Page No:
Date:

NAME : UDHAYARAJAN J.
SUB : MCA-I
DATE : 2025-11-22
CLASS : JAVA PROGRAMMING.
ROLLNO : 2202561
TYPE : JAVA PROGRAMMING PRACTICAL LAB.

INDEX.

Name : UDHAYARAJAN. J
Sub : MCA - 1
Date : 2025-11-17
Class : JAVA PROGRAMMING.
Regno : 2202561
Type : JAVA PROGRAMMING LAB PRACTICAL.

Introduction of arrays and different operation on arrays design an application by using arrays.

① Finding the Largest / Smallest Element.

```
public class FindMinMax {
    public static void main(String[] args) {
        int arr[] = {12, 45, 9, 78, 23, 5};
        int max = arr[0];
        int min = arr[0];
        for (int i=1; arr.length > i; i++) {
            if (arr[i] > max) {
                max = arr[i];
            }
            if (arr[i] < min) {
                min = arr[i];
            }
        }
        System.out.println("Largest Element " : + max);
        System.out.println("Smallest Element: " + min);
    }
}
```

Output:
Largest Element : 78.
Smallest Element : 5.

Page No:
Date.

② Reversing an array:

```
public class ReverseArray {
    public static void main(String[] args) {
        int[] original = {1, 2, 3, 4, 5};
        int[] reversedArray = new int[original.length];
        for(int i=0; i<original.length; i++) {
            reversedArray[i] = original[original.length-1-i];
        }
        System.out.println("Original Array: " + java.util.Arrays.
                toString(original));
        System.out.println("Reversed Array: " + java.util.Arrays.
                toString(reversedArray));
    }
}
```

Output:
Original Array: [1, 2, 3, 4, 5]
Reversed Array: [5, 4, 3, 2, 1].

③ Sum of all elements in a 2D Array.

```
public class SumOf2DArrayElements {
    public static void main(String[] args) {
        int[][] numbers = {{10, 20, 30}, {40, 50, 60}};
        int sum = 0;
        for(int i=0; i<numbers.length; i++) {
            for(int j=0; j<numbers[i].length; j++) {
                sum += numbers[i][j];
            }
        }
        System.out.println("Sum of all elements: " + sum);
    }
}
```

Output:
Sum of all Elements: 210.

④. Sum of each elements row of 2D array.

```
public class SumEachRow2D {
    public static void main(String[] args) {
        int[][] matrix = {{10,20,30},{40,50,60},
                          {70,80,90}};
        for(int i=0; i<matrix.length;i++) {
            int rowSum = 0;
            for (int j=0; j<matrix[i].length; j++) {
                rowsum += matrix[i][j];
            }
            System.out.println("Sum of row" + i + ":" +rowSum);
        }
    }
}
```

output:
Sum of row 0 : 60.
Sum of row 1 : 150.
Sum of row 2 : 240.

⑤ package.

write a program for implementation of package to create class and method in mymath function. package.

1. Create user defined package.

```
package MyMathFunctions;
public class MathUtils {
    public static double findSqrt(double number) {
        return math.sqrt(number);
    }
```

```
    public static double findPower(double base, double
                                   exponent) {
        return Math.pow(base, exponent);
    }
}
```

2. Importing

```
import MyMathFunctions.MathUtils;
public class MainApp {
    public static void main(String[] args) {
        double num = 25.0;
        double base = 2.0;
        double exponent = 3.0;
        double sqrtResult = MathUtils.findSqrt(num);
        double powerResult = MathUtils.findPower(base, exponent
        System.out.println("the Square root of " + num + " is "
                           sqrtResult);
        System.out.println("base + " raised to the power of "
                           + exponent + " is " + powerResult);
    }
}
```

output:
the square root of 25.0 is: 5.0.
2.0 raised to the power of 3.0 is: 8.

⑥ Abstract class Expression for calculating areas.

```
abstract class Shape {
    public abstract void calculateArea();
}
```

```
class Rectangle extends Shape {
    private double length;
    private double width;
    public Rectangle(double length, double width) {
        this.length = length;
        this.width = width;
    }
    @Override
    public void calculateArea() {
        double area = length * width;
        System.out.println("Area of Rectangle: " + area);
    }
}

public class AreaCalculator {
    public static void main(String[] args) {
        Rectangle rectangle = new Rectangle(10, 5);
        rectangle.calculateArea();
    }
}
```

output:
Area of Rectangle: 50.0

④ Interface program for calculating product amount.

```
interface AmountCalculate {
    double calTotalAmount(double price, int quantity);
}

class product implements AmountCalculate {
    public double calTotalAmount(double price, int quantity) {
        return price * quantity;
    }
}
```

```
public class productAmt {
    public static void main(String[] args) {
        AmountCalculate p = new product();
        double totalAmount = p.calTotalAmount(15.50, 5);
        System.out.println("Total amount for standard
                            product : $" + String.format
                            ("%.2f", totalAmount);
    }
}
```

output:
Total Amount for standard product : $ 77.50.

⑥ Introduction to string and its different operations.

use of different string method.

```
public class StrMain {
    public static void main(String[] args) {
        String firstName = "John";
        String lastName = "Doe";
        String fullName = firstName + " " + lastName;
        String fullMessage = "welcome".concat(fullname);
        String s1 = "hello";
        String s2 = "hello";
        boolean isEqual = s1.Equals(s2);
        String s = "this is a test.";
        System.out.println("original:" + s);
        String upper = s.toUpperCase();
        String lower = s.toLowerCase();
        System.out.println("Upper case:" + upper);
        System.out.println("Lower case:" + lower);
        System.out.println("full name:" + fullmessage);
        System.out.println("Equal or not" + isEqual);
        System.out.println("length of string s =" + s.length);
    }
}
```

output :
original : This is a test.
Upper case : THIS IS A TEST.
Lower case : this is a test.
Full name : welcom John Doe.
Equal or not true.
Length of the string s = 15.

⑨. Design application using String, String Builder, string Tokenizer Experiment.

```
import java.util.Scanner;
import java.util.StringTokenizer;
public class StringApp {
    public static void main(String[] args) {
        Scanner scanner = new Scanner(System.in);
        System.out.println("Enter a sentence");
        String originalSentence = scanner.nextLine();
        System.out.println("original sentence :" + originalSentence);

        StringBuilder modifiedSentenceBuilder = new
                StringBuilder(originalSentence);
        System.Out.println("Enter text to Append");
        String textToAppend = scanner.nextLine();
        modifiedSentenceBuilder.append(" ").append(
                textToAppend);
        System.out.println("modified sentence (appended) :" +
                modifiedSentenceBuilder.toString());
        System.out.println("Enter text to insert at index 5:");
        String textToInsert = scanner.nextLine();
        if(modifiedSentenceBuilder.length() >= 5) {
            modifiedSentenceBuilder.insert(5," " +
                    textToInsert + " ");
        }
```

```
        else {
            System.out.println("Sentence too short to insert
                                at index 5");
        }
        String finalModifiedSentence = modifiedSentenseBuilder.
                                toString();
        System.out.println("Modified sentence (inserted): " +
                            finalModifiedSentence);

        System.out.println("Tokenizing the modified
                            sentence");
        StringTokenizer tokenizer = new StringTokenizer
                            (finalModifiedSentence, ".,!?;");
        int tokenCount = 0;
        while (tokenizer.hasMoreToken()) {
            String token = tokenizer.nextToken();
            System.out.println("Token:" + token);
            tokenCount++;
        }
        System.out.println("Total tokens found: " +
                            tokenCount);
        Scanner.close();
    }
}
```

Output:

```
Enter a sentence : tynny strone.
original sentence : tynny strone.
Enter text to Append : one.
modified sentence (appended) : tynny strone one.
Enter text to insert at index 5 : t.
modified sentence (inserted) : tynny t strone one t.
```

Tokenizing the modified Sentence:
Token : tynny.
Token : t.
Token : Strone.
Token : one.
Total Tokens found : 4.

(10) Exception handling.

Test any five of standard exception and user defined custom Exception in Java.

Exception for array out of bound index.

```
public class MainArray {
    public static void main(String[] args) {
        try {
            int[] number = {1, 2, 3};
            System.out.println(number[5]);
        }
        catch(ArrayIndexOutOfBoundException e) {
            System.out.println("Error: Array index out of
                                Bound!");
            System.out.println("Exception message : " +
                                e.getMessage());
        }
        System.out.println("program continous after
                            exception handling");
    }
}
```

output:
Error: Array index out of bounds!
Exception message : Index 5 out of bounds for length 3.
Program continous after exception handling.

(11) Try - catch - finally block with multiple catch statements :

```
public class ExceptionHandling {
    public static void main(String[] args) {
        try {
            int res = 10/0;
            String s = null;
            System.out.println(s.length());
        }
        catch (ArithmeticException e) {
            System.out.println("caught Arithmetic
                Exception : " + e);
        }
        catch (NullPointerException e) {
            System.out.println("caught NullPointer
                exception : " + e);
        }
        finally {
            System.out.println("this code in the
                finally block always executes.");
        }
        System.out.println("program continues after
            exception handling");
    }
}
```

output :

```
Caught ArithmeticException: java.lang.ArithmeticException: / by
                                                    zero.
This code in the finally block always executes.
program continues after exception handling.
```

(12) User defined Exception.

```
public class ThrowExample {
    static void checkEligibility(int stuage, int stuweight){
        if (stuage > 12 && stuweight > 40) {
            System.out.println("Student entry is valid");
        }
        else {
            throw new ArithmeticException("Student is
                not eligible for registration");
        }
    }
    public static void main(String[] args) {
        System.out.println("welcome to the registration
                process!");
        checkEligibility(10, 39);
        System.out.println("Have a nice day...");
    }
}
```

output:

```
welcome to the registration process!
Exception in thread "main" java.lang.ArithmeticException:
        Student is not eligible for registration
    at ThrowExample.checkEligibility(ThrowExample.java:4)
    at ThrowExample.main(ThrowExample.java:12)
```

(13) Multi-threading.

implementing a thread. - 1. Extending the thread class.

```
class CookingTask extends Thread {
    private String task;
    CookingTask(String data) {
        this.task = data;
    }
    public void run() {
        String data = this.task;
        System.out.println(data + " is being prepared
                by " + Thread.currentThread().
                getName());
    }
}

public class Restaurant {
    public Restaurant() { }
    public static void main(String[] args) {
        CookingTask var1 = new CookingTask("pasta");
        CookingTask var2 = new CookingTask("Salad");
        CookingTask var3 = new CookingTask("Dessert");
        CookingTask var4 = new CookingTask("Rice");
        var1.start();
        var2.start();
        var3.start();
        var4.start();
    }
}
```

output:

Dessert is being prepared by Thread-2.
pasta is being prepared by Thread-0.
Rice is being prepared by Thread-3.
Salad is being prepared by Thread-1

(14) Implement Thread using the Runnable Interface.

```
public class MyRunnableTask implements Runnable {
    public void run() {
        System.out.println("Thread " + Thread.currentThread().getName() + " is running.");
        for(int i = 0; i < 5; i++) {
            System.out.println(Thread.currentThread().getName() + ": " + i);
            try {
                Thread.Sleep(100);
            }
            catch(InterruptedException e) {
                Thread.currentThread().interrupt();
            }
        }
    }
}
public class MainRunnableTask {
    public static void main(String[] args) {
        Thread T1 = new Thread(new MyRunnableTask(), "Thread-1");
        Thread T2 = new Thread(new MyRunnableTask(), "Thread-2");
        Thread T3 = new Thread(new MyRunnableTask(), "Thread-3");
        Thread T4 = new Thread(new MyRunnableTask(), "Thread-4");
        T1.start();
        T2.start();
        T3.start();
        T4.start();
    }
}
```

Page No:
Date:

Output:

```
Thread-4 is running.
Thread-1 is running.
Thread-2 is running.
Thread-3 is running.
Thread-4:0
Thread-2:0
Thread-3:0
Thread-1:0
Thread-1:1
Thread-2:1
.....
.....
.....
```

⑮ Collections.

Add or remove element using ArrayList.

```
import java.util.*;
class ArrayExample4
    public static void main(String[] args) {
        ArrayList<String> arrList = new ArrayList<String>();
        arrList.add("Steve");
        arrList.add("Tim");
        arrList.add("Lucy");
        arrList.add("Pat");
        arrList.add("Angela");
        arrList.add("Tom");
        System.out.println(arrList);
        arrList.add(3, "Steve");
        System.out.println(arrList);
        arrList.Remove(3);
        System.out.println(arrList);
    }
}
```

output:
[Steve, Tim, Lucy, pat, Angela, TOM].
[Steve, Tim, Lucy, steve, pat, Angela, TOM].
[Steve, Tim, Lucy, pat, Angela, TOM].

(16). To find maximum element vector using predefined method.

```
import java.io.*;
import java.util.collections;
import java.util.vector;
public class vectorMain {
    public static void main(String[] args) {
        vector<Integer> v=new vector<Integer>();
        v.add(7);
        v.add(50);
        v.add(0);
        v.add(67);
        v.add(98);
        int n= collection.max(v);
        System.out.println("the maximum value
                present in vector is :" + n);
    }
}
```

output:
The Maximum value present in vector is : 98.

(17) To get the element of LinkedList:

```
import java.io.*;
import java.util.linkelist;
public class LinkedListmain {
    public static void main(String[] args) {
        LinkedList<String> gfg=new LinkedList<String>();
```

```
        gtg.add("India");
        gtg.add("USA");
        gtg.add("U.K.");
        for(int i=0; i<gtg.size(); i++) {
            System.out.println("Element at index "
                +i+" is : "+gtg.get(i));
        }
    }
}
```

output:

Element at index 0 is : India.
Element at index 1 is : USA
Element at index 2 is : UK.

(18) To get the element of HashSet.

```
import java.util.HashSet;
public class HashSetExample {
    public static void main(String[] args) {
        HashSet<String> fruits = new HashSet<String>();
        fruits.add("Apple");
        fruits.add("Banana");
        fruits.add("Orange");
        fruits.add("Apple");
        System.out.println("Fruits in HashSet : "+fruits);
        System.out.println("Contains 'Banana' ? "+
                fruits.contains("Banana"));
        System.out.println("Contains 'Grape' ? "+fruits.
                contains("Grape"));
        fruits.remove("Orange");
        System.out.println("Fruits after removing
                Orange : "+fruits);
        System.out.println("Iterating through fruits:");
```

```
        for(String fruit : fruits) {
            System.out.println(fruit);
        }
        System.out.println("Size of HashSet" : +
                            fruits.size());
        fruits.clear();
        System.out.println("Fruits after clearing"
                            + fruits);
    }
}
```

output:

Fruits in Hashset : [Apple, orange, Banana]
Contains 'Banana' ? true.
Contains 'Grape' ? false.
Fruits after removing orange : [Apple, Banana]
Iterating through fruits.
Apple
Banana.
Size of HashSet : 2.
Fruits after clearing : []

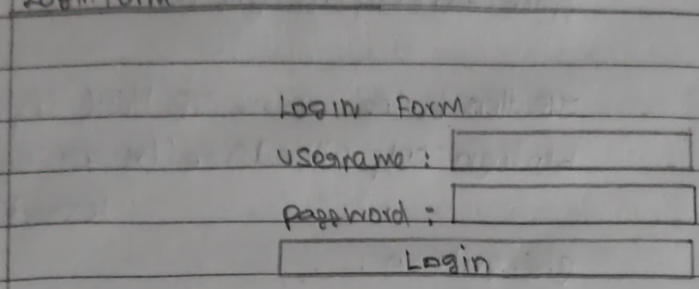
19) write a program for Login page using swing.

```
import javax.swing.*;
import java.awt.*;
public class Loginpage {
    public static void main(String[] args) {
        JFrame frame = new JFrame("Login Form");
        frame.setDefaultCloseOperation(JFrame.EXIT_ON_
                                        CLOSE);

        frame.setSize(600,400);
        frame.setLocationRelativeTo(null);
        frame.setLayout(new GridBagLayout());

        GridBagConstraints gbc = new GridBagConstraints();
        gbc.insets = new insets(8,8,8,8);
        gbc.anchor = GridBagConstrainst.WEST;

        JLabel title = new JLabel("Login Form");
        title.setFont(new Font("Arial", FONT.BOLD,20));
        gbc.gridx = 0;
        gbc.gridy = 0;
        gbc.gridwidth = 2;
        gbc.anchor = GridBagConstraints.WEST;
        frame.add(title, gbc);

        gbc.gridx = 0;
        gbc.gridy = 1;
        frame.add(new JLabel("Username :"), gbc);

        JTextField txtUser = new JTextField(18);
        gbc.gridx = 1;
        frame.add(txtUser, gbc);
```

```
gbc.gridx = 0;
gbc.gridy = 2;
frame.add(new JLabel("password :"), gbc);

JpasswordField txtpass = new JpasswordField(12);
gbc.gridx = 1;
frame.add(txtpass, gbc);

JButton btnLogin = new JButton("Login");
btnLogin.setpreffeoodSize(txtUser.getPrefferodSize());

gbc.gridx = 0;
gbc.gridy = 3;
gbc.gridwidth = 2;
gbc.fill = GridBagConstraints.HORIZONTAL;
frame.add(btnLogin, gbc);

btnLogin.addActionListener(e -> {
        String user = txtUser.getText();
        String pass = new String(txtpass.getpassword());
        if(user.equals("admin") && pass.equals("123")) {
            JOptionpane.showMessageDialog(frame, "Login successfully !",
                "Message", JOptionpane.INFORMATION_MESSAGE);
        }
        else {
            JOptionpane.showMessageDialog(frame,
                "Login Failed", "Message", JOptionpane.ERROR_MESSAGE);
        }
});
```

```
        frame.setVisible(true);
    }
}
```

output:



20) write a program for Simple Calculator using swing:

```
import javax.swing.*;
import java.awt.*;
import java.awt.event.*;

public class SimpleCalculator extends JFrame implements
                                    ActionListener {
    JTextField t1, t2, result;
    JButton add, sub, div, mul;

    public SimpleCalculator() {
        setTitle("Simple Calculator");
        SetSize(400, 300);
        setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
```

```
setLayout(new GridBagLayout());
GridBagConstraints gbc = new GridBagConstraints();
gbc.insets = new Insets(10,10,10,10);
gbc.fill = GridBagConstraints.HORIZONTAL;

gbc.gridx=0;
gbc.gridy=0;
add(new JLabel("Number 1:"), gbc);

t1=new JTextField(15);
gbc.gridx=1;
add(t1,gbc);

gbc.gridx=0;
gbc.gridy=1;
add(new JLabel("Number 2:"), gbc);

t2=new JTextField(15);
gbc.gridx=1;
add(t2,gbc);

gbc.gridx=0;
gbc.gridy=2;
add(new JLabel("Result:"), gbc);

result=new JTextField(15);
result.setEditable(false);
gbc.gridx=1;
add(result, gbc);

Jpanel btnpanel=new Jpanel(new GridLayout
                           (1,4,10,10));
add=new JButton("Add");
sub=new JButton("Subtract");
mul=new JButton("Multiply");
```

```
div = new JButton("Divide");

btnpanel.add(add);
btnpanel.add(sub);
btnpanel.add(mul);
btnpanel.add(div);

add.addActionListener(this);
sub.addActionListener(this);
mul.addActionListener(this);
div.addActionListener(this);

gbc.gridx = 0;
gbc.gridy = 3;
gbc.gridwidth = 2;
add(btnpanel, gbc);

setVisible(true);
}
@Override
public void actionPerformed(ActionEvent e) {
    try {
        double n1 = Double.parseDouble(t1.getText());
        double n2 = Double.parseDouble(t2.getText());
        double ref = 0;
        if(e.getSource() == add)
            ref = n1+n2;
        else if(e.getSource() == sub)
            ref = n1-n2;
        else if(e.getSource() == mul)
            ref = n1*n2;
        else if(e.getSource() == div)
            ref = n1/n2;

        result.setText(String.valueOf(ref));
```

```
        }
        catch (Exception e) {
            JoptionPane.showMessageDialog(this,
                    "please Enter numbers");
        }.
    }.

    public static void main(String [] args) {
        new SimpleCalculator();
    }
}.
```

output :