Batch: D3	**Roll No.: 16010123294**
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:

6.1 Vector Implementation

Create a class Employee which stores E-Name, E-Id and E-Salary of an Employee. Use class Vector to maintain an array of Employees 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.

6.2 ArrayList Implementation:

Create a class Employee that stores E-Name, E-Id, and E-Salary of an employee. Use the class ArrayList to maintain a list of employees sorted by E-Salary. Implement the following functionalities:

1. **Create():**
This method will accept n employee records in any order and arrange them in the sorted order based on E-Salary.
2. **Insert():**
This method will insert a given employee record into the appropriate index in the ArrayList, ensuring that the list remains sorted by E-Salary.
3. **deleteByE-name():**
This method will accept the name of an employee and delete the record associated with the given E-Name.
4. **deleteByE-Id():**
This method will accept the ID of an employee and delete the record associated with the given E-Id.

Additional Functions

- **boolean add(E e):**
This method appends the specified element to the end of the ArrayList.
- **int lastIndexOf(Object o, int index):**
This method returns the index of the last occurrence of the specified element in the ArrayList, searching backward from the specified index, or returns -1 if the element is not found.
- **void remove(int index):**
This method deletes the element at the specified index in the ArrayList.

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 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

Algorithm:

Initialize the System:

Create an instance of the EmployeeManager class.

Create a Scanner object for user input.

Set a boolean variable running to true to control the main loop.

Display Menu:

While running is true, repeat the following steps:

Print the menu options:

Create Employees

Insert New Employee

Delete Employee by Name

Delete Employee by ID

Display Employees

Exit

Prompt the user to choose an option.

Handle User Choice:

Read the user's choice and use a switch statement to determine the action:

Case 1: Create Employees

Prompt the user to enter the number of employees to create.

For each employee:

Ask for the employee's name, ID, and salary.

Create an Employee object and store it in an array.

Add the array of employees to the manager.

Print a confirmation message.

Case 2: Insert New Employee

Prompt for the new employee's name, ID, and salary.

Create an Employee object and add it to the manager.

Print a confirmation message.

Case 3: Delete Employee by Name

Prompt for the name of the employee to delete.

Call the delete method in the manager and print a confirmation message.

Case 4: Delete Employee by ID

Prompt for the ID of the employee to delete.

Call the delete method in the manager and print a confirmation message.

Case 5: Display Employees

Call the display method in the manager to show all employees.

Case 6: Exit

Set running to false to end the loop and print a goodbye message.

Default: Handle invalid input by printing an error message.

Implementation details:

6.1

```
import java.util.Vector;

import java.util.Scanner;


class Employee {

   private String name;

   private String empId;

   private double salary;


   public Employee(String name, String empId, double salary) {

       this.name = name;

       this.empId = empId;

       this.salary = salary;

   }
```

```
    public String getName() {
        return name;
    }

    public String getEmpId() {
        return empId;
    }

    public double getSalary() {
        return salary;
    }

    public String toString() {
        return "Employee{Name='" + name + "', ID='" + empId + "', Salary=" + salary + 
"}";
    }
}

class EmployeeManager {
    private Vector<Employee> employees;
```

```
public EmployeeManager() {

    employees = new Vector<>();

}


public void create(Employee[] employeeArray) {

    for (Employee emp : employeeArray) {

        employees.add(emp);

    }

}


public void insert(Employee newEmployee) {

    employees.add(newEmployee);

}


public void deleteByEName(String name) {

    employees.removeIf(emp -> emp.getName().equalsIgnoreCase(name));

}


public void deleteByEId(String empId) {

    employees.removeIf(emp -> emp.getEmpId().equalsIgnoreCase(empId));
    }

    public void displayEmployees() {
        if (employees.isEmpty()) {
            System.out.println("No employees to display.");
        } else {
            for (Employee emp : employees) {
                System.out.println(emp);
            }
        }
    }
}

public class Main {
    public static void main(String[] args) {
        EmployeeManager manager = new EmployeeManager();
        Scanner scanner = new Scanner(System.in);
        boolean running = true;

        while (running) {
            System.out.println("\nEmployee Management System");
            System.out.println("1. Create Employees");

            System.out.println("2. Insert New Employee");

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

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

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

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

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


            int choice = scanner.nextInt();

            scanner.nextLine(); // Consume newline


            switch (choice) {

                case 1:

                    System.out.print("Enter the number of employees to create: ");

                    int n = scanner.nextInt();

                    scanner.nextLine(); // Consume newline

                    Employee[] employeeArray = new Employee[n];


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

                        System.out.print("Enter name of employee " + (i + 1) + ": ");

                        String name = scanner.nextLine();
                        System.out.print("Enter ID of employee " + (i + 1) + ": ");
                        String empId = scanner.nextLine();
                        System.out.print("Enter salary of employee " + (i + 1) + ": ");
                        double salary = scanner.nextDouble();
                        scanner.nextLine(); // Consume newline
                        employeeArray[i] = new Employee(name, empId, salary);
                    }

                    manager.create(employeeArray);
                    System.out.println("Employees created successfully.");
                    break;

                case 2:
                    System.out.print("Enter name of new employee to insert: ");
                    String newName = scanner.nextLine();
                    System.out.print("Enter ID of new employee: ");
                    String newEmpId = scanner.nextLine();
                    System.out.print("Enter salary of new employee: ");
                    double newSalary = scanner.nextDouble();
                    scanner.nextLine(); // Consume newline
```

```
          Employee newEmployee = new Employee(newName, newEmpId,
newSalary);

          manager.insert(newEmployee);

          System.out.println("Employee " + newName + " inserted successfully.");

          break;


        case 3:

          System.out.print("Enter name of employee to delete by name: ");

          String deleteByName = scanner.nextLine();

          manager.deleteByEName(deleteByName);

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

          break;


        case 4:

          System.out.print("Enter ID of employee to delete by ID: ");

          String deleteById = scanner.nextLine();

          manager.deleteByEId(deleteById);

          System.out.println("Employee with ID " + deleteById + " deleted
successfully.");

          break;
```

```
            case 5:
                System.out.println("Employees:");
                manager.displayEmployees();
                break;

            case 6:
                running = false;
                System.out.println("Exiting the program. Goodbye!");
                break;

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

        scanner.close();
    }
}
```

6.2:

```
import java.util.ArrayList;

import java.util.Scanner;


class Employee {

   private String name;

   private String empId;

   private double salary;


   public Employee(String name, String empId, double salary) {

       this.name = name;

       this.empId = empId;

       this.salary = salary;

   }


   public String getName() {

       return name;

   }


   public String getEmpId() {

       return empId;
```

```
    }

    public double getSalary() {
        return salary;
    }

    public String toString() {
        return "Employee{Name='" + name + "', ID='" + empId + "', Salary=" + salary +
"}";
    }
}

class EmployeeManager {
    private ArrayList<Employee> employees;

    public EmployeeManager() {
        employees = new ArrayList<>();
    }

    public void create(Employee[] employeeArray) {
        for (Employee emp : employeeArray) {
```

```
        employees.add(emp);
    }
}

public void insert(Employee newEmployee) {
    employees.add(newEmployee);
}

public void deleteByEName(String name) {
    employees.removeIf(emp -> emp.getName().equalsIgnoreCase(name));
}

public void deleteByEId(String empId) {
    employees.removeIf(emp -> emp.getEmpId().equalsIgnoreCase(empId));
}

public void displayEmployees() {
    if (employees.isEmpty()) {
        System.out.println("No employees to display.");
    } else {
        for (Employee emp : employees) {
                System.out.println(emp);
            }
        }
    }
}

public class Main {
    public static void main(String[] args) {
        EmployeeManager manager = new EmployeeManager();
        Scanner scanner = new Scanner(System.in);
        boolean running = true;

        while (running) {
            System.out.println("\nEmployee Management System");
            System.out.println("1. Create Employees");
            System.out.println("2. Insert New Employee");
            System.out.println("3. Delete Employee by Name");
            System.out.println("4. Delete Employee by ID");
            System.out.println("5. Display Employees");
            System.out.println("6. Exit");
            System.out.print("Choose an option: ");
```

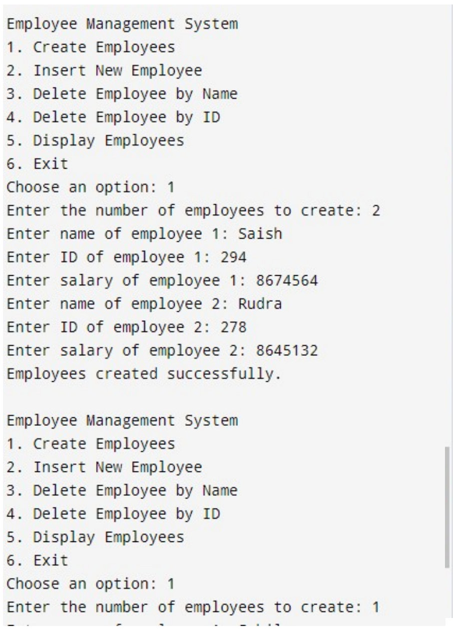
```
int choice = scanner.nextInt();
scanner.nextLine(); // Consume newline

switch (choice) {
    case 1:
        System.out.print("Enter the number of employees to create: ");
        int n = scanner.nextInt();
        scanner.nextLine(); // Consume newline
        Employee[] employeeArray = new Employee[n];

        for (int i = 0; i < n; i++) {
            System.out.print("Enter name of employee " + (i + 1) + ": ");
            String name = scanner.nextLine();
            System.out.print("Enter ID of employee " + (i + 1) + ": ");
            String empId = scanner.nextLine();
            System.out.print("Enter salary of employee " + (i + 1) + ": ");
            double salary = scanner.nextDouble();
            scanner.nextLine(); // Consume newline
            employeeArray[i] = new Employee(name, empId, salary);
        }
```

```
            manager.create(employeeArray);
            System.out.println("Employees created successfully.");
            break;

        case 2:
            System.out.print("Enter name of new employee to insert: ");
            String newName = scanner.nextLine();
            System.out.print("Enter ID of new employee: ");
            String newEmpId = scanner.nextLine();
            System.out.print("Enter salary of new employee: ");
            double newSalary = scanner.nextDouble();
            scanner.nextLine(); // Consume newline

            Employee newEmployee = new Employee(newName, newEmpId,
newSalary);
            manager.insert(newEmployee);
            System.out.println("Employee " + newName + " inserted successfully.");
            break;

        case 3:
```

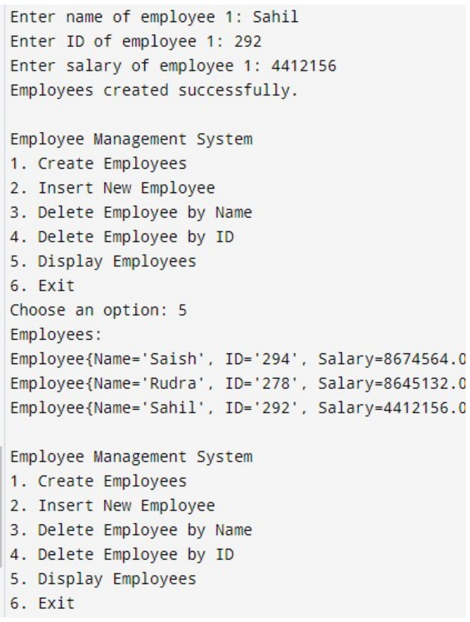
```
                    System.out.print("Enter name of employee to delete by name: ");
                    String deleteByName = scanner.nextLine();
                    manager.deleteByEName(deleteByName);
                    System.out.println("Employee " + deleteByName + " deleted
successfully.");
                    break;

                case 4:
                    System.out.print("Enter ID of employee to delete by ID: ");
                    String deleteById = scanner.nextLine();
                    manager.deleteByEId(deleteById);
                    System.out.println("Employee with ID " + deleteById + " deleted
successfully.");
                    break;

                case 5:
                    System.out.println("Employees:");
                    manager.displayEmployees();
                    break;

                case 6:
                    running = false;

                    System.out.println("Exiting the program. Goodbye!");

                    break;


                default:

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

            }

        }


        scanner.close();

    }

}
```

Output:

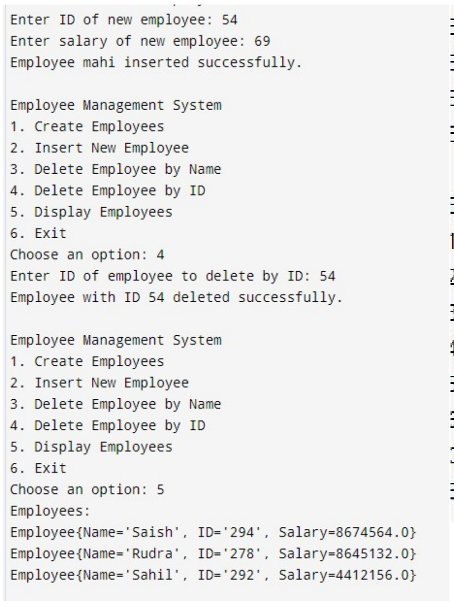
6.1



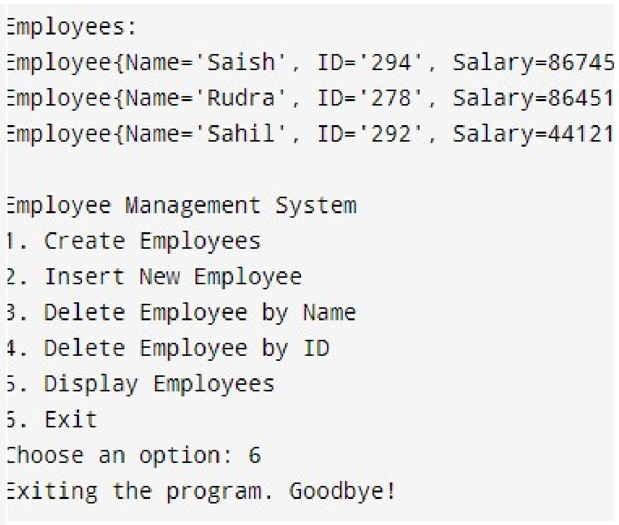
```
Employee Management System
1. Create Employees
2. Insert New Employee
3. Delete Employee by Name
4. Delete Employee by ID
5. Display Employees
6. Exit
Choose an option: 1
Enter the number of employees to create: 2
Enter name of employee 1: Saish
Enter ID of employee 1: 294
Enter salary of employee 1: 8674564
Enter name of employee 2: Rudra
Enter ID of employee 2: 278
Enter salary of employee 2: 8645132
Employees created successfully.

Employee Management System
1. Create Employees
2. Insert New Employee
3. Delete Employee by Name
4. Delete Employee by ID
5. Display Employees
6. Exit
Choose an option: 1
Enter the number of employees to create: 1
```



```
Enter name of employee 1: Sahil
Enter ID of employee 1: 292
Enter salary of employee 1: 4412156
Employees created successfully.

Employee Management System
1. Create Employees
2. Insert New Employee
3. Delete Employee by Name
4. Delete Employee by ID
5. Display Employees
6. Exit
Choose an option: 5
Employees:
Employee{Name='Saish', ID='294', Salary=8674564.0
Employee{Name='Rudra', ID='278', Salary=8645132.0
Employee{Name='Sahil', ID='292', Salary=4412156.0

Employee Management System
1. Create Employees
2. Insert New Employee
3. Delete Employee by Name
4. Delete Employee by ID
5. Display Employees
6. Exit
```

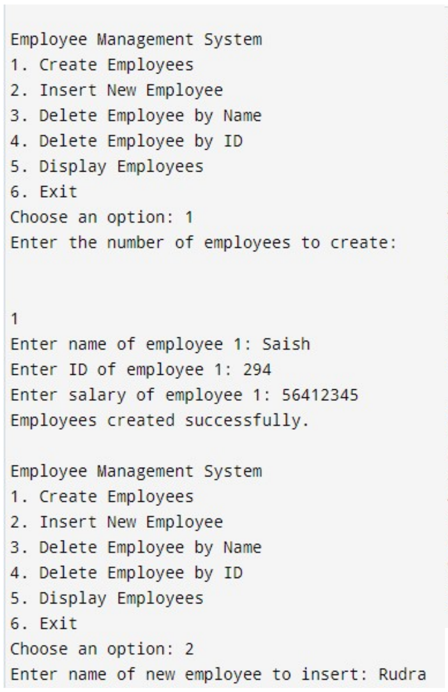


```
Enter ID of new employee: 54
Enter salary of new employee: 69
Employee mahi inserted successfully.

Employee Management System
1. Create Employees
2. Insert New Employee
3. Delete Employee by Name
4. Delete Employee by ID
5. Display Employees
6. Exit
Choose an option: 4
Enter ID of employee to delete by ID: 54
Employee with ID 54 deleted successfully.

Employee Management System
1. Create Employees
2. Insert New Employee
3. Delete Employee by Name
4. Delete Employee by ID
5. Display Employees
6. Exit
Choose an option: 5
Employees:
Employee{Name='Saish', ID='294', Salary=8674564.0}
Employee{Name='Rudra', ID='278', Salary=8645132.0}
Employee{Name='Sahil', ID='292', Salary=4412156.0}
```

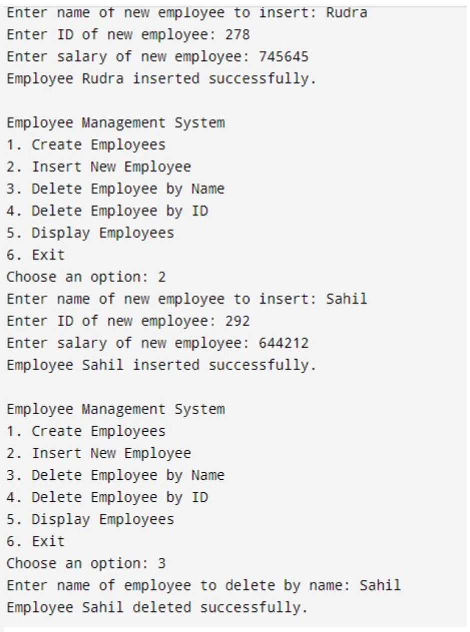


```
Employees:
Employee{Name='Saish', ID='294', Salary=86745
Employee{Name='Rudra', ID='278', Salary=86451
Employee{Name='Sahil', ID='292', Salary=44121

Employee Management System
1. Create Employees
2. Insert New Employee
3. Delete Employee by Name
4. Delete Employee by ID
5. Display Employees
6. Exit
Choose an option: 6
Exiting the program. Goodbye!
```

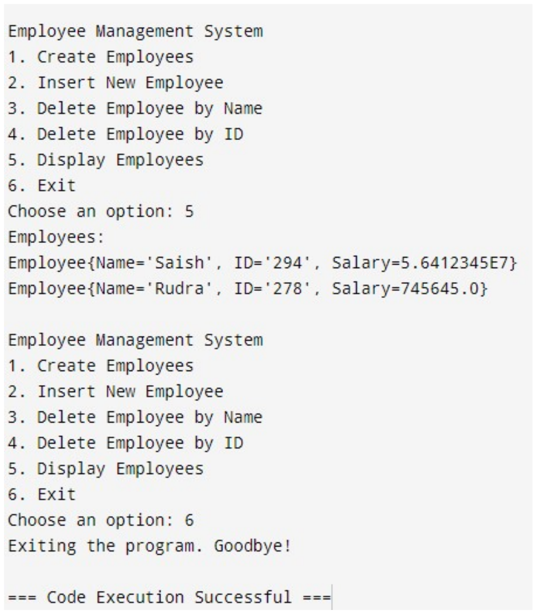
6.2



```
Employee Management System
1. Create Employees
2. Insert New Employee
3. Delete Employee by Name
4. Delete Employee by ID
5. Display Employees
6. Exit
Choose an option: 1
Enter the number of employees to create:


1
Enter name of employee 1: Saish
Enter ID of employee 1: 294
Enter salary of employee 1: 56412345
Employees created successfully.

Employee Management System
1. Create Employees
2. Insert New Employee
3. Delete Employee by Name
4. Delete Employee by ID
5. Display Employees
6. Exit
Choose an option: 2
Enter name of new employee to insert: Rudra
```



```
Enter name of new employee to insert: Rudra
Enter ID of new employee: 278
Enter salary of new employee: 745645
Employee Rudra inserted successfully.

Employee Management System
1. Create Employees
2. Insert New Employee
3. Delete Employee by Name
4. Delete Employee by ID
5. Display Employees
6. Exit
Choose an option: 2
Enter name of new employee to insert: Sahil
Enter ID of new employee: 292
Enter salary of new employee: 644212
Employee Sahil inserted successfully.

Employee Management System
1. Create Employees
2. Insert New Employee
3. Delete Employee by Name
4. Delete Employee by ID
5. Display Employees
6. Exit
Choose an option: 3
Enter name of employee to delete by name: Sahil
Employee Sahil deleted successfully.
```



```
Employee Management System
1. Create Employees
2. Insert New Employee
3. Delete Employee by Name
4. Delete Employee by ID
5. Display Employees
6. Exit
Choose an option: 5
Employees:
Employee{Name='Saish', ID='294', Salary=5.6412345E7}
Employee{Name='Rudra', ID='278', Salary=745645.0}

Employee Management System
1. Create Employees
2. Insert New Employee
3. Delete Employee by Name
4. Delete Employee by ID
5. Display Employees
6. Exit
Choose an option: 6
Exiting the program. Goodbye!

=== Code Execution Successful ===
```

Conclusion: Learned Implementation of Vectors and array List for the suitable codes

Date:_______ **Signature of faculty in-charge**

<u>Post Lab Descriptive Questions</u>

1) Write a note on the collection framework.

The Java Collection Framework (JCF) is a set of classes and interfaces in Java that provides data structures and algorithms for storing and manipulating groups of objects.

Interfaces:

Collection: Base interface for all collections.

List: Ordered collection allowing duplicates (e.g., ArrayList).

Set: Unordered collection that does not allow duplicates (e.g., HashSet).

Map: Collection of key-value pairs with unique keys (e.g., HashMap).

Classes: Various implementations of these interfaces, offering different performance characteristics.

Algorithms: Utility methods for sorting and searching (e.g., Collections.sort()).

Iterators: For traversing collections without exposing their underlying structure.

Benefits:

Reusability: Pre-built data structures and algorithms.

Flexibility: Supports various types of collections.

Efficiency: Optimized for performance.

2) Explain any 10 methods of Vector class in detail with the help of example

```
Vector<String> vector = new Vector<>();
vector.add("Apple");
vector.add("Banana");
System.out.println(vector); // Output: [Apple, Banana]

vector.add(1, "Orange");
System.out.println(vector); // Output: [Apple, Orange, Banana]

vector.remove("Apple");
System.out.println(vector); // Output: [Orange, Banana]

vector.remove(0);
System.out.println(vector); // Output: [Banana]

String fruit = vector.get(0);
System.out.println(fruit); // Output: Banana

int size = vector.size();
System.out.println(size); // Output: 1

boolean empty = vector.isEmpty();
System.out.println(empty); // Output: false

vector.clear();
System.out.println(vector); // Output: []

vector.add("Banana");
boolean hasBanana = vector.contains("Banana");
System.out.println(hasBanana); // Output: true

Object[] array = vector.toArray();
System.out.println(Arrays.toString(array)); // Output: [Banana]
```

3) What is an Arraylist? How does it differ from the array?

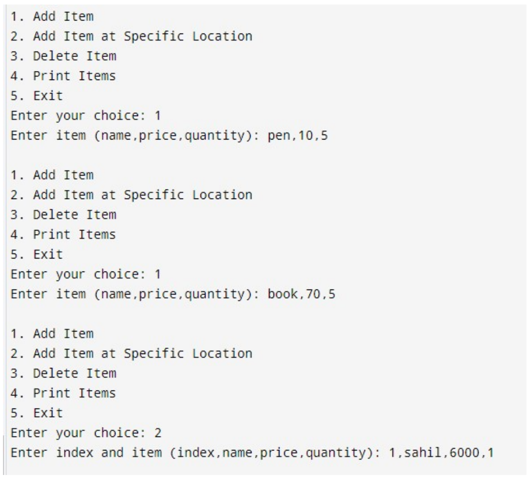
An ArrayList is a resizable array implementation of the List interface in Java, part of the Java Collection Framework. It allows dynamic resizing, meaning you can add or remove elements without needing to define the size at the time of creation.

Feature	Array	ArrayList
Size	Fixed size (static)	Dynamic size (grows/shrinks)
Type	Can hold primitive types	Only holds objects (reference types)
Memory	Allocated at compile-time	Allocated at runtime
Performance	Faster access	Slightly slower due to dynamic resizing
Methods	No built-in methods	Provides many useful methods (add, remove, etc.)

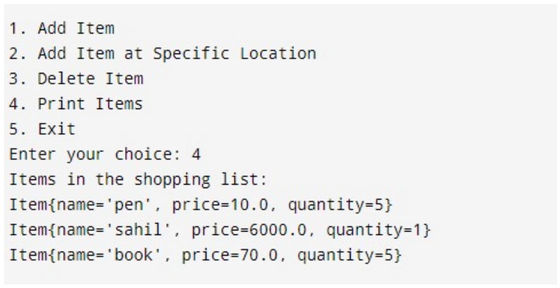
4) Implement a menu driven program for the following:
Accepts a shopping list (name, price and quantity)from the command line and stores them in a vector.
To delete specific item (given by user) in the vector
Add item at the end of the vector
Add item at specific location
Print the contents of vector using the enumeration interface.

```
import java.util.*;

class Item {
    private String name;
    private double price;
    private int quantity;

    public Item(String name, double price, int quantity) {
        this.name = name;
        this.price = price;
        this.quantity = quantity;
    }

    @Override
    public String toString() {
        return "Item{" + "name='" + name + '\'' + ", price=" + price + ", quantity=" +
quantity + '}';
```

```
    }
}

public class ShoppingList {
    private Vector<Item> items;

    public ShoppingList() {
        items = new Vector<>();
    }

    public void addItem(Item item) {
        items.add(item);
    }

    public void addItemAt(int index, Item item) {
        if (index >= 0 && index <= items.size()) {
            items.add(index, item);
        } else {
            System.out.println("Invalid index.");
        }
    }

    public void deleteItem(String name) {
        boolean removed = items.removeIf(item -> item.toString().contains(name));
        if (!removed) {
            System.out.println("Item not found.");
        }
    }

    public void printItems() {
        Enumeration<Item> enumeration = items.elements();
        while (enumeration.hasMoreElements()) {
            System.out.println(enumeration.nextElement());
        }
    }

    public static void main(String[] args) {
        ShoppingList list = new ShoppingList();
        Scanner sc = new Scanner(System.in);

        for (String arg : args) {
            String[] parts = arg.split(",");
            if (parts.length == 3) {
                String name = parts[0];
                double price = Double.parseDouble(parts[1]);
                int quantity = Integer.parseInt(parts[2]);
                list.addItem(new Item(name, price, quantity));
            }
        }

        while (true) {
            System.out.println("\n1. Add Item \n2. Add Item at Specific Location \n3. Delete
Item \n4. Print Items \n5. Exit");
            System.out.print("Enter your choice: ");
            int choice = sc.nextInt();
            sc.nextLine();

            switch (choice) {
                case 1:
                    System.out.print("Enter item (name,price,quantity): ");
                    String itemInput = sc.nextLine();
                    String[] itemParts = itemInput.split(",");
                    if (itemParts.length == 3) {
                        String name = itemParts[0];
                        double price = Double.parseDouble(itemParts[1]);
                        int quantity = Integer.parseInt(itemParts[2]);
                        list.addItem(new Item(name, price, quantity));
                    } else {
                        System.out.println("Invalid input.");
                    }
                    break;

                case 2:
                    System.out.print("Enter index and item (index,name,price,quantity): ");
                    String locInput = sc.nextLine();
                    String[] locParts = locInput.split(",");
                    if (locParts.length == 4) {
                        int index = Integer.parseInt(locParts[0]);
                        String name = locParts[1];
                        double price = Double.parseDouble(locParts[2]);
                        int quantity = Integer.parseInt(locParts[3]);
                        list.addItemAt(index, new Item(name, price, quantity));
                    } else {
                        System.out.println("Invalid input.");
                    }
                    break;
                case 3:
                    System.out.print("Enter item name to delete: ");
                    String nameToDelete = sc.nextLine();
                    list.deleteItem(nameToDelete);
                    break;

                case 4:
                    System.out.println("Items in the shopping list:");
                    list.printItems();
                    break;

                case 5:
                    System.exit(0);
                    break;

                default:
                    System.out.println("Invalid choice. Please try again.");
            }
        }
    }
}
```



```
1. Add Item
2. Add Item at Specific Location
3. Delete Item
4. Print Items
5. Exit
Enter your choice: 1
Enter item (name,price,quantity): pen,10,5

1. Add Item
2. Add Item at Specific Location
3. Delete Item
4. Print Items
5. Exit
Enter your choice: 1
Enter item (name,price,quantity): book,70,5

1. Add Item
2. Add Item at Specific Location
3. Delete Item
4. Print Items
5. Exit
Enter your choice: 2
Enter index and item (index,name,price,quantity): 1,sahil,6000,1
```



```
1. Add Item
2. Add Item at Specific Location
3. Delete Item
4. Print Items
5. Exit
Enter your choice: 4
Items in the shopping list:
Item{name='pen', price=10.0, quantity=5}
Item{name='sahil', price=6000.0, quantity=1}
Item{name='book', price=70.0, quantity=5}
```