|  |  |
| --- | --- |
| **Ex.No:** | **Implementation of Classes, Constructors and Methods** |
| **Date:** |

**AIM:**

To write a program to implementation of Class, Constructors and Methods.

**ALGORITHM:**

STEP 1: Start the program.

STEP 2: Create a class Employee

STEP 3: Inside the class Employee, create and variable baseSalary and a Construtor to assign the baseSalary that give in the main class

STEP 4: In the class, there is a method calculateSalary to add 20% of baseSalary to the baseSalary.

STEP 5: In the main class, create an object for the class and pass the baseSalary to the Constructor.

STEP 6: Then print the employees calculateSalary for all the employees.

STEP 7: Stop the program.

**PROGRAM:**

class Employee {

private double baseSalary;

public Employee(double baseSalary) {

this.baseSalary = baseSalary;

}

public double calculateSalary() {

return baseSalary + 0.2 \* baseSalary;

}

}

public class prac1 {

public static void main(String[] args) {

Employee emp1 = new Employee(50000);

Employee emp2 = new Employee(40000);

Employee emp3 = new Employee(60000);

Employee emp4 = new Employee(30000);

Employee emp5 = new Employee(20000);

System.out.println("1nd Employee salary adding 20% bonus:"+emp1.calculateSalary());

System.out.println("2nd Employee salary adding 20% bonus:"+emp2.calculateSalary());

System.out.println("3rd Employee salary adding 20% bonus:"+emp3.calculateSalary());

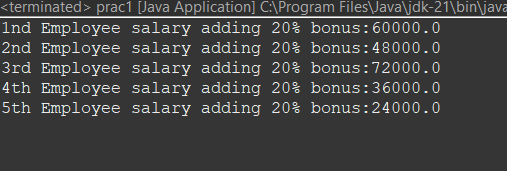
System.out.println("4th Employee salary adding 20% bonus:"+emp4.calculateSalary());

System.out.println("5th Employee salary adding 20% bonus:"+emp5.calculateSalary());

}

}

**OUTPUT:**

****

**RESULT:**

|  |  |
| --- | --- |
| **Ex.No:** | **Implementation of Hierarchical Inheritance** |
| **Date:** |

**AIM:**

To write a java program to implement hierarchical inheritance.

**ALGORITHM:**

STEP 1: Start the program.

STEP 2: Define a superclass with common properties and behaviours that you want to

share among multiple subclasses.

STEP 3: Define subclasses that inherit from the superclass. Each subclass will have its

own unique properties and behaviours in addition to those inherited from the

superclass.

STEP 4: If necessary, override methods from the superclass in the subclasses to

provide specialized behaviour.

STEP 5: Create objects of the subclasses and use them to access both inherited and

subclass-specific methods.

STEP 6: Stop the program.

**PROGRAM:**

class Animal {

void eat() {

System.out.println("Animal is eating...");

}

}

class Dog extends Animal {

void bark() {

System.out.println("Dog is barking...");

}

}

class Cat extends Animal {

void meow() {

System.out.println("Cat is meowing...");

}

}

public class Main {

public static void main(String[] args) {

Dog dog = new Dog();

Cat cat = new Cat();

dog.eat();

dog.bark();

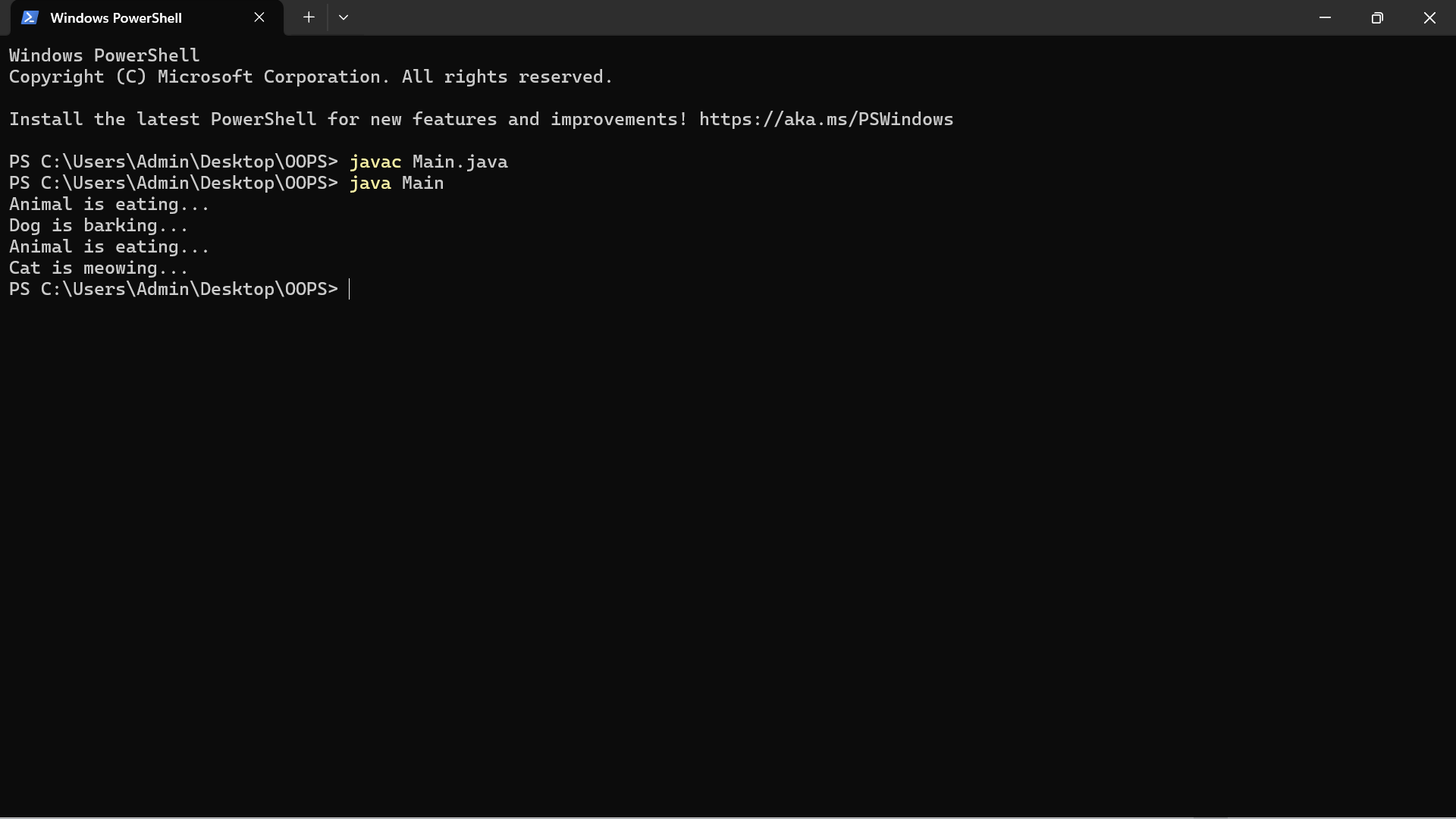
cat.eat();

cat.meow();

}

}

**OUTPUT :**

****

**RESULT:**

|  |  |
| --- | --- |
| **Ex.No:** | **Implementation of Multilevel Inheritance** |
| **Date:** |

**AIM:**

To write java program to implement multilevel inheritance.

**ALGORITHM:**

STEP 1: Create a class name as shape and method name as display

STEP 2: Create a rectangle class and inheritance the Shape class

STEP 3: Create a class name as Cube and inheritance the rectangle class and create method to display the volume

STEP 4: In main class to create an object as a class Cube It will call a method use this object

STEP 5: Stop the program

**PROGRAM:**

class Shape {

public void display() {

System.out.println("Inside display");

}

}

class Rectangle extends Shape {

public void area() {

System.out.println("Inside area");

}

}

class Cube extends Rectangle {

public void volume() {

System.out.println("Inside volume");

}

}

public class Tester {

public static void main(String[] arguments) {

Cube cube = new Cube();

cube.display();

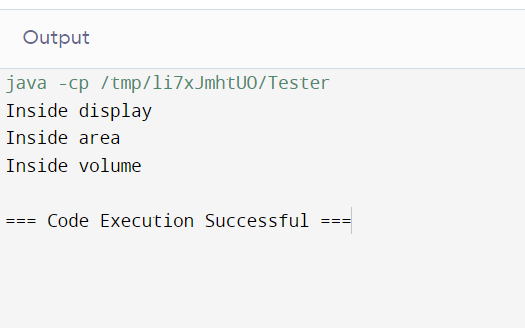
cube.area();

cube.volume();

}

}

**OUTPUT:**

****

**RESULT:**

|  |  |
| --- | --- |
| **Ex.No:** | **Implementation of Math Package** |
| **Date:** |

**AIM:**

To write a Java Program to demonstrate various arithmetic calculation using Package

**ALGORITHEM:**

STEP1**:** Start the Program

STEP2**:**  Create a new class name as ArithmeticCalculation and it define main method

STEP3**:** Get the input from the user and store the variable num1, num2

STEP4**:** Use the Math methods to perform the Arithmetic Calculations and Print it

STEP5**:** End the process

**PROGRAM:**

package program;

import java.util.Scanner;

public class ArithmeticCalculation {

public static void main(String[] args) {

Scanner sc=new Scanner (System.in);

System.out.println("Enter the First Number : ");

int num1=sc.nextInt();

System.out.println("Enter the Second Number : ");

int num2=sc.nextInt();

System.out.println("Addition of two numbers : "+ Math.addExact(num1, num2));

System.out.println("Subraction of two numbers : "+ Math.subtractExact(num1, num2));

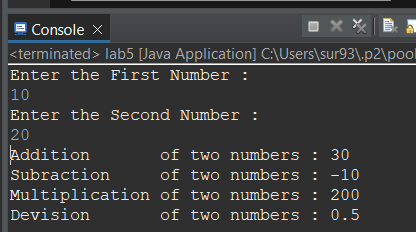
System.out.println("Multiplication of two numbers : "+ Math.multiplyExact(num1, num2));

System.out.println("Devision of two numbers : "+ (double)num1/num2);

}

}

**OUTPUT:**



**RESULT:**

|  |  |
| --- | --- |
| **Ex.No:** | **Implementation of Abstract Class** |
| **Date:** |

**AIM:**

To write a java program to implement abstract class.

**ALGORITHM:**

**Step 1:** Start the program**.**

**Step 2:** create abstract class ‘Animal’ with an abstract method ‘sound()’.

**Step 3:** Define a concrete subclass ‘Dog’ that extends ‘Animal’ and implements the ‘Sound()’ method to print “Meow!”.

**Step 4**: In the ‘Main’ method ,to create instances of ‘Dog’ and ‘cat’ and assing them to variable of type ‘Animal’.

**Step 5:** To call the ‘sound’ method on the ‘Dog’ and ‘cat’ instances to print their respective sounds.

**Step 6:** stop the program.

**PROGRAM:**

abstract class Animal {

abstract void sound();

}

class Dog extends Animal {

void sound() {

System.out.println("Woof!");

}

}

class Cat extends Animal {

void sound() {

System.out.println("Meow!");

}

}

public class MainAB{

public static void main(String[] args) {

Animal dog = new Dog();

Animal cat = new Cat();

dog.sound();

cat.sound();

}

}

**OUTPUT:**



**RESULT:**

|  |  |
| --- | --- |
| **Ex.No:** | **Implementation of String Handling** |
| **Date:** |

**AIM:**

To write a java program to implement string handling.

**ALGORITHM:**

**Step 1:** Start the program.

**Step 2**: Import the Scanner class from java.util package.

**Step 3**: Create a class named StringHandlingExample.

**Step 4:** Define the main method inside the class**.**

**Step 5:** Initialize a Scanner object to take input from the user.

**Step 6:** Prompt the user to enter their aim, algorithm, and program one by one using println statements.

**Step 7:** Read the input provided by the user for aim, algorithm, and program using the nextLine() method of Scanner and store them in respective String variables.

**Step 8:** Print the entered aim, algorithm, and program as output.

**Step 9:** Close the Scanner object to release the resources

**Step 10:** stop the program.

**PROGRAM:**

public class StringHandlingExample {

public static void main(String[] args) {

String str1 = "Hello";

String str2 = "World";

String result = str1 + " " + str2;

System.out.println("Concatenated String: " + result);

int length = result.length();

System.out.println("Length of String: " + length);

String sub = result.substring(0, 5);

System.out.println("Substring: " + sub);

String upperCase = result.toUpperCase();

System.out.println("Uppercase: " + upperCase);

String lowerCase = result.toLowerCase();

System.out.println("Lowercase: " + lowerCase);

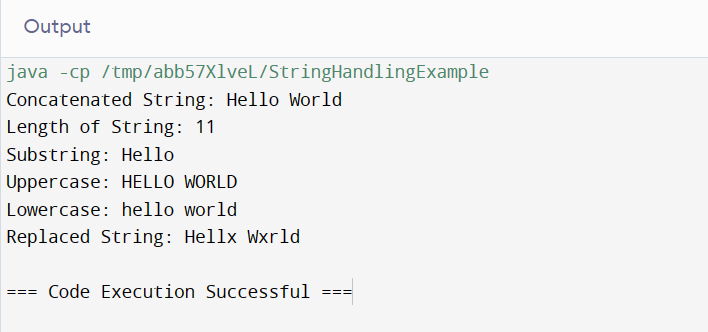
String replaced = result.replace('o', 'x');

System.out.println("Replaced String: " + replaced);

}

}

**OUTPUT:**

****

**RESULT:**

|  |  |
| --- | --- |
| **Ex.No:** | **Implementation of Exception Handling** |
| **Date:** |

**AIM:**

To write a java program to implement exception handling.

**ALGORITHM:**

**STEP 1**: Start the program.

**STEP 2**: Wrap the code that might throw an exception inside **try** block.

**STEP 3**: Immediately after the **try** block, include a **catch** block to catch the specific type of exception.

**STEP 4**: Inside the **catch** block, write code to handle the exception, such as displaying an error message.

**STEP 5**: Optionally, include a **finally** block for cleanup operations, which will execute regardless of whether an exception occurs.

**STEP 6**: Stop the program.

**PROGRAM:**

import java.util.Scanner;

public class Main {

public static void main(String[] args) {

Scanner scanner = new Scanner(System.in);

try {

System.out.println("Enter a number:");

int num = scanner.nextInt();

System.out.println("You entered: " + num);

} catch (Exception e) {

System.out.println("Error: Input must be a valid integer.");

} finally {

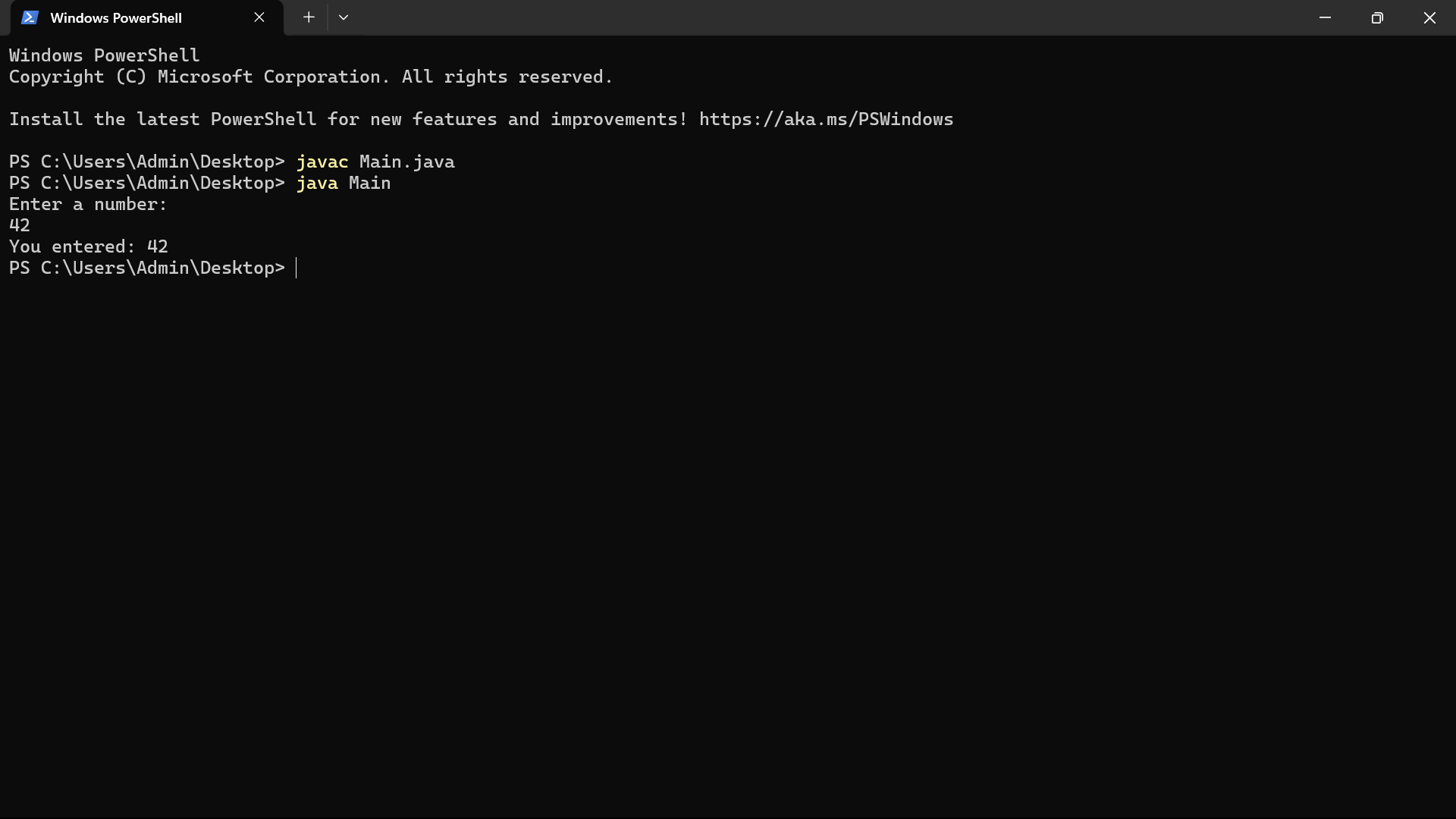
scanner.close();

}

}

}

**OUTPUT :**

****

**RESULT:**

|  |  |
| --- | --- |
| **Ex.No:** | **Implementation of LinkedList class** |
| **Date:** |

**AIM:**

To write a java program to demonstrate linked list class.

**ALGORITHM:**

STEP 1: Create a new node with the given data.

STEP 2: Check if the head of the linked list is null.

- If the head is null, set the head to point to the new node and exit the method.

STEP 3: If the head is not null, create a reference variable current and set it to point to the head node.

STEP 4: Traverse the list until you reach the last node.

- Start a loop that continues until current.next is null.

- Inside the loop, update current to point to the next node in the list (current.next).

STEP 5: Once the loop exits, current will be pointing to the last node in the list.

STEP 6: Set the next reference of the last node (current.next) to point to the new node created in step 1.

STEP 7: Exit the method.

**PROGRAM:**

public class LinkedList {

private Node head;

private static class Node {

int data;

Node next;

public Node(int data) {

this.data = data;

this.next = null;

}

}

public void add(int data) {

Node newNode = new Node(data);

if (head == null) {

head = newNode;

} else {

Node current = head;

while (current.next != null) {

current = current.next;

}

current.next = newNode;

}

}

public void printList() {

Node current = head;

while (current != null) {

System.out.print(current.data + " -> ");

current = current.next;

}

System.out.println("null");

}

public static void main(String[] args) {

LinkedList list = new LinkedList();

list.add(1);

list.add(2);

list.add(3);

list.add(4);

list.add(5);

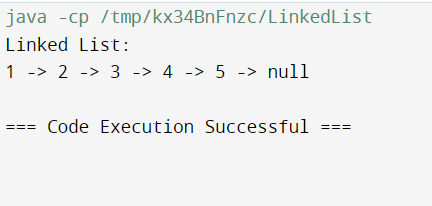
System.out.println("Linked List:");

list.printList();

}

}

**OUTPUT:**

****

**RESULT:**

|  |  |
| --- | --- |
| **Ex.No:** | **Implementation of HashSet class** |
| **Date:** |

**AIM:**

To write a java program to demonstrate hashset class.

**ALGORITHM:**

**Step 1**: Start the program.

**Step 2:** Import the required classes from the java.util package (HashSet and Iterator).

**Step 3:** Create a class named HashSetExample.

**Step 4:** Define the main method inside the class.

**Step 5:** Create a HashSet named hashSet to store strings.

**Step 6:** Add elements ("Apple", "Banana", "Orange") to the hashSet. Note that duplicate elements are ignored.

**Step 7:** Display the contents of the HashSet.

**Step 8:** Output the size of the HashSet.

**Step 9:** Remove the element "Banana" from the HashSet.

**Step 10:** Check if the element "Apple" exists in the HashSet.

**Step 11:** Iterate over the HashSet using an iterator and print each element.

**Step 12:** Clear the HashSet.

**Step 13:** Check if the HashSet is empty.

**PROGRAM:**

import java.util.HashSet;

import java.util.Iterator;

public class HashSetExample {

public static void main(String[] args) {

HashSet<String> hashSet = new HashSet<>();

hashSet.add("Apple");

hashSet.add("Banana");

hashSet.add("Orange");

hashSet.add("Apple"); // Duplicate elements are ignored

System.out.println("HashSet: " + hashSet);

System.out.println("Size of HashSet: " + hashSet.size());

hashSet.remove("Banana");

System.out.println("Contains 'Apple': " + hashSet.contains("Apple"));

System.out.println("Iterating over the HashSet:");

Iterator<String> iterator = hashSet.iterator();

while (iterator.hasNext()) {

System.out.println(iterator.next());

}

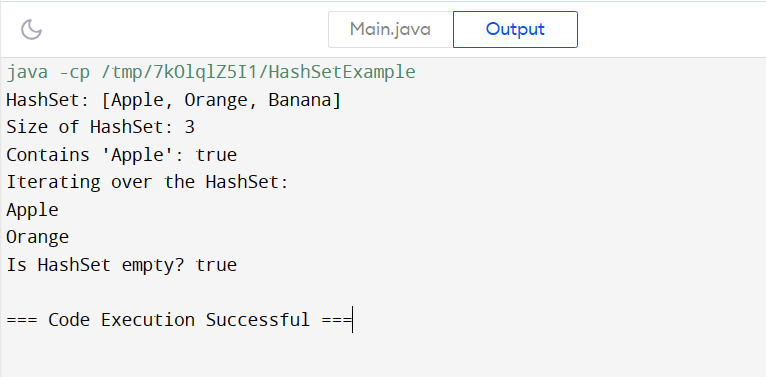
hashSet.clear();

System.out.println("Is HashSet empty? " + hashSet.isEmpty());

}

}

**OUTPUT:**

****

**RESULT:**

|  |  |
| --- | --- |
| **Ex.No:** | **Implementation of File Handling** |
| **Date:** |

**AIM:**

To write a java program to file handling (Create, Read, Write, Delete) with name and absolute path of the file

**ALGORITHM:**

STEP1: Start the program.

STEP2: Create a file with the class name as File\_hande and import all the file packages.

STEP3: Define the file name and file path.

STEP4: Define a function with name “createfile” to create a new text file by using the file name and path as parameters.

File file = new File(filePath + fileName);

STEP5: Define a function with name “writefile” to write into the text file by using the file name, file path and content to be written as parameters.

writer.write(content);

STEP6: Define a function with name “readfile” to read the text file by using the file name and file path as parameters.

FileReader reader = new FileReader(filePath + fileName);

STEP7: Define a function with name “deletefile ” to delete a file by using the file name and path as parameters.

file.delete()

STEP8: In the main class call all the functions to execute.

STEP9**:** Stop the process.

**PROGRAM:**

package Basics;

import java.io.File;

import java.io.FileReader;

import java.io.FileWriter;

import java.io.IOException;

public class File\_Hande {

public static void main(String[] args) {

String fileName = "example.txt";

String filePath = "D:/notes/Java/";

createFile(fileName, filePath);

writeFile(fileName, filePath, "Hello, World!");

String content = readFile(fileName, filePath);

System.out.println("Content read from the file: " + content);

deleteFile(fileName, filePath);

}

public static void createFile(String fileName, String filePath) {

try {

File file = new File(filePath + fileName);

if (file.createNewFile()) {

System.out.println("File created: " + file.getName());

System.out.println("Absolute Path: " + file.getAbsolutePath());

} else {

System.out.println("File already exists.");

}

} catch (IOException e) {

System.out.println("An error occurred while creating the file.");

e.printStackTrace();

}

}

public static void writeFile(String fileName, String filePath, String content) {

try {

FileWriter writer = new FileWriter(filePath + fileName);

writer.write(content);

writer.close();

System.out.println("Successfully wrote to the file.");

} catch (IOException e) {

System.out.println("An error occurred while writing to the file.");

e.printStackTrace();

}

}

public static String readFile(String fileName, String filePath) {

StringBuilder content = new StringBuilder();

try {

FileReader reader = new FileReader(filePath + fileName);

int character;

while ((character = reader.read()) != -1) {

content.append((char) character);

}

reader.close();

} catch (IOException e) {

System.out.println("An error occurred while reading the file.");

e.printStackTrace();

}

return content.toString();

}

public static void deleteFile(String fileName, String filePath) {

File file = new File(filePath + fileName);

if (file.delete()) {

System.out.println("File deleted successfully.");

} else {

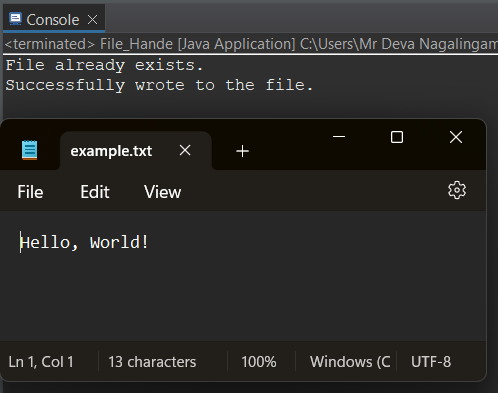
System.out.println("Failed to delete the file.");

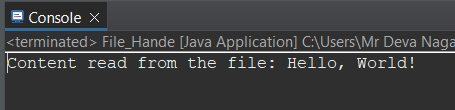
}

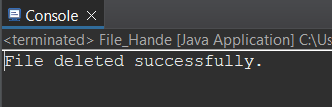
}

}

**OUTPUT :**

****

****

****

**RESULT:**

|  |  |
| --- | --- |
| **Ex.No:** | **Calculator Application** |
| **Date:** |

**AIM:**

**To write a java program to create a calculator application.**

**ALGORITHM:**

**STEP 1: Initialization**

a.] The program starts by importing necessary packages and defining the main class CalculatorWithGUI, which extends Frame and implements ActionListener.

b.] Inside the class, various components are declared, including a JTextField for displaying input and output, a JPanel for holding buttons, an array of button strings, an array of JButton objects, and variables to store numbers, results, and mathematical operators.

**STEP 2: Constructor (Calculator With GUI()):**

a.] This method sets up the GUI components.

b.] It initializes the font and creates a text field (textInput) and a panel (panel).

c.] It sets the layout of the panel to a 5x4 grid.

d.] It creates buttons using the button strings, sets their font, adds action listeners, and adds them to the panel.

e.] Finally, it adds a window listener to exit the program when the window is closed.

**STEP 3: Action Performed (actionPerformed(ActionEvent ae)):**

a.] This method is invoked whenever a button is clicked.

b.] It retrieves the text from the clicked button and performs different actions based on the text.

c.] If the button represents an arithmetic operator (+, -, \*, /, ^, √, %), it sets the operator (charSymbol) and stores the first operand (num1), then clears the text field for the next input.

d.] If the button is "C" (clear), it clears the text field and resets num1.

e.] If the button is the backspace symbol (\u232b), it removes the last character from the text field.

f.] If the button is "=" (equals), it calculates the result based on the stored operator and operands, then displays the result in the text field.

g.] If the button is a digit, it appends the digit to the text field.

**STEP 4: Main Method (main()):**

a.] This method creates an instance of CalculatorWithGUI, sets its title, size, and colors, and makes it visible.

**PROGRAM:**

import javax.swing.\*;

import java.awt.\*;

import java.awt.event.\*;

import static java.lang.Double.parseDouble;

class CalculatorWithGUI extends Frame implements ActionListener {

JTextField textInput;

JPanel panel;

String[] btnString = { "7", "8", "9", "+", "4", "5", "6", "-", "1", "2", "3", "\*", "C", "0", "=", "÷", "^", "√", "%", "\u232b"};

int n = 20; //number of buttons

JButton[] btn = new JButton[n];

double num1 = 0;

double num2 = 0;

String result = " ";

char charSymbol;

public CalculatorWithGUI() {

Font f = new Font("MONOSPACED", Font.BOLD, 18);

textInput = new JTextField(10);

textInput.setFont(f);

panel = new JPanel();

add(textInput, "North");

add(panel, "Center");

panel.setLayout(new GridLayout(5, 4));

for (int i = 0; i < n; i++) {

btn[i] = new JButton(btnString[i]);

btn[i].setFont(f);

btn[i].addActionListener(this);

panel.add(btn[i]);

}

addWindowListener(new WindowAdapter() {

public void windowClosing(WindowEvent we) {

System.exit(0);

}

});

}

public void actionPerformed(ActionEvent ae) {

String str = ae.getActionCommand();

switch (str) {

case "+" -> {

charSymbol = '+';

num1 = parseDouble(textInput.getText());

textInput.setText("");

}

case "-" -> {

charSymbol = '-';

num1 = parseDouble(textInput.getText());

textInput.setText("");

}

case "\*" -> {

charSymbol = '\*';

num1 = parseDouble(textInput.getText());

textInput.setText("");

}

case "÷" -> {

charSymbol = '÷';

num1 = parseDouble(textInput.getText());

textInput.setText("");

}

case "^"-> {

charSymbol = '^';

num1 = parseDouble(textInput.getText());

textInput.setText("");

}

case "√"-> {

charSymbol = '√';

textInput.setText("");

}

case "%" -> {

charSymbol = '%';

num1 = parseDouble(textInput.getText());

textInput.setText("");

}

case "\u232b" -> {

String theText = textInput.getText();

if (theText.length() == 0) {

result = "";

} else {

result = theText.substring(0, theText.length()-1);

}

textInput.setText(result + "");

result = " ";

}

case "=" -> {

num2 = parseDouble(textInput.getText());

switch (charSymbol) {

case '+' -> result = String.valueOf(num1 + num2);

case '-' -> result = String.valueOf(num1 - num2);

case '\*' -> result = String.valueOf(num1 \* num2);

case '÷' -> result = String.valueOf(num1 / num2);

case '^' -> result = Double.toString(Math.pow(num1, num2));

case '√' -> result = Double.toString(Math.sqrt(num2));

case '%' -> {

if(num2 == 0){

result ="DIVISOR IS 0";

}

else {

result = String.valueOf(Math.floorMod((long) num1, (long) num2));

}

}

}

textInput.setText(result + "");

result = " ";

}

case "C" -> {

textInput.setText("");

num1 = 0;

}

default -> textInput.setText(textInput.getText() + str);

}

}

public static void main(String[] args) {

CalculatorWithGUI m = new CalculatorWithGUI();

m.setTitle("Calculator using Java (AWT)");

m.setSize(250, 300);

m.setBackground(Color.CYAN);

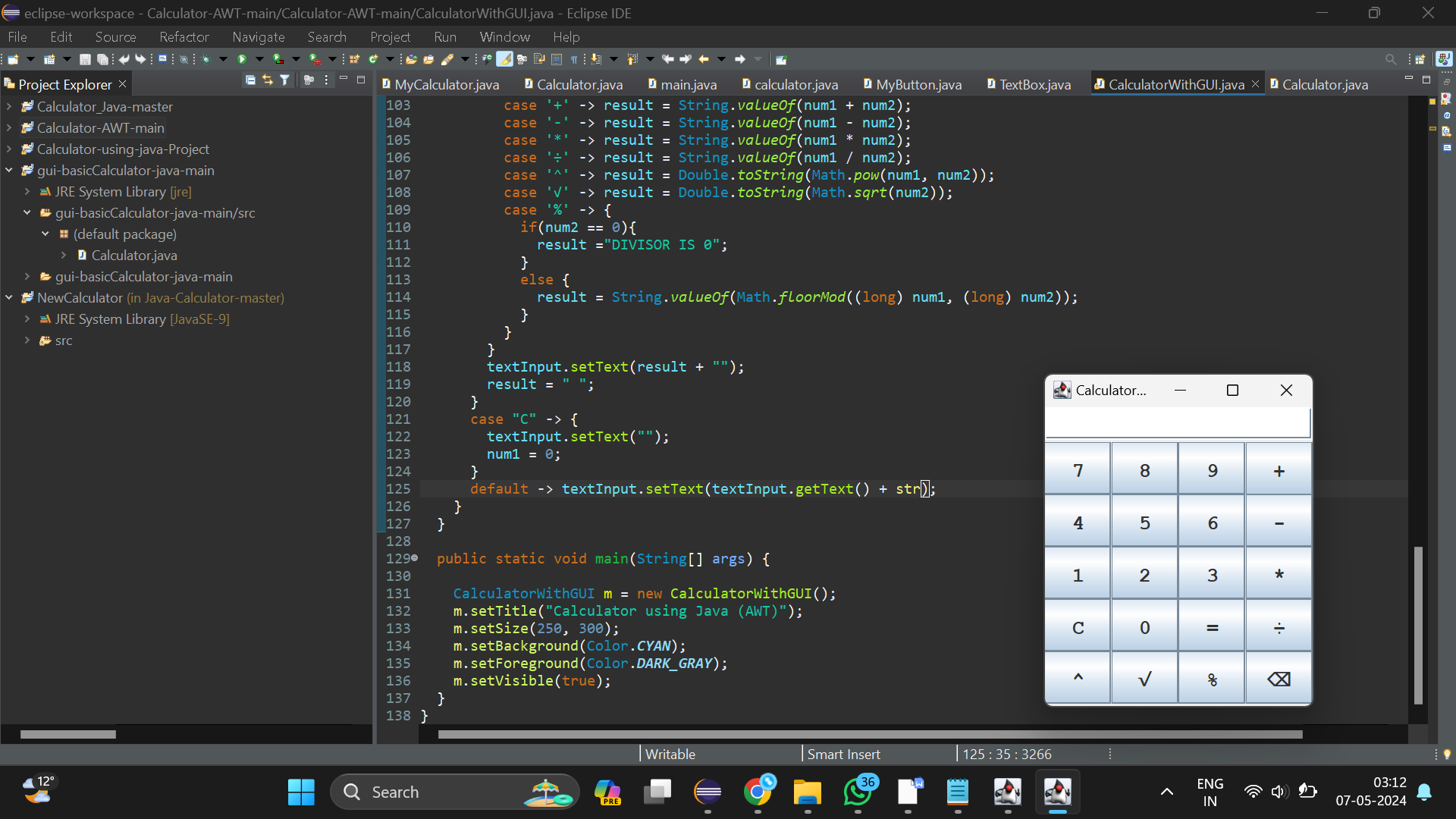
m.setForeground(Color.DARK\_GRAY);

m.setVisible(true);

}

}

**OUTPUT:**



**RESULT:**