box1.calculateVolume());
        System.out.println();
        // Creating objects using parameterized constructor
        Box box2 = new Box(2.5, 3.0, 4.0);
        System.out.println("Box 2 - Parameterized Constructor");
        System.out.println("Width: " + box2.width);
        System.out.println("Height: " + box2.height);
        System.out.println("Depth: " + box2.depth);
        System.out.println("Volume: " + box2.calculateVolume());
    }
}
```

OUTPUT:

```
Box 1 - Default Constructor
Width: 1.0
Height: 1.0
Depth: 1.0
Volume: 1.0

Box 2 - Parameterized Constructor
Width: 2.5
Height: 3.0
Depth: 4.0
Volume: 30.0
```

3. Write a Java program for adding two numbers using method overloading.

```java
public class AddNumbers {
    // Method to add two integers
    public int add(int num1, int num2) {
        return num1 + num2;
    }

    // Method to add two doubles
    public double add(double num1, double num2) {
        return num1 + num2;
    }

    // Method to add three integers
    public int add(int num1, int num2, int num3) {
        return num1 + num2 + num3;
    }

    public static void main(String[] args) {
        AddNumbers adder = new AddNumbers();

        // Using the first method (integers)
        int sum1 = adder.add(5, 10);
        System.out.println("Sum of 5 and 10 is: " + sum1);

        // Using the second method (doubles)
        double sum2 = adder.add(3.5, 2.7);
        System.out.println("Sum of 3.5 and 2.7 is: " + sum2);

        // Using the third method (integers)
        int sum3 = adder.add(2, 4, 6);
        System.out.println("Sum of 2, 4, and 6 is: " + sum3);
    }
}
```

OUTPUT:

```
Sum of 5 and 10 is: 15
Sum of 3.5 and 2.7 is: 6.2
Sum of 2, 4, and 6 is: 12
```

4. Write a Java program to convert an integer 257 to byte using narrowing type conversion and widening type conversion.

```
public class TypeConversion {
    public static void main(String[] args) {
        int intValue = 257;

        // Narrowing type conversion (explicit casting)
        byte narrowedByteValue = (byte) intValue; // Explicitly casting int to
        byte

        // Widening type conversion (implicit casting)
        //byte widenedByteValue = (byte) intValue; // Implicitly casting int
        to byte
        double widenedByteValue = intValue;

        System.out.println("Original Int Value: " + intValue);
        System.out.println("Narrowed Byte Value (Explicit Casting): " +
        narrowedByteValue);
        System.out.println("Widened Byte Value (Implicit Casting): " +
        widenedByteValue);
    }
}
```

OUTPUT:

```
Original Int Value: 257
Narrowed Byte Value (Explicit Casting): 1
Widened Byte Value (Implicit Casting): 257.0
```

5. Write a Java program to sort the string elements in a 1-dimensional array.

```
import java.util.Arrays;

public class Sort {
    public static void main(String[] args) {
        String[] stringArray = {"Apple", "Orange", "Banana", "Grape",
        "Pineapple"};

        // Sorting the string array
        Arrays.sort(stringArray);

        // Displaying the sorted array
        System.out.println("Sorted String Array:");
        //for (String element : stringArray)
        //    System.out.println(element);
        for(int i=0; i<stringArray.length;i++)
           System.out.println(stringArray[i]);

    }
}
```

OUTPUT:

```
Sorted String Array:
Apple
Banana
Grape
Orange
Pineapple
```

6. Write a Java program to sum all the elements of an array using for-each version of for loop.

```
public class Sum {
    public static void main(String[] args) {
        int[] numbers = {1, 2, 3, 4, 5};
        // Variable to store the sum
        int sum = 0;

        // Calculate the sum using for-each loop
        for (int number : numbers) {
            sum += number;
        }

        // Display the sum
        System.out.println("Sum of the elements in the array: " + sum);
    }
}
```

OUTPUT:

```
Sum of the elements in the array: 15
```

7. Create a Java class Customer with the following details as variables within it: CustID, Name, Age, Phone, Place. Write a Java program to create n Customers objects and print the CustID, Name, Age, Phone and Place of these objects with suitable headings using "this" keyword.

```
class Customer {
    private int custID;
    private String name;
    private int age;
    private String phone;
    private String place;

    // Constructor to initialize the Customer object
    public Customer(int custID, String name, int age, String phone, String
     place) {
        this.custID = custID;
        this.name = name;
        this.age = age;
        this.phone = phone;
        this.place = place;
    }

    // Method to display customer details
    public void displayDetails() {
        System.out.println("Customer ID: " + this.custID);
        System.out.println("Name: " + this.name);
        System.out.println("Age: " + this.age);
        System.out.println("Phone: " + this.phone);
        System.out.println("Place: " + this.place);
        System.out.println("------------------------------");
    }
}

public class CustomerDetails {
    public static void main(String[] args) {
        int n = 3; // Number of Customer objects to create

        // Creating an array of Customer objects
        Customer[] customers = new Customer[n];

        // Initializing Customer objects and displaying details
        customers[0] = new Customer(1, "Alice", 25, "1234567890", "New York");
        customers[1] = new Customer(2, "Bob", 30, "9876543210", "London");
        customers[2] = new Customer(3, "Charlie", 28, "5551234567", "Sydney");

        // Displaying customer details using for-each loop
        for (Customer customer : customers) {
            customer.displayDetails();
        }
    }
}
```

OUTPUT:

```
Customer ID: 1
Name: Alice
Age: 25
Phone: 1234567890
Place: New York
------------------------------
Customer ID: 2
Name: Bob
Age: 30
Phone: 9876543210
Place: London
------------------------------
Customer ID: 3
Name: Charlie
Age: 28
Phone: 5551234567
Place: Sydney
------------------------------
```

8. Design a super class called Employee with details as EmpID, Name, Phone, Salary, extend this class by writing two subclasses namely Tester (ProjectID, ProjectName), Developer (ProjectName). Write a Java Program to read and display at least 2 Employee objects of all two categories using Inheritance.

```java
//Superclass Employee
class Employee {
   int empID;
   String name;
   String phone;
   double salary;

 // Constructor to initialize Employee object
 public Employee(int empID, String name, String phone, double salary) {
     this.empID = empID;
     this.name = name;
     this.phone = phone;
     this.salary = salary;
 }

 // Method to display employee details
 public void displayDetails() {
     System.out.println("Employee ID: " + empID);
     System.out.println("Name: " + name);
     System.out.println("Phone: " + phone);
     System.out.println("Salary: $" + salary);
 }
}

//Subclass Tester extending Employee
class Tester extends Employee {
 private int projectID;
 private String projectName;

 // Constructor to initialize Tester object
 public Tester(int empID, String name, String phone, double salary, int
projectID, String projectName) {
     super(empID, name, phone, salary);
     this.projectID = projectID;
     this.projectName = projectName;
 }

 // Method to display tester details
 @Override
 public void displayDetails() {
     super.displayDetails();
     System.out.println("Project ID: " + projectID);
     System.out.println("Project Name: " + projectName);
     System.out.println("------------------------------");
 }
}

//Subclass Developer extending Employee
class Developer extends Employee {
 private String projectName;

 // Constructor to initialize Developer object
 public Developer(int empID, String name, String phone, double salary, String
 projectName) {
     super(empID, name, phone, salary);
     this.projectName = projectName;
 }

 // Method to display developer details
 @Override
 public void displayDetails() {
     super.displayDetails();
     System.out.println("Project Name: " + projectName);
```

```
      System.out.println("------------------------------");
 }
}

public class OverrideDemo {
   public static void main(String[] args) {
     // Creating Tester objects
      Tester tester1 = new Tester(1, "Alice", "1234567890", 60000, 101,
     "ProjectA");
     Tester tester2 = new Tester(2, "Bob", "9876543210", 65000, 102,
     "ProjectB");

     // Creating Developer objects
     Developer developer1 = new Developer(3, "Charlie", "5551234567", 70000,
     "ProjectC");
     Developer developer2 = new Developer(4, "David", "9998887776", 75000,
     "ProjectD");

     // Displaying details of Tester and Developer objects
     System.out.println("Tester 1 Details:");
     tester1.displayDetails();
     System.out.println("Tester 2 Details:");
     tester2.displayDetails();
     System.out.println("Developer 1 Details:");
     developer1.displayDetails();
     System.out.println("Developer 2 Details:");
     developer2.displayDetails();
 }
}
```

OUTPUT:

```
Tester 1 Details:
Employee ID: 1
Name: Alice
Phone: 1234567890
Salary: $60000.0
Project ID: 101
Project Name: ProjectA
------------------------------
Tester 2 Details:
Employee ID: 2
Name: Bob
Phone: 9876543210
Salary: $65000.0
```

```
Project ID: 102
Project Name: ProjectB
------------------------------
Developer 1 Details:
Employee ID: 3
Name: Charlie
Phone: 5551234567
Salary: $70000.0
Project Name: ProjectC
------------------------------
Developer 2 Details:
Employee ID: 4
Name: David
Phone: 9998887776
Salary: $75000.0
Project Name: ProjectD
------------------------------
```

PART B: ADVANCED CONCEPTS

1. Develop a Java program to create an interface named Shape that contains a method named printArea(). Provide three classes named Rectangle, Triangle and Circle such that each one of the classes extends the interface Shape. Each one of the classes contains only the method printArea() that prints the area of the given shape.

```java
//Interface Shape
interface Shape {
 void printArea();
}

//Rectangle class implementing Shape interface
class Rectangle implements Shape {
 private double length;
 private double width;

 // Constructor for Rectangle class
 public Rectangle(double length, double width) {
     this.length = length;
     this.width = width;
 }

 // Implementation of printArea() method for Rectangle
 @Override
 public void printArea() {
     double area = length * width;
```

```
      System.out.println("Area of Rectangle: " + area);
  }
}

//Triangle class implementing Shape interface
class Triangle implements Shape {
  private double base;
  private double height;

  // Constructor for Triangle class
  public Triangle(double base, double height) {
      this.base = base;
      this.height = height;
  }

  // Implementation of printArea() method for Triangle @Override
  public void printArea() {
      double area = 0.5 * base * height;
      System.out.println("Area of Triangle: " + area);
  }
}
//Circle class implementing Shape interface
class Circle implements Shape {
  private double radius;

  // Constructor for Circle class
  public Circle(double radius) {
      this.radius = radius;
  }

  // Implementation of printArea() method for Circle  @Override
  public void printArea() {
      double area = Math.PI * radius * radius;
      System.out.println("Area of Circle: " + area);
  }
}

public class InterfaceDemo {
  public static void main(String[] args) {
      // Creating objects of Rectangle, Triangle, and Circle
      Rectangle rectangle = new Rectangle(5, 10);
      Triangle triangle = new Triangle(6, 8);
      Circle circle = new Circle(4);

      // Printing areas of shapes using printArea() method
      rectangle.printArea();
      triangle.printArea();
```

```
        circle.printArea();
    }
}
```

OUTPUT:

```
Area of Rectangle: 50.0
Area of Triangle: 24.0
Area of Circle: 50.26548245743669
```

2. Create a class Book which contains four members: name, author, price, num of pages. Include a constructor to set the values for the members. Include methods to set and get the details of the objects. Include a toString() method that could display the complete details of the book. Develop a Java program to create n book objects.

```
class Book1 {
    private String name;
    private String author;
    private double price;
    private int numOfPages;

    // Getter and setter methods for name
    public String getName() {
        return name;
    }
    public void setName(String name) {
        this.name = name;
    }
    // Getter and setter methods for author
    public String getAuthor() {
        return author;
    }
    public void setAuthor(String author) {
        this.author = author;
    }

    // Getter and setter methods for price
    public double getPrice() {
        return price;
    }
    public void setPrice(double price) {
        this.price = price;
    }

    // Getter and setter methods for number of pages
    public int getNumOfPages() {
        return numOfPages;
    }
    public void setNumOfPages(int numOfPages) {
        this.numOfPages = numOfPages;
    }

    // toString() method to display complete details of the book @Override
    public String toString() {
        return "Book Details: \n" +
               "Name: " + name + "\n" +
               "Author: " + author + "\n" +
               "Price: $" + price + "\n" +
               "Number of Pages: " + numOfPages;
    }
}

public class BookDemo1 {
    public static void main(String[] args) {
        // Number of book objects to create
        int n = 2;
        // Creating an array of Book objects
        Book1[] books = new Book1[n];

        // Initializing Book objects with details
        books[0] = new Book1();
        books[0].setName("Java Programming");
        books[0].setAuthor("John Doe");
        books[0].setPrice(29.99);
        books[0].setNumOfPages(400);
        books[1] = new Book1();
        books[1].setName("Data Structures and Algorithms");
        books[1].setAuthor("Jane Smith");
        books[1].setPrice(39.95);
        books[1].setNumOfPages(550);

        System.out.println("Book Details using getter methods");
        for (int i = 0; i < n; i++) {
             System.out.println("Book " + (i+1));
             System.out.println(books[i].getName());
             System.out.println(books[i].getAuthor());
             System.out.println(books[i].getPrice());
             System.out.println(books[i].getNumOfPages());
        }

        // Displaying details of Book objects
        System.out.println("\nBook Details using toString()");
        for (int i = 0; i < n; i++) {
             System.out.println(books[i].toString());
        }
    }
}
```

OUTPUT:

```
Book Details using getter methods
Book 1
Java Programming
John Doe
29.99
400
Book 2
Data Structures and Algorithms
Jane Smith
39.95
550

Book Details using toString()
Book Details:
Name: Java Programming
Author: John Doe
Price: $29.99
Number of Pages: 400
Book Details:
Name: Data Structures and Algorithms
Author: Jane Smith
Price: $39.95
Number of Pages: 550
```

3. Write a program that demonstrates handling of exceptions in inheritance. Create a base class called "Father" and derived class called "Son" which extends the base class. In Father class, implement a constructor which takes the age and throws the exception Wrong Age () i.e when the input age is equal to father's age.

```java
import java.util.Scanner;
class Father
{
      int Fage;
      Scanner input = new Scanner(System.in);
      Father()
      {
            System.out.println("Enter father's age:");
            Fage=input.nextInt();

        }
}
class Son extends Father
{
    int Sage;
    Scanner input = new Scanner(System.in);
     Son()
     {
            //super();
            System.out.println("Enter son's age:");
            Sage=input.nextInt();
     }
}

class WrongAgeException extends Exception
{
    public WrongAgeException(String str) {
            System.out.println(str);
     }
}

class ExceptionDemo
{
     public static void main(String args[]) throws WrongAgeException
     {
            Son s=new Son();
            try {
                  if(s.Sage>=s.Fage)
                  throw new WrongAgeException("Exception:");
                 else
                  System.out.println("You have entered a Valid Age");
        }
            catch(WrongAgeException e) {
                  System.out.println(e + " SON'S AGE >= FATHER'S AGE");
            }
     }
}
```

OUTPUT1:

```
Enter father's age:
45
Enter son's age:
20
```

```
You have entered a Valid Age
```

OUTPUT2:

```
Enter father's age:
20
Enter son's age:
30
WrongAgeException: Exception: SON'S AGE >= FATHER'S AGE
```

4. Write a program which creates two threads, one thread displaying "Vidyavardhaka College of Engineering" once every ten seconds and another displaying "CSE" once every two seconds.

5.

```
class CollegeNameThread extends Thread {
     public void run() {
        while (true) {
            System.out.println("Vidyavardhaka College of Engineering");
            try {
                Thread.sleep(10000); // Sleep for 10 seconds
            } catch (InterruptedException e) {
                System.out.println(e);
            }
        }
    }
}

class DepartmentThread extends Thread {
    public void run() {
        while (true) {
            System.out.println("CSE");
            try {
                Thread.sleep(2000); // Sleep for 2 seconds
            } catch (InterruptedException e) {
                e.printStackTrace();
            }
        }
    }
}

public class ThreadDemo {
    public static void main(String[] args) {
        // Creating threads
        CollegeNameThread collegeNameThread = new CollegeNameThread();
        DepartmentThread departmentThread = new DepartmentThread();

        // Starting threads
        collegeNameThread.start();
        departmentThread.start();
    }
}
```

OUTPUT:

```
Vidyavardhaka College of Engineering
CSE
CSE
CSE
CSE
CSE
Vidyavardhaka College of Engineering
CSE
CSE
CSE
CSE
CSE
Vidyavardhaka College of Engineering
CSE
CSE
CSE
CSE
CSE
```

6. Write a program that sorts an array of strings using compareTo() to determine bubble sort ordering.

```
public class BubbleSort {
    public static void main(String[] args) {
        String[] stringArray = {"Apple", "Orange", "Banana", "Grape",
        "Pineapple"};

        // Sorting the string array using Bubble Sort
        int n = stringArray.length;
        for (int i = 0; i < n - 1; i++) {
            for (int j = 0; j < n - i - 1; j++) {
                if (stringArray[j].compareTo(stringArray[j + 1]) > 0) {
                    // Swap stringArray[j] and stringArray[j + 1]
                    String temp = stringArray[j];
                    stringArray[j] = stringArray[j + 1];
                    stringArray[j + 1] = temp;
                }
            }
        }

        // Displaying the sorted array
        System.out.println("Sorted String Array:");
        for (String element : stringArray) {
            System.out.println(element);
        }
    }
}
```

OUTPUT:

```
Sorted String Array:
Apple
Banana
Grape
Orange
Pineapple
```

7. Develop a java program to create an enum as session and demonstrate the usage of value(), valueOf() and ordinal() methods.

```
//Enum representing different session types
enum Session {
 MORNING, AFTERNOON, EVENING, NIGHT
}
public class EnumDemo {
 public static void main(String[] args) {
     // Using values() method to get all enum constants
     Session[] sessions = Session.values();
     System.out.println("All Enum Constants:");
     for (Session session : sessions) {
         System.out.println(session);
     }

     // Using valueOf() method to get enum constant by name
     String sessionName = "MORNING";
     Session session = Session.valueOf(sessionName); // Returns
      Session.MORNING
     System.out.println("\nEnum Constant for Name '" + sessionName + "': " +
      session);

     // Using ordinal() method to get the position/index of enum constant
     int ordinal = session.ordinal(); // Returns 0 for Session.MORNING
     System.out.println("\nPosition of Enum Constant " + session + ": " +
      ordinal);
 }
}
```

OUTPUT:

```
All Enum Constants:
MORNING
AFTERNOON
EVENING
NIGHT

Enum Constant for Name 'MORNING': MORNING

Position of Enum Constant MORNING: 0
```

8. Write a Java program to implement the SQL commands using JDBC.

```java
import java.sql.*;
public class Database {
	public static void main(String[] args)  {
		try{
			Class.forName("com.mysql.cj.jdbc.Driver");
			System.out.println("Driver loaded successfully");
			Connection con=DriverManager.getConnection("jdbc:mysql:"
					+ "//localhost:3306/student","root","root@123");
			System.out.println("Connection established sucessfully"+con);
			Statement stmt=con.createStatement();
			stmt.executeUpdate("insert into student (Name, USN, Sem) "
					+ "values ('Kennedy','4VV21CS001',3)");
			ResultSet rs=stmt.executeQuery("select * from student");
			System.out.println("Name  \tUSN  \t\tSem");
			while(rs.next())
				System.out.println(rs.getString(1)+"\t"+
					rs.getString(2)+"\t"+rs.getInt(3));
			con.close();
		}
		catch(Exception e) {
			System.out.println(e);
		}
	}
}
```

OUTPUT:

```
Driver loaded successfully
```

```
Conenction established
sucessfully...com.mysql.cj.jdbc.ConnectionImpl@7d9d0818
Name       USN           Sem
Kennedy    4VV21CS001    3
Alice      4VV21IS001    4
Bob  4VV21IS002    3
Charlie    4VV21IS003    3
Danny      4VV21IS004    4
Esha 4VV21IS005    4
```

9. Write a JSP program which shows a Sample Order Form.

file.jsp

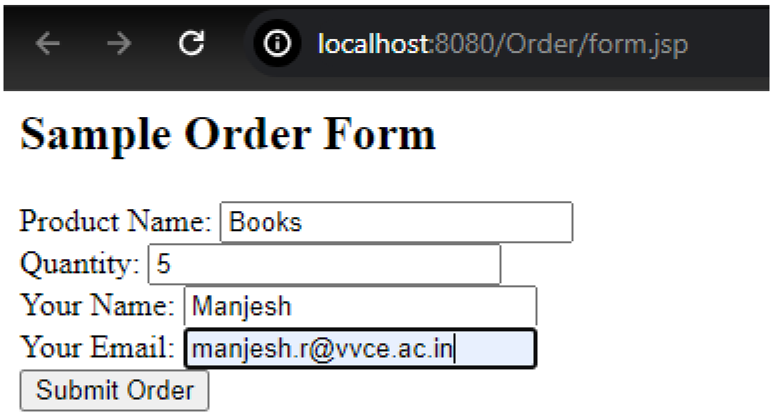
```
<%@ page language="java" contentType="text/html; charset=ISO-8859-1"
pageEncoding="ISO-8859-1"%>
<html>
<head>
    <title>Sample Order Form</title>
</head>
<body>
    <h2>Sample Order Form</h2>
    <form method="post" action="./final.html">
        <label for="productName">Product Name:</label>
        <input type="text" id="productName" name="productName" required>
        <br>
        <label for="quantity">Quantity:</label>
        <input type="number" id="quantity" name="quantity" required>
        <br>
        <label for="customerName">Your Name:</label>
        <input type="text" id="customerName" name="customerName" required>
        <br>
        <label for="email">Your Email:</label>
        <input type="email" id="email" name="email" required>
        <br>
        <input type="submit" value="Submit Order">
    </form>
</body>
</html>
```

final.html

```
<!DOCTYPE html>
<html>
<head>
     <meta charset="ISO-8859-1">
     <title>Insert title here</title>
</head>
<body>
```

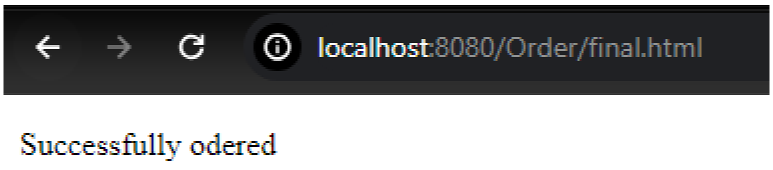
```
    <p>Successfully odered</p>
</body>
</html>
```

OUTPUT:


localhost:8080/Order/form.jsp

Sample Order Form

Product Name: Books
Quantity: 5
Your Name: Manjesh
Your Email: manjesh.r@vvce.ac.in


localhost:8080/Order/final.html

Successfully odered

APPENDIX

1. How to connect Java program with MySQL database?

   Step 1: https://dev.mysql.com/downloads/installer/

   i. Download and install mysql server or enterprise edition and J connector.
   ii. Note the port number(default:3306), username and password during installation.

   Step 2: Open mysql command line from start. -->Enter password(entered during installation).

   i. Create a database.
   ii. use database.
   iii. create a table.
   iv. Enter 1 or more rows in table.

   Step 3: Open Eclipse.

   i. Create a new project.
   ii. Right click on project--> Build path-->configure build path-->libraries-->add external jars.
   iii. Select the connector (jar file) from location of mysql installation. (Eg:- C:\Program files(x86)\Mysql\..)
   iv. Apply & close.

2. How to execute JSP pages on Web Server?

   Step 1: Create a Dynamic Web Project

   1. Open Eclipse and go to `File -> New -> Dynamic Web Project`.
   2. Enter a project name and select web module version as 4.0 and click `Finish`.

   Step 2: Write JSP Code

   1. Inside the project created right-click on the `webapp` folder under the main, under src folder, go to `New -> JSP File`.
   2. Enter a file name (e.g., `index.jsp`) and click `Finish`.

   Step 3: Install the Web Server

   1. Under the Server tab below the Eclipse -> Click on No Servers Available
   2. Select Apache-> Select Tomcat version 10.0 server.
   3. Click on Download and Install, Select download folder.
   4. After installation, Add all web projects from left column to right column, click Finish.

   Step 4: Run the Project

   1. Right-click on your project in Eclipse.
   2. Select Run As -> Run on Server.
   3. Choose your server (e.g., Apache Tomcat) and click Finish.

Experimental Weightage:

Type of Experiment	Program -No	Weightage
Exercise	1, 2	10%
Structure Enquiry	3, 7, 8	37%
Demonstration	4, 5, 6	37%
Open ended		10%

Course Outcomes:

CO1	**Apply and demonstrate the usage of various programming constructs**
CO2	**Design and Develop applications to solve problems across various technical and real-world domains**
CO3	**Explore various programming tools and techniques**

CO-PO-PSO Mapping

CO	**PO**												**PSO**		
	PO1	**PO2**	**PO3**	**PO4**	**PO5**	**PO6**	**PO7**	**PO8**	**PO9**	**PO10**	**PO11**	**PO12**	**PSO1**	**PSO2**	**PSO3**
CO1	3														3
CO2			3												3
CO3					2				2						2
AVG.	**3**		**3**		**2**				**2**						**2.66**