27. import java.util.HashMap;

import java.util.Map;

public class EmployeeRecord {

public static void main(String[] args) {

// Creating a map to store employee records

Map<Integer, Employee> employeeMap = new HashMap<>();

// Adding objects to the map

addEmployee(employeeMap, 101, new Employee("John", "Doe", "john@example.com", "IT"));

addEmployee(employeeMap, 102, new Employee("Jane", "Doe", "jane@example.com", "HR"));

addEmployee(employeeMap, 103, new Employee("Alice", "Smith", "alice@example.com", "Finance"));

// Checking if the map is empty

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

// Removing a specified object

removeEmployee(employeeMap, 102);

// Clearing the map

employeeMap.clear();

// Checking if the map is empty after clearing

System.out.println("Is employeeMap empty after clearing? " + employeeMap.isEmpty());

}

// Method to add an employee to the map

private static void addEmployee(Map<Integer, Employee> employeeMap, int id, Employee employee) {

employeeMap.put(id, employee);

System.out.println("Added employee: " + employee);

}

// Method to remove an employee from the map

private static void removeEmployee(Map<Integer, Employee> employeeMap, int id) {

if (employeeMap.containsKey(id)) {

Employee removedEmployee = employeeMap.remove(id);

System.out.println("Removed employee: " + removedEmployee);

} else {

System.out.println("Employee with ID " + id + " not found.");

}

}

// Employee class representing employee details

static class Employee {

private String firstName;

private String lastName;

private String email;

private String department;

public Employee(String firstName, String lastName, String email, String department) {

this.firstName = firstName;

this.lastName = lastName;

this.email = email;

this.department = department;

}

@Override

public String toString() {

return "Employee{" +

"firstName='" + firstName + '\'' +

", lastName='" + lastName + '\'' +

", email='" + email + '\'' +

", department='" + department + '\'' +

'}';

}

}

}

28. import java.util.Arrays;

public class GenericSort<T extends Comparable<T>> {

private T[] array;

public GenericSort(T[] array) {

this.array = array;

}

public void sort() {

Arrays.sort(array);

}

public void printSortedArray() {

for (T value : array) {

System.out.print(value + " ");

}

System.out.println();

}

public static void main(String[] args) {

// Example usage

Integer[] intArray = {5, 3, 8, 1, 2};

GenericSort<Integer> intSort = new GenericSort<>(intArray);

intSort.sort();

System.out.print("Sorted Integer Array: ");

intSort.printSortedArray();

String[] stringArray = {"banana", "apple", "orange", "grape"};

GenericSort<String> stringSort = new GenericSort<>(stringArray);

stringSort.sort();

System.out.print("Sorted String Array: ");

stringSort.printSortedArray();

}

}

29. import java.util.ArrayList;

import java.util.List;

import java.util.ListIterator;

public class ListIteratorExample {

public static void main(String[] args) {

List<String> elements = new ArrayList<>();

elements.add("C");

elements.add("A");

elements.add("E");

elements.add("B");

elements.add("D");

elements.add("F");

// Using ListIterator to append "+" symbol to each element

ListIterator<String> iterator = elements.listIterator();

while (iterator.hasNext()) {

String element = iterator.next();

iterator.set(element + "+");

}

// Printing elements in reverse order

System.out.println("Elements in reverse order with '+' appended:");

while (iterator.hasPrevious()) {

System.out.print(iterator.previous() + " ");

}

System.out.println();

}

}

30. public class ArithmeticOperations {

public static void main(String[] args) {

int a = 10;

int b = 0;

// Addition

int sum = a + b;

System.out.println("Sum: " + sum);

// Subtraction

int difference = a - b;

System.out.println("Difference: " + difference);

// Multiplication

int product = a \* b;

System.out.println("Product: " + product);

try {

// Division

if (b == 0) {

throw new ArithmeticException("Division by zero is not allowed");

}

int quotient = a / b;

System.out.println("Quotient: " + quotient);

} catch (ArithmeticException e) {

System.out.println("Error: " + e.getMessage());

}

}

}

31. public class ParallelThreads {

public static void main(String[] args) {

// Create and start three threads

Thread thread1 = new Thread(new NumberPrinter(1));

Thread thread2 = new Thread(new NumberPrinter(2));

Thread thread3 = new Thread(new NumberPrinter(3));

thread1.start();

thread2.start();

thread3.start();

}

static class NumberPrinter implements Runnable {

private final int threadNumber;

private static final int MAX\_NUMBERS = 10;

public NumberPrinter(int threadNumber) {

this.threadNumber = threadNumber;

}

@Override

public void run() {

for (int i = 1; i <= MAX\_NUMBERS; i++) {

// Only allow the current thread to print if it's the correct thread number

if (i % 3 == threadNumber - 1) {

System.out.println("Thread " + threadNumber + ": " + i);

}

try {

// Pause execution for 500 milliseconds to allow other threads to run

Thread.sleep(500);

} catch (InterruptedException e) {

e.printStackTrace();

}

}

}

}

}

32. public class FindMissingNumber {

public static void main(String[] args) {

int n = 8;

int[] a = {1, 4, 5, 3, 7, 8, 6};

int expectedSum = (n \* (n + 1)) / 2;

int actualSum = 0;

for (int num : a) {

actualSum += num;

}

int missingNumber = expectedSum - actualSum;

System.out.println("Missing number: " + missingNumber);

}

}

33. class Parent {

public void parentMethod() {

System.out.println("This is parent class");

}

}

class Child extends Parent {

public void childMethod() {

System.out.println("This is child class");

}

}

public class Main {

public static void main(String[] args) {

// Creating objects for parent and child classes

Parent parentObj = new Parent();

Child childObj = new Child();

// 1 - Method of parent class by object of parent class

parentObj.parentMethod();

// 2 - Method of child class by object of child class

childObj.childMethod();

// 3 - Method of parent class by object of child class

childObj.parentMethod(); // This is possible because Child class inherits from Parent class

}

}

34. public class Student {

private String name;

private int registerNumber;

private int[] marks = new int[5];

// Constructor to initialize student details

public Student(String name, int registerNumber, int[] marks) {

this.name = name;

this.registerNumber = registerNumber;

this.marks = marks;

}

// Method to calculate total marks

public int calculateTotalMarks() {

int total = 0;

for (int mark : marks) {

total += mark;

}

return total;

}

// Method to calculate average marks

public double calculateAverageMarks() {

return calculateTotalMarks() / 5.0;

}

// Method to display student details and marks

public void displayStudentDetails() {

System.out.println("Student Name: " + name);

System.out.println("Register Number: " + registerNumber);

System.out.println("Subject Marks:");

for (int i = 0; i < marks.length; i++) {

System.out.println("Subject " + (i + 1) + ": " + marks[i]);

}

System.out.println("Total Marks: " + calculateTotalMarks());

System.out.println("Average Marks: " + calculateAverageMarks());

}

public static void main(String[] args) {

int[] marks = {85, 90, 75, 80, 95}; // Example marks

Student student = new Student("John", 123456, marks);

student.displayStudentDetails();

}

}

35. import java.io.BufferedReader;

import java.io.FileReader;

import java.io.IOException;

public class FileStatistics {

public static void main(String[] args) {

String filePath = "example.txt"; // Path to your file

try (BufferedReader reader = new BufferedReader(new FileReader(filePath))) {

int wordCount = 0;

int charCount = 0;

int lineCount = 0;

String line;

while ((line = reader.readLine()) != null) {

lineCount++;

String[] words = line.split("\\s+"); // Split the line into words

wordCount += words.length; // Increment word count by the number of words in the line

charCount += line.length(); // Increment char count by the length of the line

}

System.out.println("Number of words: " + wordCount);

System.out.println("Number of characters: " + charCount);

System.out.println("Number of lines: " + lineCount);

} catch (IOException e) {

System.err.println("Error reading file: " + e.getMessage());

}

}

}

36. public class MultiplyStrings {

public static String multiply(String num1, String num2) {

// Convert strings to character arrays and reverse them for easier manipulation

char[] n1 = new StringBuilder(num1).reverse().toString().toCharArray();

char[] n2 = new StringBuilder(num2).reverse().toString().toCharArray();

// Create an array to store the intermediate results of multiplication

int[] result = new int[n1.length + n2.length];

// Multiply each digit of num1 with each digit of num2

for (int i = 0; i < n1.length; i++) {

for (int j = 0; j < n2.length; j++) {

result[i + j] += (n1[i] - '0') \* (n2[j] - '0');

}

}

// Perform carry-over

StringBuilder sb = new StringBuilder();

for (int i = 0; i < result.length; i++) {

int digit = result[i] % 10;

int carry = result[i] / 10;

sb.insert(0, digit);

if (i < result.length - 1) {

result[i + 1] += carry;

}

}

// Remove leading zeros

while (sb.length() > 1 && sb.charAt(0) == '0') {

sb.deleteCharAt(0);

}

return sb.toString();

}

public static void main(String[] args) {

String num1 = "123";

String num2 = "456";

String product = multiply(num1, num2);

System.out.println("Product of " + num1 + " and " + num2 + " is: " + product);

}

}