# LAB # 02

**LAB TASKS 1:**

import java.util.Vector;

public class VectorExample {

public static void main(String[] args) {

// Initialize the Vector with 10 integers

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

// Adding 10 integers to the Vector

for (int i = 1; i <= 10; i++) {

numbers.add(i);

}

// Display the integers in the Vector

System.out.println("The integers in the Vector are:");

for (int number : numbers) {

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

}

// Calculate and display the sum of integers

int sum = 0;

for (int number : numbers) {

sum += number;

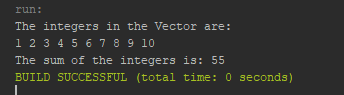
}

System.out.println("\nThe sum of the integers is: " + sum);

}

}

**OUTPUT**



**LAB TASKS 2:**

import java.util.ArrayList;

import java.util.Scanner;

public class Arraylist {

// Function to display all elements of the ArrayList

public static void displayElements(ArrayList<String> list) {

System.out.println("The elements in the ArrayList are:");

for (String str : list) {

System.out.println(str);

}

}

// Function to find the largest string in the ArrayList

public static String findLargestString(ArrayList<String> list) {

if (list.isEmpty()) {

return null;

}

String largest = list.get(0);

// Iterate through the list and find the largest string by length

for (String str : list) {

if (str.length() > largest.length()) {

largest = str;

}

}

return largest;

}

public static void main(String[] args) {

// Create an ArrayList of strings

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

// Add some strings to the list

stringList.add("Pakistan");

stringList.add("Bangladesh");

stringList.add("Bangkok");

stringList.add("Malaysia");

stringList.add("Swizerland");

Scanner sc = new Scanner(System.in);

int choice;

do {

System.out.println("\nMenu:");

System.out.println("1. Display all elements");

System.out.println("2. Display the largest string");

System.out.println("3. Exit");

System.out.print("Enter your choice: ");

choice = sc.nextInt();

sc.nextLine(); // Consume newline left-over

switch (choice) {

case 1:

displayElements(stringList);

break;

case 2:

String largestString = findLargestString(stringList);

if (largestString != null) {

System.out.println("The largest string is: " + largestString);

} else {

System.out.println("The list is empty.");

}

break;

case 3:

System.out.println("Exiting...");

break;

default:

System.out.println("Invalid choice! Please try again.");

}

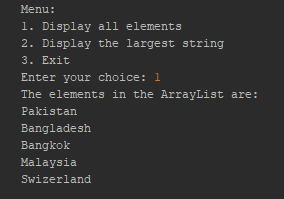
} while (choice != 3);

sc.close();

}

}

**OUTPUT**



**LAB TASKS 3:**

import java.util.ArrayList;

import java.util.Collections;

import java.util.Comparator;

// Employee class implementing Comparable interface

class Employee implements Comparable<Employee> {

int empId;

String empName;

int yearOfJoining;

// Constructor to initialize employee details

public Employee(int empId, String empName, int yearOfJoining) {

this.empId = empId;

this.empName = empName;

this.yearOfJoining = yearOfJoining;

}

// Implementing the compareTo method to sort by year of joining

@Override

public int compareTo(Employee other) {

return Integer.compare(this.yearOfJoining, other.yearOfJoining); // Corrected comparison

}

// Display employee details

@Override

public String toString() {

return empName + " (ID: " + empId + ", Year of Joining: " + yearOfJoining + ")";

}

}

public class EmployeeList {

public static void main(String[] args) {

// Create an ArrayList to store employee objects

ArrayList<Employee> employees = new ArrayList<>();

// Adding employee details to the list

employees.add(new Employee(101, "Alice", 2015));

employees.add(new Employee(102, "Bob", 2020));

employees.add(new Employee(103, "Charlie", 2018));

employees.add(new Employee(104, "Diana", 2019));

employees.add(new Employee(105, "Eve", 2016));

// Sorting employees by year of joining using Comparable

Collections.sort(employees); // Sorts by Year of Joining as per Comparable interface

System.out.println("Employees sorted by Year of Joining:");

for (Employee emp : employees) {

System.out.println(emp);

}

// Sorting employees by name using Comparator

Collections.sort(employees, new Comparator<Employee>() {

@Override

public int compare(Employee e1, Employee e2) {

return e1.empName.compareTo(e2.empName); // Corrected string comparison for name

}

});

System.out.println("\nEmployees sorted by Name:");

for (Employee emp : employees) {

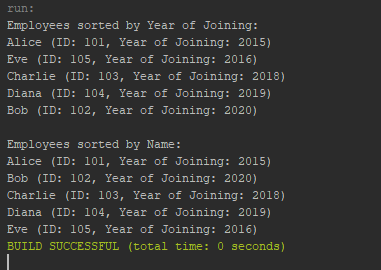
System.out.println(emp);

}

}

}

**OUTPUT**



**LAB TASKS 4:**

import java.util.Vector;

public class VectorExample1 {

public static void main(String[] args) {

// Initialize the Vector with 10 integers

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

// Adding 10 integers to the Vector

for (int i = 1; i <= 10; i++) {

numbers.add(i \* 10); // Adding multiples of 10 for demonstration

}

// Display the integers in the Vector

System.out.println("The integers in the Vector are:");

for (int number : numbers) {

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

}

// Calculate and display the sum of integers

int sum = 0;

for (int number : numbers) {

sum += number;

}

System.out.println("\nSum of the integers: " + sum);

// Find and display the maximum element

int max = numbers.get(0);

for (int number : numbers) {

if (number > max) {

max = number;

}

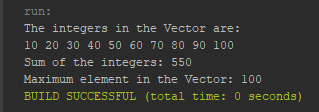
}

System.out.println("Maximum element in the Vector: " + max);

}

}

**OUTPUT**



**LAB TASKS 5:**

import java.util.ArrayList;

import java.util.Collections;

public class KthSmallestElement {

public static void main(String[] args) {

// Create a sorted ArrayList of integers

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

// Add elements to the ArrayList

numbers.add(3);

numbers.add(8);

numbers.add(15);

numbers.add(17);

numbers.add(24);

numbers.add(30);

numbers.add(42);

// Display the sorted ArrayList

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

// Define k

int k = 4; // We want to find the 4th smallest element

// Check if k is valid

if (k > 0 && k <= numbers.size()) {

// Since the ArrayList is already sorted, we can directly access the k-th smallest element

int kthSmallest = numbers.get(k - 1); // k-1 because index starts from 0

System.out.println("The " + k + "-th smallest element is: " + kthSmallest);

} else {

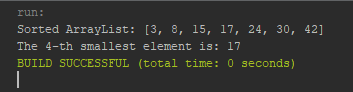
System.out.println("Invalid value of k. It should be between 1 and " + numbers.size());

}

}

}

**OUTPUT**



**HOME TASK 1:**

import java.util.ArrayList;

public class MergedArrayList {

public static void main(String[] args) {

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

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

list1.add(1);

list1.add(3);

list1.add(5);

list2.add(2);

list2.add(4);

list2.add(6);

ArrayList<Integer> MergedList = new ArrayList<>(list1);

MergedList.addAll(list2);

System.out.println("Merged List:"+ MergedList);

}

}

**OUTPUT**



**HOME TASK 2:**

import java.util.Scanner;

public class HashCodeExample {

public static void main(String[] args) {

Scanner scanner = new Scanner(System.in);

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

String userInput = scanner.nextLine();

// Calculate and display the hash code value

int hashCodeValue = userInput.hashCode();

System.out.println("Hash code value of the input \"" + userInput + "\": " + hashCodeValue);

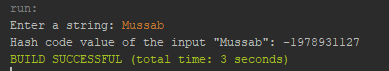
// Close the scanner

scanner.close();

}

}

**OUTPUT**



**HOME TASK 3:**

import java.util.Objects;

public class Employee {

private String name;

private int id;

// Constructor

public Employee(String name, int id) {

this.name = name;

this.id = id;

}

// Getters

public String getName() {

return name;

}

public int getId() {

return id;

}

// Override hashCode() method

@Override

public int hashCode() {

return Objects.hash(name, id);

}

// Override equals() method

@Override

public boolean equals(Object obj) {

if (this == obj) return true;

if (obj == null || getClass() != obj.getClass()) return false;

Employee employee = (Employee) obj;

return id == employee.id && Objects.equals(name, employee.name);

}

@Override

public String toString() {

return "Employee{" +

"name='" + name + '\'' +

", id=" + id +

'}';

}

}

import java.util.HashSet;

import java.util.Scanner;

public class EmployeeManagement {

private HashSet<Employee> employees;

// Constructor

public EmployeeManagement() {

employees = new HashSet<>();

}

// Add new employee

public boolean addEmployee(String name, int id) {

Employee newEmployee = new Employee(name, id);

return employees.add(newEmployee); // Returns true if added, false if it already exists

}

// Check if an employee exists

public boolean employeeExists(String name, int id) {

Employee emp = new Employee(name, id);

return employees.contains(emp);

}

// Display all employees

public void displayEmployees() {

if (employees.isEmpty()) {

System.out.println("No employees found.");

} else {

System.out.println("Employee Records:");

for (Employee employee : employees) {

System.out.println(employee);

}

}

}

public static void main(String[] args) {

EmployeeManagement management = new EmployeeManagement();

Scanner scanner = new Scanner(System.in);

String choice;

do {

System.out.println("\nEmployee Management System");

System.out.println("1. Add Employee");

System.out.println("2. Check Employee Exists");

System.out.println("3. Display All Employees");

System.out.println("4. Exit");

System.out.print("Choose an option: ");

choice = scanner.nextLine();

switch (choice) {

case "1":

System.out.print("Enter employee name: ");

String name = scanner.nextLine();

System.out.print("Enter employee ID: ");

int id = Integer.parseInt(scanner.nextLine());

if (management.addEmployee(name, id)) {

System.out.println("Employee added successfully.");

} else {

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

}

break;

case "2":

System.out.print("Enter employee name: ");

String checkName = scanner.nextLine();

System.out.print("Enter employee ID: ");

int checkId = Integer.parseInt(scanner.nextLine());

if (management.employeeExists(checkName, checkId)) {

System.out.println("Employee exists.");

} else {

System.out.println("Employee does not exist.");

}

break;

case "3":

management.displayEmployees();

break;

case "4":

System.out.println("Exiting...");

break;

default:

System.out.println("Invalid choice. Please try again.");

}

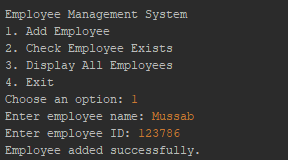
} while (!choice.equals("4"));

scanner.close();

}

}

**OUTPUT**



**HOME TASK 4:**

import java.util.Objects;

public class Color {

private int red;

private int green;

private int blue;

// Constructor

public Color(int red, int green, int blue) {

this.red = red;

this.green = green;

this.blue = blue;

}

// Getters

public int getRed() {

return red;

}

public int getGreen() {

return green;

}

public int getBlue() {

return blue;

}

// Override equals() method

@Override

public boolean equals(Object obj) {

if (this == obj) return true; // Check if both references are the same

if (obj == null || getClass() != obj.getClass()) return false; // Check for null or different class

Color color = (Color) obj; // Cast the object to Color

return red == color.red && green == color.green && blue == color.blue; // Compare RGB values

}

// Override hashCode() method

@Override

public int hashCode() {

return Objects.hash(red, green, blue); // Create hash code based on RGB values

}

@Override

public String toString() {

return "Color{" +

"red=" + red +

", green=" + green +

", blue=" + blue +

'}';

}

}

public class colortest {

public static void main(String[] args) {

// Create some Color objects

Color color1 = new Color(255, 0, 0); // Red

Color color2 = new Color(255, 0, 0); // Red

Color color3 = new Color(0, 255, 0); // Green

// Check if color1 is equal to color2

System.out.println("color1 equals color2: " + color1.equals(color2)); // true

// Check if color1 is equal to color3

System.out.println("color1 equals color3: " + color1.equals(color3)); // false

// Display the colors

System.out.println("Color 1: " + color1);

System.out.println("Color 2: " + color2);

System.out.println("Color 3: " + color3);

}

}

**OUTPUT**

