Visvesvaraya Technological University
"Jnana Sangama", Belagavi-590018

****

Third Semester B.E.

[As per Choice Based Credit System (CBCS) Scheme]

(For Internal Circulation Only)

Integrated Professional Core Course (IPCC)
OOP with Java (BCS306A)
Lab Manual

(For Reference Only)

Name	
USN	
Section	
Lab Batch	
Day /Time	

****

Kalpataru Institute of Technology, Tiptur - 572 201

Department of Artificial Intelligence and Machine Learning

AY: 2024-2025

VISVESVARAYA TECHNOLOGICAL UNIVERSITY, BELAGAVI

Integrated Professional Core Course (IPCC)

OOP with Java

LAB MANUAL

BCS306A

III Semester

KALPATARU INSTITUTE OF TECHNOLOGY, TIPTUR

DEPARTMENT OF ARTIFICIAL INTELLIGENCE AND

MACHINE LEARNING

2024-25

CONTENTS

KALPATARU INSTITUTE OF TECHNOLOGY

(Accredited by NBA, Approved by A.I.C.T.E. New Delhi, Recognized by Govt. of Karnataka & Affiliated to V.T U., Belagavi)

DEPARTMENT OF ARTIFICIAL INTELLIGENCE AND MACHINE LEARNING

Vision and Mission of the Institution

Vision

"To bring forth technical graduates of high caliber with a strong character and to uphold the spiritual and cultural values of our country."

Mission

"To impart quality technical and managerial education at graduate and post graduate levels through our dedicated and well qualified faculty."

Vision and Mission of the AI & ML Department

Vision

"To create a community of AI and ML specialists distinguished by their technical excellence and moral principles, who champion responsible innovation and honor our country's spiritual and cultural traditions, advancing technology for the betterment of society."

Mission

M1: "To advance AI and ML through ground-breaking development, encouraging students and faculty to pursue innovative solutions that address global challenges and drive societal progress."

M2: "To provide a supportive and inclusive environment that nurtures the intellectual and personal growth of our students, preparing them to become leaders in AI&ML."

M3: "To integrate the nation's spiritual and cultural values into the AI and ML curriculum, fostering respect for our heritage and guiding students to consider the cultural impacts of their innovations."

KALPATARU INSTITUTE OF TECHNOLOGY

(Accredited by NBA, Approved by A.I.C.T.E. New Delhi, Recognized by Govt. of Karnataka & Affiliated to V.T U., Belagavi)

DEPARTMENT OF ARTIFICIAL INTELLIGENCE AND MACHINE LEARNING

Program Outcomes	
a.	**Engineering Knowledge:** Apply knowledge of mathematics, science, engineering fundamentals and an engineering specialization to the solution of complex engineering problems.
b.	**Problem Analysis: Identify,** formulate, research literature and analyze complex engineering problems reaching substantiated conclusions using first principles of mathematics, natural sciences and engineering sciences
c.	**Design/ Development of Solutions**: Design solutions for complex engineering problems and design system components or processes that meet specified needs with appropriate consideration for public health and safety, cultural, societal and environmental considerations.
d.	**Conduct investigations of complex problems** using research-based knowledge and research methods including design of experiments, analysis and interpretation of data and synthesis of information to provide valid conclusions.
e.	**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 under- standing of the limitations.
f.	**The Engineer and Society**: Apply reasoning informed by contextual knowledge to assess societal, health, safety, legal and cultural issues and the Consequent responsibilities relevant to professional engineering practice.
g.	**Environment and Sustainability**: Understand the impact of professional Engineering solutions in societal and environmental contexts and demonstrate knowledge of and need for sustainable development.
h.	**Ethics**: Apply ethical principles and commit to professional ethics and Responsibilities and norms of engineering practice.
i.	**Individual and Team Work:** Function effectively as an individual, and as amember or leader in diverse teams and in multi disciplinary settings.
j.	**Communication:** Communicate effectively on complex engineering activities withthe 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**.**
k.	**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.
l.	**Project Management and Finance:** Demonstrate knowledge and understanding of 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.
Program Specific Outcomes	
m.	**PSO1**: The ability to comprehend, analyse, and apply knowledge of human cognition, Artificial Intelligence, Machine Learning, and data engineering to real-world problems, addressing future challenges.
n.	**PSO2**: The ability to cultivate computational knowledge and project development skills, utilizing innovative tools and techniques to address problems in Deep Learning, Machine Learning, and Artificial Intelligence.

1. Develop a JAVA program to add TWO matrices of suitable order N (The value of N should be read from command line arguments).

Save Filename As: Addition

Solution:-

```
import java.util.Scanner;
class Addition
{
  public static void main(String args[])
  {
   int row, column, i, j;
   Scanner in = new Scanner(System.in);
   System.out.println("Enter the number of rows of matrix: ");
   row = in.nextInt();
   System.out.println("Enter the number of columns of matrix: ");
   column = in.nextInt();
   if(row==column)
   {
   int matrixA[][] = new int[row][column];
   int matrixB[][] = new int[row][column];
   int sumMatrix[][] = new int[row][column];

   System.out.println("Enter the elements of first matrix: ");
   for (i = 0 ; i <row ; i++ )
     for (j = 0 ; j <column ; j++ )
       matrixA[i][j] = in.nextInt();
   System.out.println("Enter the elements of second matrix: ");
```

```
  for (i = 0 ; i <row ; i++ )
    for (j = 0 ; j <column ; j++ )
      matrixB[i][j] = in.nextInt();

  System.out.println("Sum of Matrix");
  for (i = 0 ; i <row ; i++ )
    for (j = 0 ; j <column ; j++ )
      sumMatrix[i][j] =  matrixA[i][j] + matrixB[i][j];

  for (i = 0 ; i <row; i++ )
  {
    for (j = 0 ; j <column; j++ )
    {
      System.out.print(sumMatrix[i][j]+"\t");
    }
    System.out.print("\n");
   }
  }
  else
      {
          System.out.println("Addition not possible");
          System.out.println("Try Again");
      }
 }
}
```

Output

```
Enter the number of rows of matrix:
2
Enter the number of columns of matrix:
2
Enter the elements of first matrix:
1
2
3
4
Enter the elements of second matrix:
1
2
3
4
Sum of Matrix
2  4
6  8
```

2. Develop a stack class to hold a maximum of 10 integers with suitable methods. Develop a JAVA main method to illustrate Stack operations

Save Filename As: stackoperation

Solution:-

```
import java.util.Scanner;
class stack
{
int top=-1;
int size=10;
int a[]=new int[size];
void push(int value)
{
if(top == (size-1))
System.out.println("\nStack is Full!!! Insertion is not
possible!!!");
else
{
top++;
a[top] = value;
System.out.println("\nInsertion success!!!");
}
}
void pop()
{
if(top == -1)
System.out.println("\nStack is Empty!!! Deletion is not possible!!!");
```

```
else
{
System.out.println("\nDeleted :"+a[top]);
top--;
}
}
void display()
{
if(top == -1)
System.out.println("\nStack is Empty!!!");
else
{
System.out.println("\nStack elements are:\n");
for(int i=top; i>=0; i--)
System.out.println(a[i]);
}
}
}
class stackoperation
{
public static void main(String[] args)
{
stack1 s=new stack1();
Scanner sc=new Scanner(System.in);
int choice;
```

```
while(true)
{
System.out.println("\n\n***** MENU *****\n");
System.out.println("1. Push\n2. Pop\n3. Display\n4. Exit");
System.out.println("\nEnter your choice: ");
choice=sc.nextInt();
switch(choice)
{
case 1: System.out.println("enter data");
        int value=sc.nextInt();
        s.push(value);
        break;
case 2: s.pop();
        break;
case 3: s.display();
        break;
case 4: System.exit(0);
default: System.out.println("\nWrong selection!!! Try again!!!");
}
}
}
}
```

Output

***** MENU *****

1. Push
2. Pop
3. Display
4. Exit

Enter your choice: 1

enter data 10

Insertion success!!!

***** MENU *****

1. Push
2. Pop
3. Display
4. Exit

Enter your choice: 1

enter data 20

Insertion success!!!

***** MENU *****

1. Push
2. Pop
3. Display
4. Exit

Enter your choice: 1

enter data 30

Insertion success!!!

```
***** MENU *****
1. Push
2. Pop
3. Display
4. Exit
Enter your choice: 3
Stack elements are:
30
20
10

***** MENU *****
1. Push
2. Pop
3. Display
4. Exit
Enter your choice: 2
Deleted :30

**** MENU *****
1. Push
2. Pop
3. Display
4. Exit
Enter your choice: 2
Deleted :20
```

```
***** MENU *****
1. Push
2. Pop
3. Display
4. Exit
Enter your choice: 2
Deleted :10

***** MENU *****
1. Push
2. Pop
3. Display
4. Exit
Enter your choice: 2
Stack is Empty!!! Deletion is not possible!!!
***** MENU *****
1. Push
2. Pop
3. Display
4. Exit
Enter your choice: 3
Stack is Empty!!!
***** MENU *****
1. Push
2. Pop
3. Display
4. Exit
Enter your choice:
4
```

3. A class called Employee, which models an employee with an ID, name and salary, is designed as shown in the following class diagram. The method raiseSalary (percent) increases the salary by the given percentage. Develop the Employee class and suitable main method for demonstration.

Save Filename as as: EmployeeMain

Solution:-

```
import java.util.Scanner;
class Employee
{
int id;
String name;
double salary;
Employee (int id, String name, double salary)
{
this.id = id;
this.name = name;
this.salary = salary;
}
int getId ()
{
return id;
}
String getName ()
{
return name;
}
double getSalary ()
{
return salary;   }
```

```
void raiseSalary (double percent)
{
salary += salary * percent / 100.0;
}
}
class EmployeeMain
{
public static void main (String[] args)
{
Scanner scanner = new Scanner (System.in);
System.out.println ("Enter Employee ID:");
int id = scanner.nextInt ();
System.out.println ("Enter Employee Name:");
scanner.nextLine (); // Consume newline left-over
String name = scanner.nextLine ();
System.out.println ("Enter Employee Salary:");
double salary = scanner.nextDouble ();
Employee emp = new Employee (id, name, salary);
System.out.println ("Employee ID: " + emp.getId ());
System.out.println ("Employee Name: " + emp.getName ());
System.out.println ("Employee Salary: " + emp.getSalary ());
System.out.println ("Enter raise percentage:");
double percent = scanner.nextDouble ();
emp.raiseSalary (percent);
System.out.println ("Employee Salary after raise: " +emp.getSalary ());
scanner.close ();
}
}
```

Output

Enter Employee ID: 1

Enter Employee Name: naveen

Enter Employee Salary: 5000

Employee ID: 1

Employee Name: naveen

Employee Salary: 5000.0

Enter raise percentage: 10

Employee Salary after raise: 5500.0

4. A class called MyPoint, which models a 2D point with x and y coordinates, is designed as follows:Two instance variables x (int) and y (int).

- **A default (or "no-arg") constructor that construct a point at the default location of (0, 0).**
- **A overloaded constructor that constructs a point with the given x and y coordinates.**
- **A method setXY() to set both x and y.**
- **A method getXY() which returns the x and y in a 2-element int array.**
- **A toString() method that returns a string description of the instance in the format "(x, y)".**
- **A method called distance(int x, int y) that returns the distance from this point to another point at the given (x, y) coordinates**
- **An overloaded distance(MyPoint another) that returns the distance from this point to the given MyPoint instance (called another)**
- **Another overloaded distance() method that returns the distance from this point to the origin (0,0).**
- **Develop the code for the class MyPoint. Also develop a JAVA program (called TestMyPoint) to test all the methods defined in the class.**

Save Filename As: TestMyPoint.java

Solution:-

```
class MyPoint
 {
   private int x;
   private int y;
   // Default constructor
   public MyPoint()
  {
      this.x = 0;
      this.y = 0;
   }
```

```
// Overloaded constructor
public MyPoint(int x, int y)
 {
    this.x = x;
    this.y = y;
 }
// Set both x and y
public void setXY(int x, int y)
  {
    this.x = x;
    this.y = y;
}
// Get x and y in a 2-element int array
public int[] getXY()
{
    return new int[]{x, y};
}
// Return a string description of the instance in the format "(x, y)"
public String toString()
{
    return "(" + x + ", " + y + ")";
}
// Calculate distance from this point to another point at (x, y) coordinates
public double distance(int x, int y)
{
    int xDiff = this.x - x;
    int yDiff = this.y - y;
    return Math.sqrt(xDiff * xDiff + yDiff * yDiff);
 }
```

```
    // Calculate distance from this point to another MyPoint instance (another)
    public double distance(MyPoint another)
   {
        return distance(another.x, another.y);
    }
    // Calculate distance from this point to the origin (0,0)
    public double distance()
   {
        return distance(0, 0);
    }
}

public class TestMyPoint   {
    public static void main(String[] args)   {
        // Creating MyPoint objects using different constructors
        MyPoint point1 = new MyPoint();
        MyPoint point2 = new MyPoint(3, 4);
        // Testing setXY and getXY methods
        point1.setXY(1, 2);
        System.out.println("Point1 coordinates after setXY: " +
point1.getXY()[0] + ", " + point1.getXY()[1]);
        // Testing toString method
       System.out.println("Point2 coordinates: " + point2.toString());
        // Testing distance methods
System.out.println("Distance from Point1 to Point2: " + point1.distance(point2));
System.out.println("Distance from Point2 to Origin: " + point2.distance());
    }
}
```

Output

Point1 coordinates after setXY: 1, 2

Point2 coordinates: (3, 4)

Distance from Point1 to Point2: 2.8284271247461903

Distance from Point2 to Origin: 5.0

5. Develop a JAVA program to create a class named shape. Create three sub classes namely: circle, triangle and square, each class has two member functions named draw () and erase (). Demonstrate polymorphism concepts by developing suitable methods, defining member data and main program

Save Filename as: ShapeMain

Solution:-

```
class Shape
{
void draw ()
{
System.out.println ("Drawing a shape");
}
void erase ()
{
System.out.println ("Erasing a shape");
}
}

class Circle extends Shape
{
void draw ()
{
System.out.println ("Drawing a circle");
}
void erase ()
{
System.out.println ("Erasing a circle");
}
}
```

```
class Triangle extends Shape
{
void draw ()
{
System.out.println ("Drawing a triangle");
}
void erase ()
{
   System.out.println ("Erasing a triangle");
}
}

class Square extends Shape
{
void draw ()
{
System.out.println ("Drawing a square");
}
void erase ()
{
System.out.println ("Erasing a square");
}
}

class ShapeMain
{
public static void main (String[] args)
{
Shape circle = new Circle();
```

```
Shape triangle = new Triangle();
Shape  square=new Square();
circle.draw();
circle.erase();
triangle.draw();
triangle.erase();
square.draw();
square.erase();
}
}
```

Output

Drawing a circle

Erasing a circle

Drawing a triangle

Erasing a triangle

Drawing a square

Erasing a square

6. Develop a JAVA program to create an abstract class Shape with abstract methods calculateArea() and calculatePerimeter(). Create subclasses Circle and Triangle that extend the Shape class and implement the respective methods to calculate the area and perimeter of each shape

Save Filename as: Shape_peri_area_Main

Solution:-

```
abstract class Shape
{
// Abstract methods to calculate area and perimeter
public abstract double calculateArea ();
public abstract double calculatePerimeter ();
}
class Circle extends Shape
{
double radius;

Circle (double radius)
{
this.radius = radius;
}
public double calculateArea ()
{
return Math.PI * radius * radius;
}
public double calculatePerimeter ()
{
return 2 * Math.PI * radius;
}
}
```

```
class Triangle extends Shape
{
double side1;
double side2;
double side3;
Triangle (double side1, double side2, double side3)
{
this.side1 = side1;
this.side2 = side2;
this.side3 = side3;
}
public double  calculateArea ()
{
double s = (side1 + side2 + side3) / 2;
return Math.sqrt(s * (s - side1) * (s - side2) * (s - side3));
}
public double calculatePerimeter ()
{
return side1 + side2 + side3;
}
}

class Shape_peri_area_Main
{
public static void main (String[] args)
{
Circle circle = new Circle (5.0);
Triangle triangle = new Triangle (3.0, 4.0, 5.0);
System.out.println ("Circle:");
```

```
System.out.println ("Area: " + circle.calculateArea ());
System.out.println ("Perimeter: " + circle.calculatePerimeter());
System.out.println ();
System.out.println ("Triangle:");
System.out.println ("Area: " + triangle.calculateArea ());
System.out.println ("Perimeter: " + triangle.calculatePerimeter());
}
}
```

Output

```
Circle:
Area: 78.53981633974483
Perimeter: 31.41592653589793

Triangle:
Area: 6.0
Perimeter: 12.0
```

7. Develop a JAVA program to create an interface Resizable with methods resizeWidth(int width) and resizeHeight(int height) that allow an object to be resized. Create a class Rectangle that implements the Resizable interface and implements the resize methods.

Save Filename as: rect_resize

Solution:-

```
interface Resizable
{
void resizeWidth (int width);
void resizeHeight (int height);
}

class Rectangle implements Resizable
{
int width;
int height;
Rectangle (int width, int height)
{
this.width = width;
this.height = height;
}
public void resizeWidth (int width)
{
this.width = width;
}
public void resizeHeight (int height)
{
this.height = height;
}
```

```
// Additional method to get the dimensions
void getDimensions ()
{
System.out.println ("Width: " + width + ", Height: " + height);
}
}

class rect_resize {
public static void main (String[] args)
{
Rectangle rectangle = new Rectangle (5, 7);
System.out.println ("Original Dimensions:");
rectangle.getDimensions ();
rectangle.resizeWidth (8);
rectangle.resizeHeight (10);
System.out.println ("\nResized Dimensions:");
rectangle.getDimensions ();
}
}
```

Output

Original Dimensions:

Width: 5, Height: 7

Resized Dimensions:

Width: 8, Height: 10

8. Develop a JAVA program to create an outer class with a function display. Create another class inside the outer class named inner with a function called display and call the two functions in the main class.

Save Filename As: InnerOuter

Solution:-

```
class Outer
{
void display()
{
System.out.println("Outer display");
}
class Inner
{
void display()
{
System.out.println("Inner display");
}
}    }

class InnerOuter   {
public static void main(String[] args)   {
Outer outer = new Outer();
Outer.Inner inner = outer.new Inner();
outer.display(); // Calls the outer class display() function
inner.display(); // Calls the inner class display() function
}   }
```

Output

Outer display

Inner display

9. Develop a JAVA program to raise a custom exception (user defined exception) for DivisionByZero using try, catch, throw and finally

Save Filename As: CustomException

Solution:-

```
class DivisionByZeroException extends Exception
{
DivisionByZeroException (String message)
{
super(message);
}
}
class CustomException
 {
public static void main (String[] args)
 {
int dividend = 10;
int divisor = 0;
try
 {
if (divisor == 0)
{
throw new DivisionByZeroException("Cannot divide by zero");
}
int result = dividend / divisor;
System.out.println ("Result: " + result);
}
catch (DivisionByZeroException e)
{
System.out.println ("Exception: " + e.getMessage ());
```

```
}
finally
{
System.out.println ("Finally block executed");
 }
}
}
```

Output

Exception: Cannot divide by zero

Finally block executed

10. Develop a JAVA program to create a package named mypack and import & implement it in a suitable class

Step 1: Create a Package

Create a directory named mypack (or any name you prefer) on your file system.

Inside the mypack directory, create a Java file named MyClass.java.

Step 2: Define the Class in the Package. Open MyClass.java and add the following code:

```
package mypack;
public class MyClass
{
public void
display ()
{
System.out.println("This is a method from the mypack package.");
}
}
```

Step 3: Create a Main Program

Create a new Java file (outside the mypack directory) named MainProgram.java and add the following code:

```
import mypack.MyClass;
public class MainProgram  {
public static void main (String[] args)  {
MyClass obj = new MyClass ();
obj.display ();
}
}
```

Step 4: Compile and Run

Use Shift+F6

Output

This is a method from the mypack package.

11. Write a program to illustrate creation of threads using runnable class. (start method start each of the newly created thread. Inside the run method there is sleep() for suspend the thread for 500 milliseconds)

Save Filename as: ThreadMain

Solution:-

```
class MyRunnable implements Runnable
{
public void run()
{
try
{
for(int i=0;i<5;i++)
{
System.out.println("naveen");
Thread.sleep(500);
}
}
catch (InterruptedException e)
{
System.out.println("Thread interrupted: " +e);
}
}
}
class ThreadMain
{
public static void main(String[] args)
{
Thread thread = new Thread(new MyRunnable());
thread.start();
```

```
}
}
```

Output

```
naveen
naveen
naveen
naveen
naveen
```

12. Develop a program to create a class MyThread in this class a constructor, call the base class constructor, using super and start the thread. The run method of the class starts after this. It can be observed that both main thread and created child thread are executed concurrently.

Save Filename as: ThreadConstructor.java

Solution:-

```
class MyThread extends Thread
{
MyThread()
{
// Calling the base class constructor using super
super("Using Thread class");
System.out.println("Child thread: " + this);
// Starting the thread
start();
}
public void run()
{
 try
{
 for (int i = 5; i > 0; i--)
{
 System.out.println("Child thread " + i);
  Thread.sleep(1000);
   }
  }
 catch (InterruptedException e)
 {
  System.out.println("Child thread interrupted");
```

```
 }
 System.out.println("Exiting child thread...");
 }
}

public class Bases
{
public static void main(String args[])
{
// Creating an instance of MyThread, which will start a new thread
new MyThread();
try
{
for (int k = 5; k > 0; k--)
{
System.out.println("Running main thread: " + k);
Thread.sleep(2000);
}
}
catch (InterruptedException e)
{
System.out.println("Main thread interrupted");
}
System.out.println("Exiting main thread...");
}
}
```

Output

```
Child thread: Thread[Using Thread class,5,main]
Running main thread: 5
Child thread 5
Child thread 4
Running main thread: 4
Child thread 3
Child thread 2
Running main thread: 3
Child thread 1
Exiting child thread...
Running main thread: 2
Running main thread: 1
Exiting main thread...
```