1. package Day8;

import java.util.ArrayList;

import java.util.Scanner;

public class ArrayListSearch {

public static void main(String[] args) {

// Create an ArrayList of integers

ArrayList<Integer> numbers = new ArrayList<>();

numbers.add(10);

numbers.add(20);

numbers.add(30);

numbers.add(40);

numbers.add(50);

// Ask user for input

Scanner sc = new Scanner(System.in);

System.out.print("Enter a number to search: ");

int num = sc.nextInt();

// Check if the number exists in the list

if (numbers.contains(num)) {

System.out.println(num + " exists in the list.");

} else {

System.out.println(num + " does not exist in the list.");

}

sc.close();

}

}

2. package Day8;

import java.util.ArrayList;

public class ClearArrayList {

public static void main(String[] args) {

// Create an ArrayList of integers

ArrayList<Integer> numbers = new ArrayList<>();

numbers.add(10);

numbers.add(20);

numbers.add(30);

numbers.add(40);

numbers.add(50);

// Display original list

System.***out***.println("Original List: " + numbers);

// Remove all elements

numbers.clear();

// Display size of the list

System.***out***.println("All elements removed.");

System.***out***.println("Size of List after clear(): " + numbers.size());

}

}

3. package Day8;

import java.util.ArrayList;

public class CopyArrayList {

public static void main(String[] args) {

// Create the first ArrayList

ArrayList<String> list1 = new ArrayList<>();

list1.add("Apple");

list1.add("Banana");

list1.add("Mango");

list1.add("Orange");

// Display original list

System.***out***.println("Original List 1: " + list1);

// Create the second ArrayList

ArrayList<String> list2 = new ArrayList<>();

// Copy all elements from list1 to list2

list2.addAll(list1);

// Display the copied list

System.***out***.println("Copied List 2: " + list2);

}

}

4. package Day8;

import java.util.ArrayDeque;

import java.util.Deque;

import java.util.Scanner;

public class DequeDemo {

// ---------- Part 1: Palindrome Checker ----------

public static boolean isPalindrome(String str) {

Deque<Character> deque = new ArrayDeque<>();

for (char ch : str.toCharArray()) {

deque.addLast(ch);

}

while (deque.size() > 1) {

if (!deque.removeFirst().equals(deque.removeLast())) {

return false;

}

}

return true;

}

// ---------- Part 2: Double-ended Order System ----------

public static void doubleEndedOrderSystem() {

Deque<String> orders = new ArrayDeque<>();

System.out.println("\n--- Double-ended Order System ---");

// Add items at front and rear

orders.addFirst("Burger");

orders.addLast("Pizza");

orders.addFirst("Sushi");

orders.addLast("Pasta");

System.out.println("After adding items: " + orders);

// Remove items from both ends

orders.removeFirst();

System.out.println("After removing from front: " + orders);

orders.removeLast();

System.out.println("After removing from rear: " + orders);

}

// ---------- Part 3: Browser History Simulation ----------

public static void browserHistorySimulation() {

Deque<String> backStack = new ArrayDeque<>();

Deque<String> forwardStack = new ArrayDeque<>();

System.out.println("\n--- Browser History Simulation ---");

// Visiting pages

backStack.push("google.com");

backStack.push("openai.com");

backStack.push("stackoverflow.com");

System.out.println("Current page: " + backStack.peek());

// Go back

forwardStack.push(backStack.pop());

System.out.println("After going back, current page: " + backStack.peek());

// Go back again

forwardStack.push(backStack.pop());

System.out.println("After going back again, current page: " + backStack.peek());

// Go forward

backStack.push(forwardStack.pop());

System.out.println("After going forward, current page: " + backStack.peek());

}

public static void main(String[] args) {

Scanner sc = new Scanner(System.in);

// Palindrome Checker

System.out.print("Enter a string to check palindrome: ");

String input = sc.nextLine();

if (isPalindrome(input)) {

System.out.println(input + " is a palindrome.");

} else {

System.out.println(input + " is not a palindrome.");

}

// Double-ended Order System

doubleEndedOrderSystem();

// Browser History Simulation

browserHistorySimulation();

sc.close();

}

}

5. package Day8;

import java.util.HashSet;

import java.util.Iterator;

public class HashSetDemo {

// Method to find maximum element in HashSet<Integer>

public static int getMax(HashSet<Integer> numbers) {

if(numbers.isEmpty()) {

throw new IllegalArgumentException("HashSet is empty");

}

int max = Integer.MIN\_VALUE;

for(int num : numbers) {

if(num > max) {

max = num;

}

}

return max;

}

public static void main(String[] args) {

// --------- Part 1: HashSet of Strings -----------

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

// Adding 5 cities

cities.add("New York");

cities.add("London");

cities.add("Tokyo");

cities.add("Paris");

cities.add("Sydney");

System.out.println("Original HashSet: " + cities);

// Trying to add a duplicate

boolean added = cities.add("Tokyo");

System.out.println("Trying to add duplicate 'Tokyo': " + added);

System.out.println("HashSet after attempting duplicate: " + cities);

// Iterating using Iterator

System.out.println("\nIterating using Iterator:");

Iterator<String> it = cities.iterator();

while(it.hasNext()) {

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

}

// Remove an element

cities.remove("Paris");

System.out.println("\nHashSet after removing 'Paris': " + cities);

// Check if a city exists

System.out.println("Does 'London' exist? " + cities.contains("London"));

System.out.println("Does 'Paris' exist? " + cities.contains("Paris"));

// Clear entire HashSet

cities.clear();

System.out.println("HashSet after clearing: " + cities);

// --------- Part 2: Maximum in HashSet<Integer> -----------

HashSet<Integer> numbers = new HashSet<>();

numbers.add(15);

numbers.add(42);

numbers.add(7);

numbers.add(89);

numbers.add(30);

System.out.println("\nHashSet of numbers: " + numbers);

System.out.println("Maximum element: " + getMax(numbers));

}

}

6. package Day8;

import java.util.LinkedHashSet;

import java.util.Iterator;

// Custom Student class

class Student {

int id;

String name;

Student(int id, String name) {

this.id = id;

this.name = name;

}

// Override equals and hashCode to handle duplicates

@Override

public boolean equals(Object obj) {

if (this == obj) return true;

if (!(obj instanceof Student)) return false;

Student other = (Student) obj;

return this.id == other.id && this.name.equals(other.name);

}

@Override

public int hashCode() {

return id \* 31 + name.hashCode();

}

@Override

public String toString() {

return "Student{id=" + id + ", name='" + name + "'}";

}

}

public class LinkedHashSetDemo {

public static void main(String[] args) {

// --------- Part 1: LinkedHashSet of Integers -----------

LinkedHashSet<Integer> numbers = new LinkedHashSet<>();

numbers.add(10);

numbers.add(5);

numbers.add(20);

numbers.add(15);

numbers.add(5); // duplicate, will not be added

System.out.println("LinkedHashSet of numbers (preserves insertion order): " + numbers);

// --------- Part 2: LinkedHashSet of Custom Objects -----------

LinkedHashSet<Student> students = new LinkedHashSet<>();

students.add(new Student(101, "Alice"));

students.add(new Student(102, "Bob"));

students.add(new Student(103, "Charlie"));

// Try adding a duplicate

boolean added = students.add(new Student(102, "Bob"));

System.out.println("\nTrying to add duplicate student: " + added);

System.out.println("LinkedHashSet of students:");

for (Student s : students) {

System.out.println(s);

}

// --------- Part 3: Merging two LinkedHashSets -----------

LinkedHashSet<Integer> moreNumbers = new LinkedHashSet<>();

moreNumbers.add(25);

moreNumbers.add(5); // already exists in first set

moreNumbers.add(30);

numbers.addAll(moreNumbers); // merge sets

System.out.println("\nMerged LinkedHashSet of numbers: " + numbers);

}

}

7. package Day8;

import java.util.LinkedList;

public class LinkedListAddFirstLast {

public static void main(String[] args) {

// Create a LinkedList of integers

LinkedList<Integer> numbers = new LinkedList<>();

// Add elements to the list

numbers.add(10);

numbers.add(20);

numbers.add(30);

System.***out***.println("Original LinkedList: " + numbers);

// Add element at the first position

numbers.addFirst(5);

// Add element at the last position

numbers.addLast(40);

// Display the updated list

System.***out***.println("Updated LinkedList after adding first and last elements: " + numbers);

}

}

8. package Day8;

import java.util.LinkedList;

public class LinkedListClone {

public static void main(String[] args) {

LinkedList<Integer> original = new LinkedList<>();

original.add(1);

original.add(2);

original.add(3);

original.add(4);

// Clone the list

*@SuppressWarnings*("unchecked")

LinkedList<Integer> cloned = (LinkedList<Integer>) original.clone();

System.***out***.println("Original LinkedList: " + original);

System.***out***.println("Cloned LinkedList: " + cloned);

}

}

9. package Day8;

import java.util.LinkedList;

class Book {

int id;

String title;

String author;

Book(int id, String title, String author){

this.id = id;

this.title = title;

this.author = author;

}

void display() {

System.***out***.println("ID: " + id + ", Title: " + title + ", Author: " + author);

}

}

public class LinkedListCustomObjects {

public static void main(String[] args) {

LinkedList<Book> books = new LinkedList<>();

books.add(new Book(101, "Java Basics", "Alice"));

books.add(new Book(102, "Data Structures", "Bob"));

books.add(new Book(103, "Algorithms", "Charlie"));

System.***out***.println("Book Details:");

for(Book b : books) {

b.display();

}

}

}

10. package Day8;

import java.util.LinkedList;

public class LinkedListExample {

public static void main(String[] args) {

// Create a LinkedList of Strings

LinkedList<String> colors = new LinkedList<>();

// Add five colors

colors.add("Red");

colors.add("Blue");

colors.add("Green");

colors.add("Yellow");

colors.add("Orange");

// Display the LinkedList using for-each loop

System.***out***.println("Colors in the LinkedList:");

for (String color : colors) {

System.***out***.println(color);

}

}

}

11. package Day8;

import java.util.LinkedList;

public class LinkedListExample {

public static void main(String[] args) {

// Create a LinkedList of Strings

LinkedList<String> colors = new LinkedList<>();

// Add five colors

colors.add("Red");

colors.add("Blue");

colors.add("Green");

colors.add("Yellow");

colors.add("Orange");

// Display the LinkedList using for-each loop

System.***out***.println("Colors in the LinkedList:");

for (String color : colors) {

System.***out***.println(color);

}

}

}

12. package Day8;

import java.util.LinkedList;

import java.util.ListIterator;

public class LinkedListIterator {

public static void main(String[] args) {

LinkedList<String> cities = new LinkedList<>();

cities.add("New York");

cities.add("London");

cities.add("Paris");

cities.add("Tokyo");

ListIterator<String> it = cities.listIterator();

System.out.println("Forward iteration:");

while(it.hasNext()) {

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

}

System.out.println("Reverse iteration:");

while(it.hasPrevious()) {

System.out.println(it.previous());

}

}

}

}

13. package Day8;

import java.util.LinkedList;

import java.util.ListIterator;

public class LinkedListIterator {

public static void main(String[] args) {

LinkedList<String> cities = new LinkedList<>();

cities.add("New York");

cities.add("London");

cities.add("Paris");

cities.add("Tokyo");

ListIterator<String> it = cities.listIterator();

System.out.println("Forward iteration:");

while(it.hasNext()) {

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

}

System.out.println("Reverse iteration:");

while(it.hasPrevious()) {

System.out.println(it.previous());

}

}

}

}

14. package Day8;

import java.util.LinkedList;

import java.util.Scanner;

public class LinkedListSearch {

public static void main(String[] args) {

LinkedList<String> list = new LinkedList<>();

list.add("Apple");

list.add("Banana");

list.add("Mango");

list.add("Orange");

Scanner sc = new Scanner(System.in);

System.out.print("Enter a string to search: ");

String input = sc.nextLine();

if(list.contains(input)) {

System.out.println(input + " is found in the LinkedList.");

} else {

System.out.println(input + " is NOT found in the LinkedList.");

}

sc.close();

}

}

15. package Day8;

import java.util.LinkedList;

import java.util.Collections;

public class LinkedListSort {

public static void main(String[] args) {

LinkedList<Integer> numbers = new LinkedList<>();

numbers.add(42);

numbers.add(10);

numbers.add(75);

numbers.add(33);

numbers.add(5);

System.out.println("Original LinkedList: " + numbers);

Collections.sort(numbers);

System.out.println("Sorted LinkedList: " + numbers);

}

}

16. package Day8;

import java.util.LinkedList;

import java.util.ArrayList;

public class LinkedListToArrayList {

public static void main(String[] args) {

LinkedList<String> linkedList = new LinkedList<>();

linkedList.add("Red");

linkedList.add("Blue");

linkedList.add("Green");

ArrayList<String> arrayList = new ArrayList<>(linkedList);

System.out.println("LinkedList: " + linkedList);

System.out.println("ArrayList: " + arrayList);

}

}

17. package Day8;

import java.util.PriorityQueue;

import java.util.Comparator;

import java.util.Queue;

class Patient {

String name;

int severityLevel;

public Patient(String name, int severityLevel) {

this.name = name;

this.severityLevel = severityLevel;

}

@Override

public String toString() {

return name + " (Severity: " + severityLevel + ")";

}

}

class PrintJob {

String jobName;

int priority;

public PrintJob(String jobName, int priority) {

this.jobName = jobName;

this.priority = priority;

}

@Override

public String toString() {

return jobName + " (Priority: " + priority + ")";

}

}

public class PriorityQueueDemo {

// Method to merge two PriorityQueues of integers

public static PriorityQueue<Integer> mergeQueues(PriorityQueue<Integer> q1, PriorityQueue<Integer> q2) {

PriorityQueue<Integer> merged = new PriorityQueue<>();

merged.addAll(q1);

merged.addAll(q2);

return merged;

}

public static void main(String[] args) {

// -------- Part 1: Hospital Emergency Queue -----------

PriorityQueue<Patient> hospitalQueue = new PriorityQueue<>(Comparator.comparingInt(p -> -p.severityLevel)); // max severity first

hospitalQueue.add(new Patient("Alice", 5));

hospitalQueue.add(new Patient("Bob", 8));

hospitalQueue.add(new Patient("Charlie", 3));

hospitalQueue.add(new Patient("David", 10));

System.out.println("Hospital Emergency Queue:");

while (!hospitalQueue.isEmpty()) {

System.out.println("Serving: " + hospitalQueue.poll());

}

// -------- Part 2: Print Jobs Priority -----------

PriorityQueue<PrintJob> printQueue = new PriorityQueue<>(Comparator.comparingInt(p -> -p.priority)); // high priority first

printQueue.add(new PrintJob("Document1", 2));

printQueue.add(new PrintJob("Report", 5));

printQueue.add(new PrintJob("Invoice", 3));

printQueue.add(new PrintJob("Presentation", 4));

System.out.println("\nPrint Jobs Queue:");

while (!printQueue.isEmpty()) {

System.out.println("Printing: " + printQueue.poll());

}

// -------- Part 3: Merge Two PriorityQueues of Integers -----------

PriorityQueue<Integer> queue1 = new PriorityQueue<>();

queue1.add(5);

queue1.add(10);

queue1.add(3);

PriorityQueue<Integer> queue2 = new PriorityQueue<>();

queue2.add(7);

queue2.add(1);

queue2.add(8);

PriorityQueue<Integer> mergedQueue = mergeQueues(queue1, queue2);

System.out.println("\nMerged and Sorted Integer Queue:");

while (!mergedQueue.isEmpty()) {

System.out.print(mergedQueue.poll() + " ");

}

}

}

18. package Day8;

import java.util.Queue;

import java.util.LinkedList;

import java.util.List;

import java.util.ArrayList;

public class QueueDemo {

// Method to return even numbers from a queue of integers

public static List<Integer> getEvenNumbers(Queue<Integer> q) {

List<Integer> evens = new ArrayList<>();

for (int num : q) {

if (num % 2 == 0) {

evens.add(num);

}

}

return evens;

}

public static void main(String[] args) {

// --------- Part 1: Bank Queue Simulation -----------

Queue<String> bankQueue = new LinkedList<>();

bankQueue.add("Alice");

bankQueue.add("Bob");

bankQueue.add("Charlie");

bankQueue.add("David");

bankQueue.add("Eve");

System.out.println("Initial Bank Queue: " + bankQueue);

while (!bankQueue.isEmpty()) {

String servedCustomer = bankQueue.poll(); // serve customer

System.out.println("Served: " + servedCustomer);

System.out.println("Remaining Queue: " + bankQueue);

}

// --------- Part 2: Task Manager -----------

Queue<String> tasks = new LinkedList<>();

tasks.add("Email client");

tasks.add("Prepare report");

tasks.add("Team meeting");

tasks.add("Code review");

System.out.println("\nTask Queue: " + tasks);

System.out.println("Next task to do: " + tasks.peek()); // peek

while (!tasks.isEmpty()) {

String completedTask = tasks.poll();

System.out.println("Completed Task: " + completedTask);

System.out.println("Remaining Tasks: " + tasks);

}

// --------- Part 3: Even Numbers from Integer Queue -----------

Queue<Integer> numbers = new LinkedList<>();

numbers.add(10);

numbers.add(15);

numbers.add(20);

numbers.add(33);

numbers.add(42);

List<Integer> evenNumbers = getEvenNumbers(numbers);

System.out.println("\nEven numbers from queue: " + evenNumbers);

}

}

19. package Day8;

import java.util.ArrayList;

import java.util.Scanner;

public class Remove\_element {

public static void main(String[] args) {

// Create an ArrayList of Strings

ArrayList<String> fruits = new ArrayList<>();

fruits.add("Apple");

fruits.add("Banana");

fruits.add("Mango");

fruits.add("Orange");

fruits.add("Grapes");

// Display initial list

System.out.println("Fruits List: " + fruits);

// Ask user for fruit to remove

Scanner sc = new Scanner(System.in);

System.out.print("Enter the fruit to remove: ");

String fruitToRemove = sc.nextLine();

// Remove the fruit

if(fruits.remove(fruitToRemove)) {

System.out.println(fruitToRemove + " removed successfully.");

} else {

System.out.println(fruitToRemove + " not found in the list.");

}

// Display updated list

System.out.println("Updated Fruits List: " + fruits);

sc.close();

}

}

20. package Day8;

import java.util.ArrayList;

import java.util.Collections;

public class ReverseArrayList {

public static void main(String[] args) {

// Create an ArrayList of characters

ArrayList<Character> chars = new ArrayList<>();

chars.add('A');

chars.add('B');

chars.add('C');

chars.add('D');

chars.add('E');

// Display original list

System.out.println("Original List: " + chars);x

// Reverse the list

Collections.reverse(chars);

// Display reversed list

System.out.println("Reversed List: " + chars);

}

}

21. package Day8;

import java.util.ArrayList;

import java.util.Collections;

public class SortArrayList {

public static void main(String[] args) {

// Create an ArrayList of integers

ArrayList<Integer> numbers = new ArrayList<>();

numbers.add(45);

numbers.add(12);

numbers.add(78);

numbers.add(5);

numbers.add(34);

numbers.add(23);

numbers.add(56);

// Display original list

System.out.println("Original List: " + numbers);

// Sort the list in ascending order

Collections.sort(numbers);

// Display sorted list

System.out.println("Sorted List: " + numbers);

}

}

22. package Day8;

import java.util.Scanner;

import java.util.Stack;

public class StackDemo {

// Method to reverse a string using Stack

public static String reverseString(String str) {

Stack<Character> stack = new Stack<>();

for(char ch : str.toCharArray()) {

stack.push(ch);

}

StringBuilder reversed = new StringBuilder();

while(!stack.isEmpty()) {

reversed.append(stack.pop());

}

return reversed.toString();

}

// Method to check balanced parentheses using Stack

public static boolean isBalanced(String expr) {

Stack<Character> stack = new Stack<>();

for(char ch : expr.toCharArray()) {

if(ch == '(') {

stack.push(ch);

} else if(ch == ')') {

if(stack.isEmpty()) return false;

stack.pop();

}

}

return stack.isEmpty();

}

// Method to convert decimal to binary using Stack

public static String decimalToBinary(int num) {

Stack<Integer> stack = new Stack<>();

while(num > 0) {

stack.push(num % 2);

num /= 2;

}

StringBuilder binary = new StringBuilder();

while(!stack.isEmpty()) {

binary.append(stack.pop());

}

return binary.length() > 0 ? binary.toString() : "0";

}

public static void main(String[] args) {

Scanner sc = new Scanner(System.in);

// --------- Part 1: Stack of Integers -----------

Stack<Integer> stack = new Stack<>();

stack.push(10);

stack.push(20);

stack.push(30);

stack.push(40);

stack.push(50);

System.out.println("Original Stack: " + stack);

// Pop top element

int popped = stack.pop();

System.out.println("Popped element: " + popped);

System.out.println("Stack after pop: " + stack);

// Peek current top

System.out.println("Current top element: " + stack.peek());

// Check if stack is empty

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

// --------- Part 2: Reverse a String -----------

System.out.print("\nEnter a string to reverse: ");

String inputStr = sc.nextLine();

System.out.println("Reversed string: " + reverseString(inputStr));

// --------- Part 3: Balanced Parentheses -----------

System.out.print("\nEnter an expression to check parentheses: ");

String expr = sc.nextLine();

if(isBalanced(expr)) {

System.out.println("The expression is Valid.");

} else {

System.out.println("The expression is Invalid.");

}

// --------- Part 4: Decimal to Binary -----------

System.out.print("\nEnter a decimal number to convert to binary: ");

int decimal = sc.nextInt();

System.out.println("Binary representation: " + decimalToBinary(decimal));

sc.close();

}

}

23. package Day8;

import java.util.ArrayList;

class Student {

int id;

String name;

double marks;

// Constructor

public Student(int id, String name, double marks) {

this.id = id;

this.name = name;

this.marks = marks;

}

// Method to display student details

public void displayDetails() {

System.***out***.println("ID: " + id + ", Name: " + name + ", Marks: " + marks);

}

}

public class StudentArrayList {

public static void main(String[] args) {

// Create an ArrayList of Student objects

ArrayList<Student> students = new ArrayList<>();

// Add students

students.add(new Student(101, "Alice", 85.5));

students.add(new Student(102, "Bob", 92.0));

students.add(new Student(103, "Charlie", 78.0));

// Display student details using a loop

System.***out***.println("Student Details:");

for (Student s : students) {

s.displayDetails();

}

}

}

24. package Day8;

import java.util.TreeSet;

import java.util.Comparator;

public class TreeSetDemo {

public static void main(String[] args) {

// -------- Part 1: TreeSet of Strings (natural order) -----------

TreeSet<String> countries = new TreeSet<>();

countries.add("India");

countries.add("Brazil");

countries.add("Canada");

countries.add("Australia");

countries.add("Denmark");

System.out.println("TreeSet of countries (sorted): " + countries);

// -------- Part 2: TreeSet of Integers -----------

TreeSet<Integer> numbers = new TreeSet<>();

numbers.add(50);

numbers.add(10);

numbers.add(30);

numbers.add(70);

numbers.add(20);

System.out.println("\nTreeSet of numbers (sorted): " + numbers);

System.out.println("First element: " + numbers.first());

System.out.println("Last element: " + numbers.last());

int reference = 30;

System.out.println("Element lower than " + reference + ": " + numbers.lower(reference));

System.out.println("Element higher than " + reference + ": " + numbers.higher(reference));

// -------- Part 3: TreeSet with custom comparator (reverse order) -----------

TreeSet<String> reverseCountries = new TreeSet<>(Comparator.reverseOrder());

reverseCountries.add("India");

reverseCountries.add("Brazil");

reverseCountries.add("Canada");

reverseCountries.add("Australia");

reverseCountries.add("Denmark");

System.out.println("\nTreeSet of countries (reverse order): " + reverseCountries);

}

}

25. package Day8;

import java.util.ArrayList;

import java.util.Iterator;

public class IterateArrayList {

public static void main(String[] args) {

// Create an ArrayList of cities

ArrayList<String> cities = new ArrayList<>();

cities.add("New York");

cities.add("London");

cities.add("Paris");

cities.add("Tokyo");

cities.add("Sydney");

// Create an Iterator

Iterator<String> it = cities.iterator();

// Use Iterator to display each city

System.out.println("Cities in the list:");

while (it.hasNext()) {

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

}

}

}

26. package Day8;

import java.util.ArrayList;

public class UpdateArrayList {

public static void main(String[] args) {

// Create an ArrayList of subjects

ArrayList<String> subjects = new ArrayList<>();

subjects.add("Math");

subjects.add("Physics");

subjects.add("Chemistry");

subjects.add("Biology");

subjects.add("English");

// Display original list

System.***out***.println("Original List: " + subjects);

// Update "Math" to "Statistics"

int index = subjects.indexOf("Math");

if (index != -1) {

subjects.set(index, "Statistics");

System.***out***.println("\"Math\" has been replaced with \"Statistics\".");

} else {

System.***out***.println("\"Math\" not found in the list.");

}

// Display updated list

System.***out***.println("Updated List: " + subjects);

}

}

27. package Day8;

import java.util.Vector;

import java.util.Enumeration;

public class VectorDemo {

// Method to calculate sum of elements in a Vector<Integer>

public static int sumVector(Vector<Integer> vec) {

int sum = 0;

for(int num : vec) {

sum += num;

}

return sum;

}

public static void main(String[] args) {

//Vector of integers

Vector<Integer> numbers = new Vector<>();

numbers.add(10);

numbers.add(20);

numbers.add(30);

numbers.add(40);

numbers.add(50);

System.out.println("Original Vector: " + numbers);

// Insert an element at 3rd position (index 2)

numbers.add(2, 25);

System.out.println("After inserting 25 at index 2: " + numbers);

// Remove the 2nd element (index 1)

numbers.remove(1);

System.out.println("After removing element at index 1: " + numbers);

// Display elements using Enumeration

System.out.println("Elements using Enumeration:");

Enumeration<Integer> en = numbers.elements();

while(en.hasMoreElements()) {

System.out.print(en.nextElement() + " ");

}

System.out.println("\nSum of elements: " + sumVector(numbers));

// ----------- Part 2: Vector of Strings -----------

Vector<String> names = new Vector<>();

names.add("Alice");

names.add("Bob");

names.add("Charlie");

names.add("David");

System.out.println("\nOriginal Names Vector: " + names);

// Check if a specific name exists

String searchName = "Bob";

if(names.contains(searchName)) {

System.out.println(searchName + " is present in the vector.");

} else {

System.out.println(searchName + " is not found.");

}

// Replace one name with another

int index = names.indexOf("Charlie");

if(index != -1) {

names.set(index, "Eve");

}

System.out.println("After replacing 'Charlie' with 'Eve': " + names);

// Clear all elements

names.clear();

System.out.println("After clearing, Names Vector: " + names);

// ----------- Part 3: Copy and Compare Vectors -----------

Vector<Integer> vec1 = new Vector<>();

vec1.add(1);

vec1.add(2);

vec1.add(3);

Vector<Integer> vec2 = new Vector<>();

vec2.addAll(vec1); // Copy elements

System.out.println("\nVector1: " + vec1);

System.out.println("Vector2 (copied from Vector1): " + vec2);

if(vec1.equals(vec2)) {

System.out.println("Both vectors are equal.");

} else {

System.out.println("Vectors are not equal.");

}

}

}