**Batch: C3 Roll No.: 16010123217**

**Experiment / assignment / tutorial No.06**

**Grade: AA / AB / BB / BC / CC / CD /DD**

**Signature of the Staff In-charge with date**

|  |
| --- |
| **TITLE :Collection Framework** |

**AIM:** Create a class Employee which stores E-Name, E-Id and E-Salary of an Employee. Use class Vector to maintain an array of Employee with respect to the E-Salary. Provide the following functions

1) Create (): this function will accept the n Employee records in any order and will arrange them in the sorted order.

2) Insert (): to insert the given Employee record at appropriate index in the vector depending upon the E-Salary.

3) delete ByE-name( ): to accept the name of the Employee and delete the record having given name

4) deleteByE-Id ( ): to accept the Id of the Employee and delete the record having given E-Id.

Provide the following functions

1. boolean add(E e) : This method appends the specified element to the end of this Vector.
2. void addElement(E obj) This method adds the specified component to the end of this vector, increasing its size by one.
3. int lastIndexOf(Object o, int index) This method returns the index of the last occurrence of the specified element in this vector, searching backwards from index, or returns -1 if the element is not found.
4. void removeElementAt(int index)This method deletes the component at the specified index.

**\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

**Expected OUTCOME of Experiment:**

**CO2:** Explore arrays, vectors, classes and objects in C++ and Java.

**\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

**Books/ Journals/ Websites referred:**

1. Ralph Bravaco , Shai Simoson , “Java Programing From the Group Up” Tata McGraw-Hill.

2.Grady Booch, Object Oriented Analysis and Design .

**\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

**Pre Lab/ Prior Concepts:**

Vectors in Java are one of the most commonly used data structures. Similar to Arrays data structures which hold the data in a linear fashion. Vectors also store the data in a linear fashion, but unlike Arrays, they do not have a fixed size. Instead, their size can be increased on demand.

Vector class is a child class of AbstractList class and implements on List interface. To use Vectors, we first have to import Vector class from java.util package:

import java.util.Vector;

**Access Elements in Vector:**

We can access the data members simply by using the index of the element, just like we access the elements in Arrays.

Example- If we want to access the third element in a vector v, we simply refer to it as v[3].

**Vectors Constructors**

Listed below are the multiple variations of vector [constructors](https://www.edureka.co/blog/constructor-in-java/) available to use:

1. **Vector(int initialCapacity, int Increment)** – Constructs a vector with given initialCapacity and its Increment in size.
2. **Vector(int initialCapacity)*–***Constructs an empty vector with given initialCapacity. In this case, Increment is zero.
3. **Vector()** – Constructs a default vector of capacity 10.
4. **Vector(Collection c)*–***Constructs a vector with a given collection, the order of the elements is same as returned by the collection’s iterator.

There are also three protected parameters in vectors

* + **Int capacityIncrement()-** It automatically increases the capacity of the vector when the size becomes greater than capacity.
  + **Int elementCount()** – tell number of elements in the vector
  + **Object[] elementData()** – array in which elements of vector are stored

**Memory allocation of vectors:**

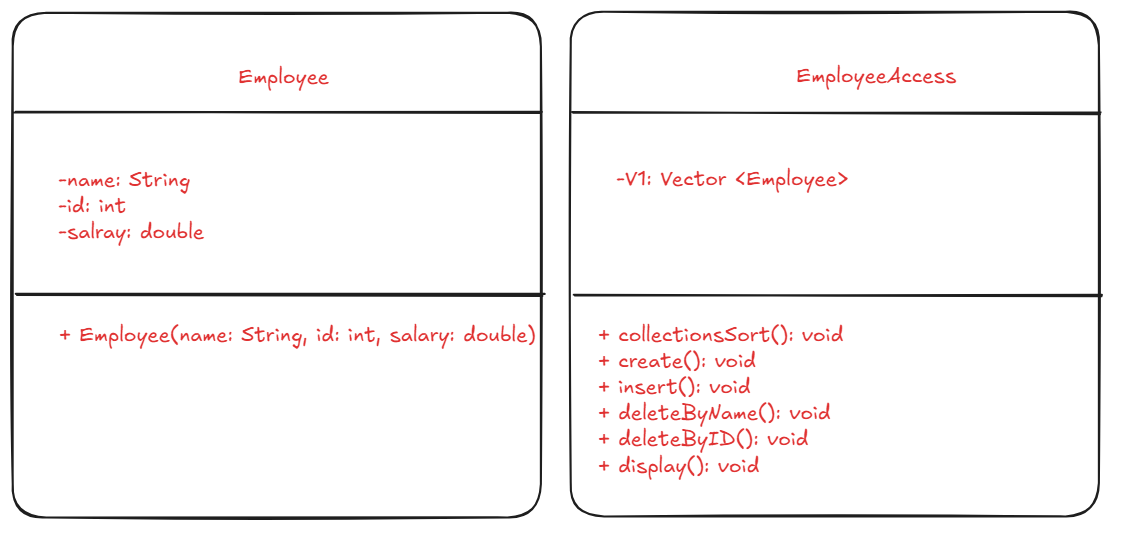
Vectors do not have a fixed size, instead, they have the ability to change their size dynamically. One might think that the vectors allocate indefinite long space to store objects. But this is not the case. Vectors can change their size based on two fields ‘capacity’ and ‘capacityIncrement’. Initially, a size equal to ‘capacity’ field is allocated when a vector is declared. We can insert the elements equal to the capacity. But as soon as the next element is inserted, it increases the size of the array by size ‘capacityIncrement’. Hence, it is able to change its size dynamically.

For a default constructor, the capacity is doubled whenever the capacity is full and a new element is to be inserted.

**Methods of Vectors :**

* Adding elements
* Removing elements
* Changing elements
* Iterating the vector

**Class Diagram:**



**Algorithm:**

1. Start

1.1. Initialize `EmployeeAccess` object.

1.2. Create a list of employees.

2. Create Function

2.1. Input the number of employees `n`.

2.2. For each employee:

- Input name, ID, and salary.

- Add employee to the list.

- Sort employees by salary using Collection sort.

3. Insert Function

3.1. Input employee name, ID, and salary.

3.2. Add the employee to the list.

3.3. Sort the employees by salary.

4. Delete Function

4.1. Input a choice: delete by name or by ID.

4.2. If by name, search and remove the employee with the given name.

4.3. If by ID, search and remove the employee with the given ID.

5. Display Function

5.1. For each employee, print their details (name, ID, salary).

6. Loop

6.1. Continue asking for input (insert, delete, display) until the user exits.

7. End

**Implementation details:**

import java.util.*\**;

class Employee {

    String name;

    int id;

    double salary;

    public Employee(String name, int id, double salary) {

*this*.name = name;

*this*.id = id;

*this*.salary = salary;

    }

}

class EmployeeAccess {

    Vector<Employee> V1 = new Vector<>();

    Scanner sc = new Scanner(System.in);

    public void Sort() {

        Collections.sort(V1, new Comparator<Employee>() {

            public int compare(Employee e1, Employee e2) {

                return Double.compare(e1.salary, e2.salary);

            }

        });

    }

*/\**

*//If sorting is done using bubble sort*

*public void Sort() {*

*for (int i = 0; i < V1.size(); i++) {*

*for (int j = 0; j < V1.size() - 1 - i; j++) {*

*if (V1.get(j).salary > V1.get(j + 1).salary) {*

*Employee temp = V1.get(j);*

*V1.set(j, V1.get(j + 1));*

*V1.set(j + 1, temp);*

*}*

*}*

*}*

*}*

*\*/*

    public void create() {

        System.out.println("Please enter the number of employees you would like to add:");

        int n = sc.nextInt();

        for (int i = 0; i < n; i++) {

            System.out.println("Enter Employee Name:");

            String name = sc.next();

            System.out.println("Enter Employee ID (positive number):");

            int id = sc.nextInt();

            System.out.println("Enter Employee Salary (positive number):");

            double salary = sc.nextDouble();

            if (id <= 0 || salary <= 0) {

                System.out.println("Invalid ID or Salary. Please try again.");

                i--;

                continue;

            }

            Employee emp = new Employee(name, id, salary);

            V1.add(emp);

        }

        Sort();

    }

    public void insert() {

        System.out.println("Enter Employee Name:");

        String name = sc.next();

        System.out.println("Enter Employee ID (positive number):");

        int id = sc.nextInt();

        System.out.println("Enter Employee Salary (positive number):");

        double salary = sc.nextDouble();

        if (id <= 0 || salary <= 0) {

            System.out.println("Invalid ID or Salary. Try again.");

            return;

        }

        Employee emp = new Employee(name, id, salary);

        V1.add(emp);

        Sort();

    }

    public void deleteByName() {

        System.out.println("Enter the name of the employee to be deleted:");

        String name = sc.next();

        boolean found = false;

        for (int i = 0; i < V1.size(); i++) {

            if (V1.get(i).name.equals(name)) {

                V1.remove(i);

                found = true;

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

                break;

            }

        }

        if (!found) {

            System.out.println("Employee with the name " + name + " not found.");

        }

    }

    public void deleteById() {

        System.out.println("Enter the ID of the employee to be deleted:");

        int id = sc.nextInt();

        boolean found = false;

        for (int i = 0; i < V1.size(); i++) {

            if (V1.get(i).id == id) {

                V1.remove(i);

                found = true;

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

                break;

            }

        }

        if (!found) {

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

        }

    }

    public void display() {

        if (V1.isEmpty()) {

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

        } else {

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

            for (Employee e : V1) {

                System.out.println("Name: " + e.name + ", ID: " + e.id + ", Salary: " + e.salary);

            }

        }

    }

    public static void main(String[] args) {

        EmployeeAccess E1 = new EmployeeAccess();

        boolean exit = false;

        while (!exit) {

            System.out.println("\nChoose an option:");

            System.out.println("1. Create Employee List");

            System.out.println("2. Insert a new Employee");

            System.out.println("3. Delete Employee by Name");

            System.out.println("4. Delete Employee by ID");

            System.out.println("5. Display All Employees");

            System.out.println("6. Exit");

            Scanner sc = new Scanner(System.in);

            int choice = sc.nextInt();

            switch (choice) {

                case 1:

                    E1.create();

                    break;

                case 2:

                    E1.insert();

                    break;

                case 3:

                    E1.deleteByName();

                    break;

                case 4:

                    E1.deleteById();

                    break;

                case 5:

                    E1.display();

                    break;

                case 6:

                    exit = true;

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

                    break;

                default:

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

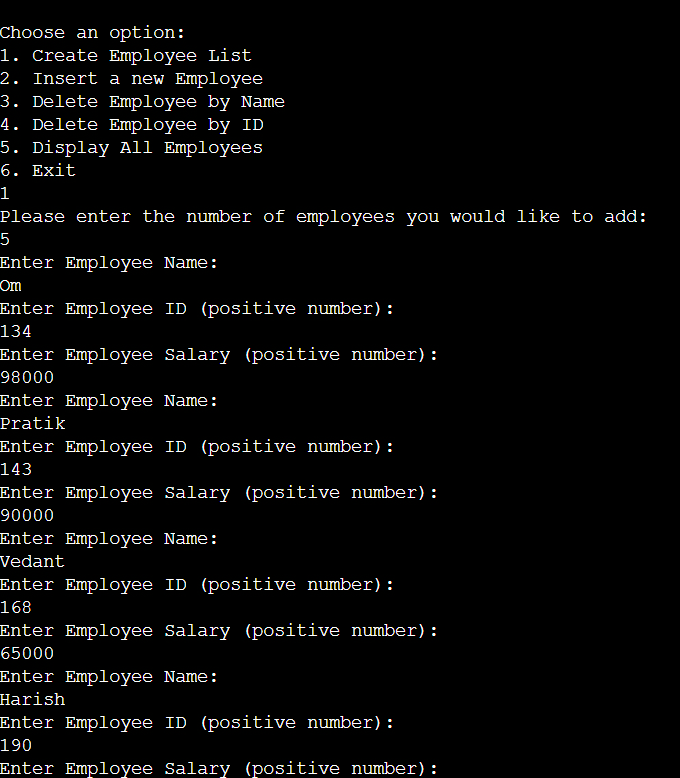
                    break;

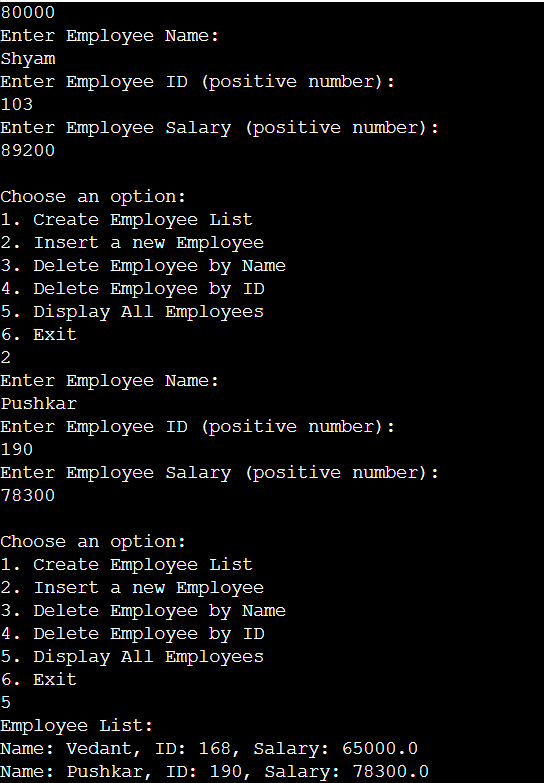
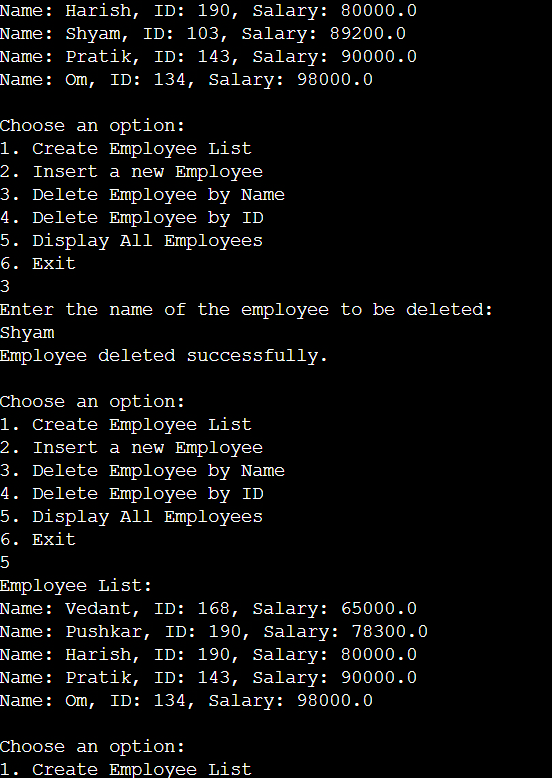
            }

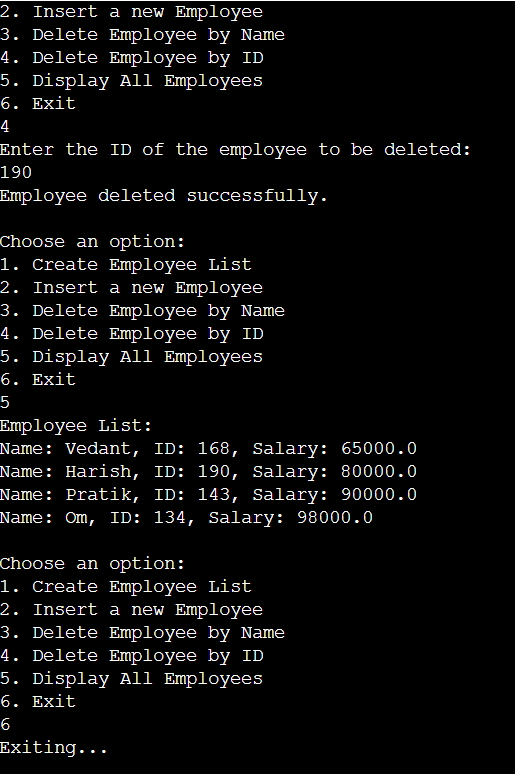
        }

    }

}

**Output:**



**Conclusion:**

**By performing this experiment I learnt about vectors in Java**

**Date: 12/09/24 Signature of faculty in-charge**

**Post Lab Descriptive Questions**

1. **Write a note on the collection framework.**

**Ans.** Java Collections Framework provides lots of different useful data types, such as linked lists (allows insertion anywhere in constant time), re-sizeable array lists.

Java Collections Framework provides abstractions, so you can refer to a list as a List, whether backed by an array list or a linked list; and you can refer to a map/dictionary as a Map, whether backed by a red-black tree or a hashtable.

Java Collections Framework allows you to use the right data structure, because one size does not fit all.

1. **Explain any 10 methods of Vector class in detail with the help of example**

**Ans.** public void addElement(Object obj)

Adds the specified component to the end of this vector, increasing its size by one. The capacity of this vector is increased if its size becomes greater than its capacity.

e.g. v.addElement(item);

public boolean removeElement(Object obj)

Removes the first (lowest-indexed) occurrence of the argument from this vector. If the object is found in this vector, each component in the vector with an index greater or equal to the object's index is shifted downward to have an index one smaller than the value it had previously.

e.g. v.removeElement(item)

public Object firstElement()

Returns the first component (the item at index 0) of this vector.

e.g. v.firstElement();

public Object lastElement()

Returns the last component of the vector.

e.g. v.lastElement();

public boolean isEmpty()

Tests if this vector has no components.

e.g. v.isEmpty();

v.elementAt(n)

Gives the name of the nth object.

v.removeElementAt(n)

Removes the item stored in the nth position of the list.

v.removeAllElements()

Removes all the elements in the vector.

v.insertElementAt(item,n)

Inserts the item at nth position.

v.copyInto(array)

Copies all items from vector to array.

1. **What is an Arraylist? How does it differ from the array?**

**Ans.** An ArrayList in Java is a part of the Java Collections Framework and represents a resizable array, which means it can grow and shrink in size dynamically based on the number of elements added or removed. It is implemented as a class in the java.util package.

The primary difference between an array and an ArrayList is size flexibility. Arrays are of fixed length, meaning once created, their size cannot be changed. In contrast, an ArrayList is dynamic and adjusts its size automatically when elements are added or removed. Arrays can store both primitive data types (such as int, char, etc.) and objects, while an ArrayList only stores objects, requiring the use of wrapper classes like Integer or Double for primitive types.

1. **Implement a menu driven program for the following:**

**Accepts a shopping list (name, price and quantity)from the command line and stores them in a vector.**

**To delete specific item (given by user) in the vector**

**Add item at the end of the vector**

**Add item at specific location**

**Print the contents of vector using enumeration interface.**

**Ans.**

import java.util.*\**;

class ShoppingItem {

    String name;

    double price;

    int quantity;

    public ShoppingItem(String name, double price, int quantity) {

*this*.name = name;

*this*.price = price;

*this*.quantity = quantity;

    }

    public String toString() {

        return "Item: " + name + ", Price: " + price + ", Quantity: " + quantity;

    }

}

public class ShoppingList {

    public static void main(String[] args) {

        Vector<ShoppingItem> shoppingList = new Vector<>();

        Scanner scanner = new Scanner(System.in);

        int choice;

        do {

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

            System.out.println("2. Delete item");

            System.out.println("3. Add item at specific location");

            System.out.println("4. Print list");

            System.out.println("5. Exit");

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

            choice = scanner.nextInt();

            scanner.nextLine();

            switch (choice) {

                case 1:

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

                    String name = scanner.nextLine();

                    System.out.print("Enter item price: ");

                    double price = scanner.nextDouble();

                    System.out.print("Enter item quantity: ");

                    int quantity = scanner.nextInt();

                    shoppingList.add(new ShoppingItem(name, price, quantity));

                    break;

                case 2:

                    System.out.print("Enter item name to delete: ");

                    String deleteName = scanner.nextLine();

                    for (int i = 0; i < shoppingList.size(); i++) {

                        if (shoppingList.get(i).name.equals(deleteName)) {

                            shoppingList.remove(i);

                            break;

                        }

                    }

                    break;

                case 3:

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

                    String newName = scanner.nextLine();

                    System.out.print("Enter item price: ");

                    double newPrice = scanner.nextDouble();

                    System.out.print("Enter item quantity: ");

                    int newQuantity = scanner.nextInt();

                    System.out.print("Enter position to insert at: ");

                    int position = scanner.nextInt();

                    shoppingList.add(position, new ShoppingItem(newName, newPrice, newQuantity));

                    break;

                case 4:

                    Enumeration<ShoppingItem> items = shoppingList.elements();

                    while (items.hasMoreElements()) {

                        System.out.println(items.nextElement());

                    }

                    break;

                case 5:

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

                    break;

                default:

                    System.out.println("Invalid choice.");

            }

        } while (choice != 5);

        scanner.close();

    }

}

Output:

