Batch: E2 Roll No.: 16010123325

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 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

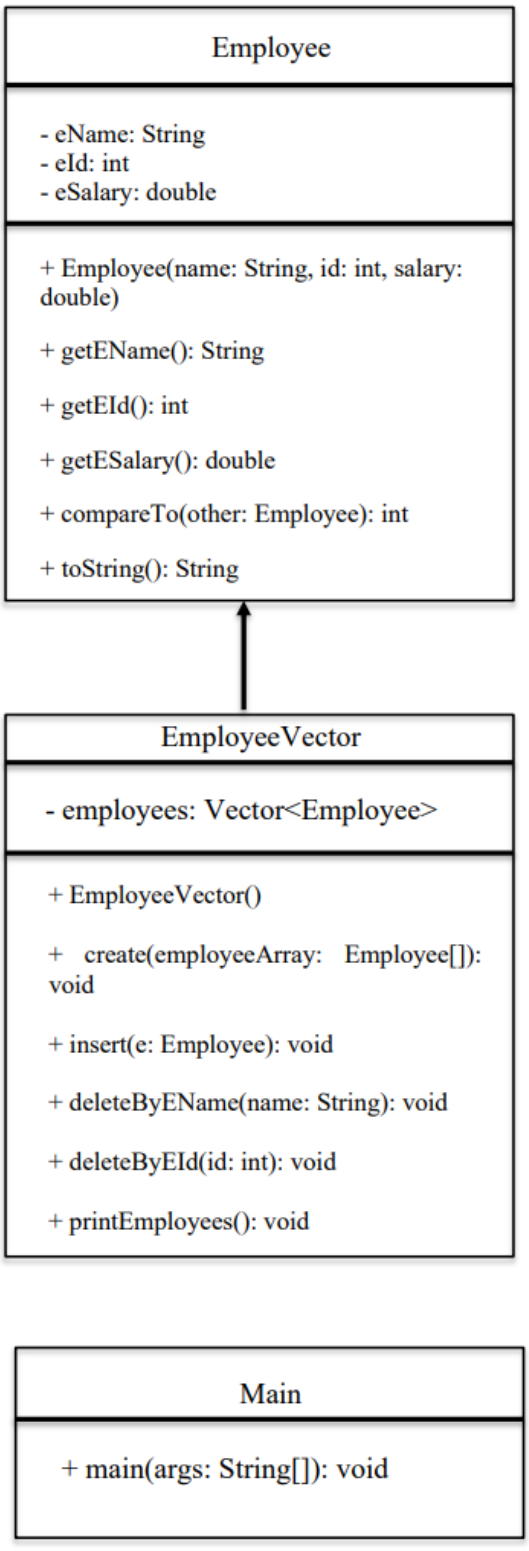
Memory allocation of vectors:

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. Initialize:

- Create an EmployeeVector instance to manage the list of employees.
- Create a Scanner object for user input.

2. Display Menu:

- Show options: Add Employees, Insert Employee, Display All Employees, Delete Employee by Name, Delete Employee by ID, Exit.

3. Process User Choice:

- Choice 1 (Add Employees):
  - Prompt for the number of employees.
  - For each employee, read details (name, ID, salary).
  - Create Employee objects and add them to the EmployeeVector.
- Choice 2 (Insert Employee):
  - Prompt for new employee details (name, ID, salary).
  - Create an Employee object and insert it into the EmployeeVector in sorted order by salary
- Choice 3 (Display All Employees):
  - Print all employees in the EmployeeVector sorted by salary
- Choice 4 (Delete Employee by Name):
  - Prompt for the employee's name to delete
  - Remove the employee with the matching name from the EmployeeVector
- Choice 5 (Delete Employee by ID):
  - Prompt for the employee's ID to delete.
  - Remove the employee with the matching ID from the EmployeeVector.
- Choice 6 (Exit):

• Exit the loop and end the program.

Implementation details:

```
import java.util.Scanner;
import java.util.Vector;

class Employee implements Comparable<Employee> {
    private String eName;
    private int eId;
    private double eSalary;

    public Employee(String eName, int eId, double eSalary) {
        this.eName = eName;
        this.eId = eId;
        this.eSalary = eSalary;
    }

    public String getEName() {
        return eName;
    }

    public int getEId() {
        return eId;
    }

    public double getESalary() {
        return eSalary;
    }

    @Override
    public int compareTo(Employee other) {
        return Double.compare(this.eSalary, other.eSalary);
    }

    @Override
    public String toString() {
        return "Employee Name: " + eName + ", ID: " + eId + ", Salary: "
+ eSalary;
    }
}
```

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

    public EmployeeVector() {
        employees = new Vector<>();
    }

    public void create(Employee[] employeeArray) {
        for (Employee e : employeeArray) {
            insert(e);
        }
    }

    public void insert(Employee e) {
        int index = 0;
        while (index < employees.size() &&
employees.get(index).compareTo(e) < 0) {
            index++;
        }
        employees.add(index, e);
    }

    public void deleteByEName(String name) {
        boolean removed = false;
        for (int i = 0; i < employees.size(); i++) {
            if (employees.get(i).getEName().equals(name)) {
                employees.remove(i);
                removed = true;
                System.out.println("Employee with name " + name + "
deleted.");
                break;
            }
        }
        if (!removed) {
            System.out.println("Employee with name " + name + " not
found.");
        }
    }

    public void deleteByEId(int id) {
        boolean removed = false;
        for (int i = 0; i < employees.size(); i++) {
            if (employees.get(i).getEId() == id) {
```

```
                employees.remove(i);
                removed = true;
                System.out.println("Employee with ID " + id + "
deleted.");
                break;
            }
        }
        if (!removed) {
            System.out.println("Employee with ID " + id + " not found.");
        }
    }

    public void printEmployees() {
        for (Employee e : employees) {
            System.out.println(e);
        }
    }
}

public class Main {
    public static void main(String[] args) {
        Scanner scanner = new Scanner(System.in);
        EmployeeVector employeeVector = new EmployeeVector();
        int choice;

        do {
            System.out.println("\nMenu:");
            System.out.println("1. Add Employees");
            System.out.println("2. Insert Employee");
            System.out.println("3. Display All Employees");
            System.out.println("4. Delete Employee by Name");
            System.out.println("5. Delete Employee by ID");
            System.out.println("6. Exit");
            System.out.print("Enter your choice: ");
            choice = scanner.nextInt();

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

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

```
                        System.out.println("Enter details for Employee "
+ (i + 1) + ":");
                        System.out.print("Name: ");
                        String name = scanner.next();
                        System.out.print("ID: ");
                        int id = scanner.nextInt();
                        System.out.print("Salary: ");
                        double salary = scanner.nextDouble();

                        employeeArray[i] = new Employee(name, id,
salary);
                    }

                    employeeVector.create(employeeArray);
                    System.out.println("Employees added and sorted by
salary.");
                    break;

                case 2:
                    System.out.println("Enter details for the new
Employee:");
                    System.out.print("Name: ");
                    String name = scanner.next();
                    System.out.print("ID: ");
                    int id = scanner.nextInt();
                    System.out.print("Salary: ");
                    double salary = scanner.nextDouble();

                    Employee newEmployee = new Employee(name, id,
salary);
                    employeeVector.insert(newEmployee);
                    System.out.println("Employee inserted in sorted
order.");
                    break;

                case 3:
                    System.out.println("Employees sorted by salary:");
                    employeeVector.printEmployees();
                    break;

                case 4:
                    System.out.print("Enter the name of the employee to
delete: ");
                    String nameToDelete = scanner.next();
```

```
                    employeeVector.deleteByEName(nameToDelete);
                    break;

                case 5:
                    System.out.print("Enter the ID of the employee to
delete: ");
                    int idToDelete = scanner.nextInt();
                    employeeVector.deleteByEId(idToDelete);
                    break;

                case 6:
                    System.out.println("Exiting...");
                    break;

                default:
                    System.out.println("Invalid choice. Please try
again.");
                    break;
            }
        } while (choice != 6);

        scanner.close();
    }
}
```

Output:

```
Menu:
1. Add Employees
2. Insert Employee
3. Display All Employees
4. Delete Employee by Name
5. Delete Employee by ID
6. Exit
Enter your choice: 1
Enter the number of employees: 2
Enter details for Employee 1:
Name: Shrey
ID: 1
Salary: 250000
Enter details for Employee 2:
Name: Mike
ID: 2
Salary: 150000
Employees added and sorted by salary.
```

```
Menu:
1. Add Employees
2. Insert Employee
3. Display All Employees
4. Delete Employee by Name
5. Delete Employee by ID
6. Exit
Enter your choice: 3
Employees sorted by salary:
Employee Name: Mike, ID: 2, Salary: 150000.0
Employee Name: Shrey, ID: 1, Salary: 250000.0
```

```
1. Add Employees
2. Insert Employee
3. Display All Employees
4. Delete Employee by Name
5. Delete Employee by ID
6. Exit
Enter your choice: 2
Enter details for the new Employee:
Name: Dan
ID: 3
Salary: 100000
Employee inserted in sorted order.

Menu:
1. Add Employees
2. Insert Employee
3. Display All Employees
4. Delete Employee by Name
5. Delete Employee by ID
6. Exit
Enter your choice: 3
Employees sorted by salary:
Employee Name: Dan, ID: 3, Salary: 100000.0
Employee Name: Mike, ID: 2, Salary: 150000.0
Employee Name: Shrey, ID: 1, Salary: 250000.0
```

```
Menu:
1. Add Employees
2. Insert Employee
3. Display All Employees
4. Delete Employee by Name
5. Delete Employee by ID
6. Exit
Enter your choice: 4
Enter the name of the employee to delete: Mike
Employee with name Mike deleted.

Menu:
1. Add Employees
2. Insert Employee
3. Display All Employees
4. Delete Employee by Name
5. Delete Employee by ID
6. Exit
Enter your choice: 3
Employees sorted by salary:
Employee Name: Dan, ID: 3, Salary: 100000.0
Employee Name: Shrey, ID: 1, Salary: 250000.0
```

```
Menu:
1. Add Employees
2. Insert Employee
3. Display All Employees
4. Delete Employee by Name
5. Delete Employee by ID
6. Exit
Enter your choice: 5
Enter the ID of the employee to delete: 3
Employee with ID 3 deleted.

Menu:
1. Add Employees
2. Insert Employee
3. Display All Employees
4. Delete Employee by Name
5. Delete Employee by ID
6. Exit
Enter your choice: 3
Employees sorted by salary:
Employee Name: Shrey, ID: 1, Salary: 250000.0
```

```
Menu:
1. Add Employees
2. Insert Employee
3. Display All Employees
4. Delete Employee by Name
5. Delete Employee by ID
6. Exit
Enter your choice: 6
Exiting...

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

Conclusion:

The provided Java program successfully implements an Employee class and an EmployeeVector class to manage employee records, sorted by salary, with functions to create, insert, delete by name, and delete by ID, utilizing a Vector to maintain the employee array.

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

Post Lab Descriptive Questions

1) Write a note on the collection framework.

The Java Collection Framework is a set of classes and interfaces that provide a way to work with collections of objects. It provides interfaces (Collection, List, Set, Map, Queue) and classes (ArrayList, LinkedList, HashSet, TreeSet, HashMap) that implement these interfaces. It offers polymorphism, generics, code reusability, flexibility, and performance.

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

The Vector class in Java is a legacy class that implements a dynamic array, similar to the ArrayList class. Here are 10 methods of the Vector class, explained in detail with examples:

1. addElement(E obj)

Adds an element to the end of the vector. This method is used to add elements to the vector.

2. add(E obj)

Adds an element to the end of the vector. This method is similar to addElement(E obj) and is used to add elements to the vector.

3. clear()

Removes all elements from the vector. This method is used to clear the vector of all its elements.

4. clone()

Returns a clone of the vector. This method is used to create a copy of the vector.

5. contains(Object obj)

Checks if the vector contains the specified element. This method is used to search for an element in the vector.

6. copyInto(Object[] anArray)

Copies the elements of the vector into an array. This method is used to copy the elements of the vector into an array.

7. elementAt(int index)

Returns the element at the specified position in the vector. This method is used to retrieve an element from the vector by its index.

8. elements()

Returns an enumeration of the elements in the vector. This method is used to iterate over the elements of the vector.

9. insertElementAt(E obj, int index)

Inserts an element at the specified position in the vector. This method is used to insert an element at a specific position in the vector.

10. removeElementAt(int index)

Removes the element at the specified position in the vector. This method is used to remove an element from the vector by its index.

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. It's a dynamic array that can grow or shrink in size as elements are added or removed.

The main differences between an ArrayList and an array are:

- Size: An array has a fixed size, whereas an ArrayList can dynamically resize itself.
- Resizability: An array cannot be resized once created, whereas an ArrayList can grow or shrink as needed.
- Type Safety: An array is type-safe, meaning it can only hold elements of the same type, whereas an ArrayList can hold elements of any type (although it's recommended to use generics for type safety).

In brief, an ArrayList is a more flexible and dynamic data structure compared to a traditional array.

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 enumeration interface.

Code:

```
import java.util.Scanner;
import java.util.Vector;
import java.util.Enumeration;

class Item
{
    String name;
    double price;
```

```
    int quantity;

    Item(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<Item> shoppingList = new Vector<>();
        Scanner sc = new Scanner(System.in);
        int choice;

        do
        {
            System.out.println("\n--- Shopping List Menu ---");
            System.out.println("1. Add item");
            System.out.println("2. Delete item");
            System.out.println("3. Add item at specific position");
            System.out.println("4. Display items");
            System.out.println("5. Exit");
            System.out.print("Enter your choice: ");
            choice = sc.nextInt();
            sc.nextLine();

            switch (choice)
            {
                case 1:

                    System.out.println("Enter item name:");
                    String name = sc.nextLine();
                    System.out.println("Enter price:");
                    double price = sc.nextDouble();
```

```
                    System.out.println("Enter quantity:");
                    int quantity = sc.nextInt();
                    shoppingList.add(new Item(name, price, quantity));
                    System.out.println("Item added to the list.");
                    break;

                case 2:
                    System.out.println("Enter the name of the item to
delete:");
                    String deleteItem = sc.nextLine();
                    deleteItemFromList(shoppingList, deleteItem);
                    break;

                case 3:
                    System.out.println("Enter position to add an item (0-
based index):");
                    int position = sc.nextInt();
                    sc.nextLine();
                    System.out.println("Enter item name:");
                    String posItemName = sc.nextLine();
                    System.out.println("Enter price:");
                    double posItemPrice = sc.nextDouble();
                    System.out.println("Enter quantity:");
                    int posItemQuantity = sc.nextInt();
                    shoppingList.add(position, new Item(posItemName,
posItemPrice, posItemQuantity));
                    System.out.println("Item added at position " +
position);
                    break;

                case 4:
                    System.out.println("Your shopping list:");
                    printList(shoppingList);
                    break;

                case 5:
                    System.out.println("Exiting..");
                    break;

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

```
        while (choice != 5);

        sc.close();
    }


    public static void deleteItemFromList(Vector<Item> list, String
itemName)
    {
        for (int i = 0; i < list.size(); i++)
        {
            if (list.get(i).name.equalsIgnoreCase(itemName))
            {
                list.remove(i);
                System.out.println(itemName + " removed from the shopping
list.");
                return;
            }
        }
        System.out.println(itemName + " not found in the shopping
list.");
    }

    public static void printList(Vector<Item> list)
    {
        if (list.isEmpty())
        {
            System.out.println("The shopping list is empty.");
        }
        else
        {
            Enumeration<Item> items = list.elements();
            while (items.hasMoreElements())
            {
                System.out.println(items.nextElement());
            }
        }
    }
}
```

```
java -cp /tmp/f2fDD8DhiQ/ShoppingList

--- Shopping List Menu ---
1. Add item
2. Delete item
3. Add item at specific position
4. Display items
5. Exit
Enter your choice: 1
Enter item name:
Strepcils
Enter price:
85
Enter quantity:
2
Item added to the list.
```

```
--- Shopping List Menu ---
1. Add item
2. Delete item
3. Add item at specific position
4. Display items
5. Exit
Enter your choice: 1
Enter item name:
Dolo 650
Enter price:
30
Enter quantity:
5
Item added to the list.
```

```
--- Shopping List Menu ---
1. Add item
2. Delete item
3. Add item at specific position
4. Display items
5. Exit4
Enter your choice: 4

4
Your shopping list:
Item: Strepcils, Price: 85.0, Quantity: 2
Item: Dolo 650, Price: 30.0, Quantity: 5
```

```
--- Shopping List Menu ---
1. Add item
2. Delete item
3. Add item at specific position
4. Display items
5. Exit
Enter your choice: 1
Enter item name:
Crocin
Enter price:
20
Enter quantity:
8
Item added to the list.
```

```
--- Shopping List Menu ---
1. Add item
2. Delete item
3. Add item at specific position
4. Display items
5. Exit
Enter your choice: 2
Enter the name of the item to delete:
Dolo 650
Dolo 650 removed from the shopping list.
```

```
--- Shopping List Menu ---
1. Add item
2. Delete item
3. Add item at specific position
4. Display items
5. Exit
Enter your choice: 3
Enter position to add an item (0-based index):
1
Enter item name:
Saridon
Enter price:
20
Enter quantity:
7
Item added at position 1
```

```
--- Shopping List Menu ---
1. Add item
2. Delete item
3. Add item at specific position
4. Display items
5. Exit
Enter your choice: 4
Your shopping list:
Item: Strepcils, Price: 85.0, Quantity: 2
Item: Saridon, Price: 20.0, Quantity: 7
Item: Crocin, Price: 20.0, Quantity: 8

--- Shopping List Menu ---
1. Add item
2. Delete item
3. Add item at specific position
4. Display items
5. Exit
Enter your choice: 5
Exiting..

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