Batch: B4 **Roll No.: 16010122221**

Experiment / assignment / tutorial No.05

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

Signature of the Staff In-charge with date

TITLE :Vector

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 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 the 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 Programming 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 the 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:

Class name	Exp5
Variables	-
Functions	main(), addAccount(Vector<Employee> arr, int n), deleteAccount(Vector<Employee> arr), displayAccount(Vector<Employee> arr)

Class name	Employee
Variables	Int id, String name, double salary
Functions	-

Algorithm:

1. Start
2. Create a Vector object Emp of the type Employee(E_Name, E_Id, E_Salary)
3. while(true)
4. 1.Create 2.Insert by salary 3.Delete by name 4.Delete by Id 5.Display 6. Exit
5. read choice
6. (Switch Case), Case 1(choice=1)
6.1 read number of employee records to be added(n)
6.2 for i=0, i<n, accept the employee details(e_name. e_id, e_salary)
6.3 Sort the employee records using comparator interface
6.3.1 if a.E_Salary>b.E_Salary, return 1
6.3.2 else if a.E_Salary<b.E_Salary, return -1
6.3.3 else, return 0
7. Case 2(choice=2)
7.1 accept the employee details
7.2 for i=0, i<Emp.size()
If Emp.get(i).E_Salary>e_salary, add the record to the i
th index.
8. Case 3(choice=3)
8.1 Declare a=0
8.2 read the name of record to be deleted 8.3 for i=0, i<Emp.size()
if e_name = Emp.get(i).E_Name, remove the corresponding record and increment a
8.4 if a=0, print “Employee name not found.”
9. Case 4(choice=4)
9.1 Declare b=0
9.2 read the name of record to be deleted 9.3 for i=0, i<Emp.size()
if e_id = Emp.get(i).E_Id, remove the corresponding record and increment b
9.4 if b=0, print “Employee id not found.”
10. Case 5(choice=5)
Print all the records
11. Case 6(choice=6)
12. Exit

Implementation details:

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

class Employee {
    private String name;
    private int id;
    private double salary;

    public Employee(String name, int id, double salary) {
        this.name = name;
        this.id = id;
        this.salary = salary;
    }

    public String getName() {
        return name;
    }

    public int getId() {
        return id;
    }

    public double getSalary() {
        return salary;
    }

    @Override
    public String toString() {
        return "Employee [ID: " + id + ", Name: " + name + ", Salary: " +
salary + "]";
    }
}
```

```
class EmployeeDatabase {
    private Vector<Employee> employees = new Vector<>();

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

    public void insert(Employee newEmployee) {
        int index = 0;
        for (; index < employees.size(); index++) {
            if (newEmployee.getSalary() <
employees.get(index).getSalary()) {
                break;
            }
        }
        employees.add(index, newEmployee);
    }

    public void deleteByName(String name) {
        for (int i = 0; i < employees.size(); i++) {
            if (employees.get(i).getName().equals(name)) {
                employees.remove(i);
                break;
            }
        }
    }

    public void deleteById(int id) {
        for (int i = 0; i < employees.size(); i++) {
            if (employees.get(i).getId() == id) {
                employees.remove(i);
                break;
            }
        }
    }

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

```
}

public class Main {
    public static void main(String[] args) {
        Scanner scanner = new Scanner(System.in);
        EmployeeDatabase database = new EmployeeDatabase();

        System.out.print("Enter the number of employees: ");
        int numEmployees = scanner.nextInt();
        scanner.nextLine(); // Consume the newline character

        Employee[] records = new Employee[numEmployees];
        for (int i = 0; i < numEmployees; i++) {
            System.out.println("\nEnter details for Employee #" + (i +
1));
            System.out.print("Name: ");
            String name = scanner.nextLine();
            System.out.print("ID: ");
            int id = scanner.nextInt();
            System.out.print("Salary: ");
            double salary = scanner.nextDouble();
            scanner.nextLine(); // Consume the newline character

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

        database.create(records);

        System.out.println("\nEmployees before deletion:");
        database.displayEmployees();

        System.out.print("\nEnter the name to delete: ");
        String nameToDelete = scanner.nextLine();
        database.deleteByName(nameToDelete);

        System.out.print("Enter the ID to delete: ");
        int idToDelete = scanner.nextInt();
        database.deleteById(idToDelete);

        System.out.println("\nEmployees after deletion:");
        database.displayEmployees();

        scanner.close();
    }
```

```
}
```

Output:

```
Enter the number of employees: 2
Enter details for Employee #1
Name: Rahul
ID: 1
Salary: 20000
Enter details for Employee #2
Name: Parv
ID: 2
Salary: 150000
Employees before deletion:
Employee [ID: 1, Name: Rahul, Salary: 20000.0]
Employee [ID: 2, Name: Parv, Salary: 150000.0]

Enter the name to delete: Rahul
Enter the ID to delete: 1
Employees after deletion:
Employee [ID: 2, Name: Parv, Salary: 150000.0]
```

Conclusion:

The concept of vectors in Java was studied and implemented.

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

Post Lab Descriptive Questions

1) **What is the output of the following Program**

```
import java.util.*;
class demo2 {
   public static void main(String[] args)
   {
      Vector v = new Vector(20);
      v.addElement("Geeksforgeeks");
      v.insertElementAt("Java", 2);
      System.out.println(v.firstElement());
   }
}
```

Output: It gives no output because of Array index Out Of Bound Exception because initially there was only one element at index 0. Next it expects a value to
be added at index 1, but we are adding at index 2. Hence an exception is raised.

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

1. void add(int index, Object element)
Inserts the specified element at the specified position in this Vector.

Example: v.add(1, “apple”);

2. boolean add(Object o)

Appends the specified element to the end of this Vector.

Example: v.add(“pear”);

3. boolean addAll(int index, Collection c)

Inserts all of the elements in the specified Collection into this Vector at the specified position.

Example: v.addAll(3, arr);

4. void addElement(Object obj)

Adds the specified component to the end of this vector, increasing its size by one.

Example: v.addElement(“plum”);

5. int capacity()

Returns the current capacity of this vector.

Example: v.capacity();

6. boolean contains(Object elem)

Tests if the specified object is a component in this vector.

Example: v.contains(“apple”);

7. void copyInto(Object[] anArray)

Copies the components of this vector into the specified array.

Example: v.copyInto(arr);

8. Object elementAt(int index)

Returns the component at the specified index.

Example: v.elementAt(2);

9. Object remove(int index)

Removes the element at the specified position in this vector.

Example: v. remove(3);

10.int size()

Returns the number of components in this vector.

Example: v.size();