Batch: A1	**Roll No.: 16010123012**

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

2. Display Menu:

Create Employees, Insert Employee, Display All Employees, Delete Employee by Name, Delete Employee by ID, Exit.

3. Process User Choice:

1. Create Employees
   Prompt for the number of employees.
   For each employee, read details (name, ID, salary).
   Create Employee objects and add them to the EmployeeVector.
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

3. Display All Employees
   Print all employees in the EmployeeVector sorted by salary
4. Delete Employee by Name
   Prompt for the employee's name to delete.
   Remove the employee with the matching name from the EmployeeVector
5. Delete Employee by ID
   Prompt for the employee's ID to delete.
   Remove the employee with the matching ID from the EmployeeVector.
6. Exit
   Exit the loop and end the program.

Implementation details:

Vector

```
import java.util.ArrayList;
import java.util.List;
import java.util.Scanner;
class Employee {
    private String name;
    private int id;
    private int salary;

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

    public String getName() {
        return name;
    }

    public int getId() {
        return id;
    }

    public int getSalary() {
        return salary;
    }

    @Override
    public String toString() {
        return "Employee name: " + name + ", Employee id: " + id + ",
Employee salary: " + salary;
    }
}

class EmployeeManager {
    private List<Employee> employeeList = new ArrayList<>();
    private Scanner sc = new Scanner(System.in);

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

    public void create() {
        System.out.print("How many employees do you want to enter: ");
        int n = sc.nextInt();
        sc.nextLine();
        for (int i = 0; i < n; i++) {
            System.out.print("Enter Employee name: ");
            String name = sc.nextLine();
            System.out.print("Enter Employee id: ");
            int id = sc.nextInt();
            System.out.print("Enter Employee salary: ");
            int salary = sc.nextInt();
            sc.nextLine();
            Employee newEmp = new Employee(name, id, salary);
            add(newEmp);
        }
    }

    public void insert() {
        System.out.print("Enter Employee name: ");
        String name = sc.nextLine();
        System.out.print("Enter Employee id: ");
        int id = sc.nextInt();
        System.out.print("Enter Employee salary: ");
        int salary = sc.nextInt();
        sc.nextLine();
```

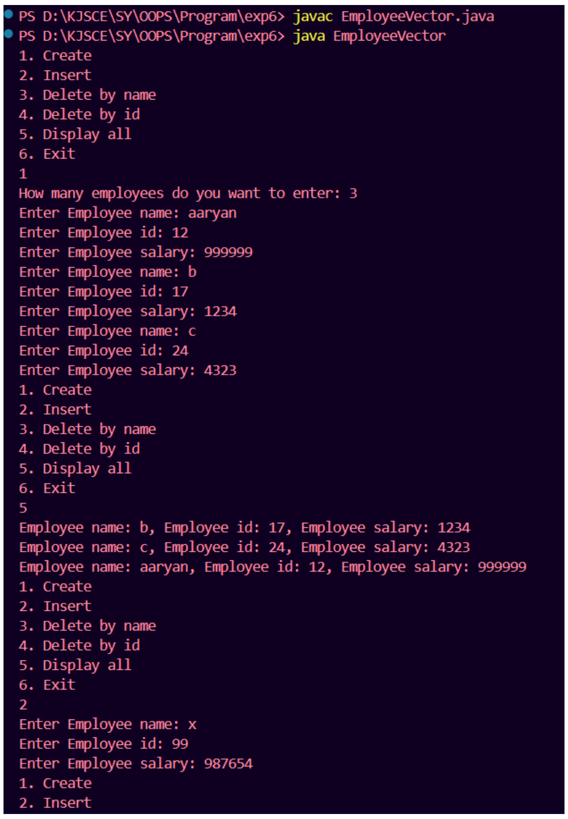
```java
        Employee newEmployee = new Employee(name, id, salary);
        add(newEmployee);
    }

    public void deleteByName() {
        System.out.print("Enter the employee name to delete: ");
        String nameToDelete = sc.nextLine();
        boolean removed = employeeList.removeIf(emp ->
emp.getName().equalsIgnoreCase(nameToDelete));
        if (removed) {
            System.out.println("Employee " + nameToDelete + " deleted.");
        } else {
            System.out.println("Employee not found.");
        }
    }

    public void deleteById() {
        System.out.print("Enter the employee id to delete: ");
        int idToDelete = sc.nextInt();
        sc.nextLine();
        boolean removed = employeeList.removeIf(emp -> emp.getId() ==
idToDelete);
        if (removed) {
            System.out.println("Employee with id " + idToDelete + "
deleted.");
        } else {
            System.out.println("Employee not found.");
        }
    }

    private void add(Employee e) {
        int i = 0;
        while (i < employeeList.size() && employeeList.get(i).getSalary()
< e.getSalary()) {
            i++;
        }
        employeeList.add(i, e);
    }
}

public class EmployeeVector {
    public static void main(String[] args) {
        Scanner sc = new Scanner(System.in);
        EmployeeManager manager = new EmployeeManager();
        int choice;
        do {
            System.out.println("1. Create");
            System.out.println("2. Insert");
            System.out.println("3. Delete by name");
            System.out.println("4. Delete by id");
            System.out.println("5. Display all");
            System.out.println("6. Exit");
            choice = sc.nextInt();
            sc.nextLine();
            switch (choice) {
                case 1:
                    manager.create();
                    break;
                case 2:
                    manager.insert();
                    break;
                case 3:
                    manager.deleteByName();
                    break;
                case 4:
                    manager.deleteById();
                    break;
                case 5:
                    manager.display();
                    break;
                case 6:
                    System.out.println("Exited");
                    break;
                default:
                    System.out.println("Invalid choice. Try again.");
            }
        } while (choice != 6);
        sc.close();
    }
}
```

ArrayList

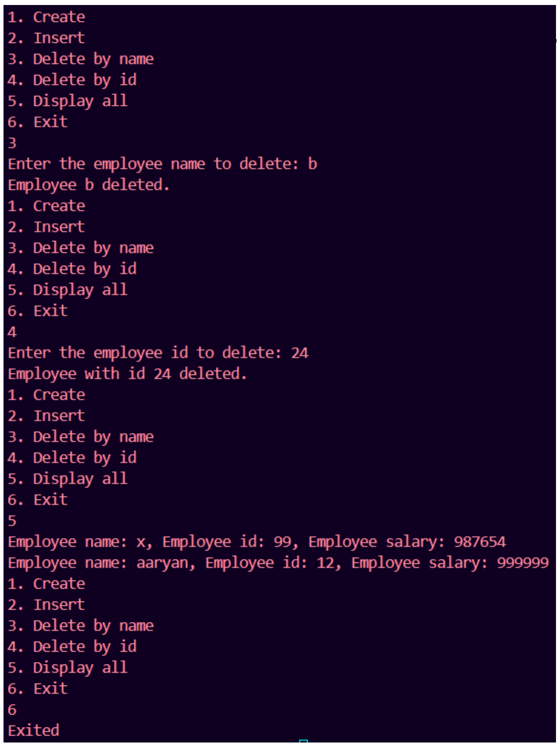
```
import java.util.ArrayList;
import java.util.Scanner;

class Employee {
    private String E_name;
    private int E_id;
    private int E_salary;
    private ArrayList<Employee> al = new ArrayList<>();
    private Scanner sc = new Scanner(System.in);

    Employee(String name, int id, int salary) {
        E_name = name;
        E_id = id;
        E_salary = salary;
    }

    public void display() {
        if (al.size() == 0) {
            System.out.println("No employees available to display.");
        } else {
            for (Employee emp : al) {
                System.out.println("Employee name: " + emp.E_name + ", Employee
id: " + emp.E_id + ", Employee salary: "
                        + emp.E_salary);
            }
        }
    }

    public void create() {
        System.out.print("How many employees you want to enter: ");
        int n = sc.nextInt();
        for (int i = 0; i < n; i++) {
            System.out.print("Enter Employee name: ");
            String name = sc.next();
            System.out.print("Enter Employee id: ");
            int id = sc.nextInt();
            System.out.print("Enter Employee salary: ");
            int salary = sc.nextInt();
            Employee newemp = new Employee(name, id, salary);
            add(newemp);
        }
    }

    public void insert() {
        System.out.print("Enter Employee name: ");
        String name = sc.next();
        System.out.print("Enter Employee id: ");
        int id = sc.nextInt();
        System.out.print("Enter Employee salary: ");
        int salary = sc.nextInt();
        Employee newEmployee = new Employee(name, id, salary);
        add(newEmployee);
    }

    public void delete_by_name() {
        System.out.print("Enter the employee name to delete: ");
        String nameToDelete = sc.next();
        for (int i = 0; i < al.size(); i++) {
            if (al.get(i).E_name.equals(nameToDelete)) {
                remove(i);
                System.out.println("Employee " + nameToDelete + " deleted.");
                return;
            }
        }
        System.out.println("Employee not found.");
    }

    public void delete_by_id() {
        System.out.print("Enter the employee id to delete: ");
        int idToDelete = sc.nextInt();
        for (int i = 0; i < al.size(); i++) {
            if (al.get(i).E_id == idToDelete) {
                remove(i);
                System.out.println("Employee with id " + idToDelete + "
deleted.");
                return;
            }
        }
        System.out.println("Employee not found.");
    }

    public boolean add(Employee e) {
        if (al.size() == 0) {
            al.add(e);
        } else {
            int i = 0;
            while (i < al.size() && al.get(i).E_salary < e.E_salary) {
                i++;
            }
            al.add(i, e);
        }
        return true;
    }

    public int lastIndexOf(int salary, int index) {
        for (int i = index; i >= 0; i--) {
            if (al.get(i).E_salary <= salary) {
                return i;
            }
        }
        return -1;
    }

    public void remove(int index) {
```

```java
        if (index >= 0 && index < al.size()) {
            al.remove(index);
        } else {
            System.out.println("Invalid index. Unable to remove employee.");
        }
    }
}

public class EmployeeArrayList {
    public static void main(String args[]) {
        Scanner sc = new Scanner(System.in);
        Employee emp = new Employee("", 0, 0);

        int choice;
        do {
            System.out.println("1. Create");
            System.out.println("2. Insert");
            System.out.println("3. Delete by name");
            System.out.println("4. Delete by id");
            System.out.println("5. Display all");
            System.out.println("6. Exit");
            choice = sc.nextInt();

            switch (choice) {
                case 1:
                    emp.create();
                    break;
                case 2:
                    emp.insert();
                    break;
                case 3:
                    emp.delete_by_name();
                    break;
                case 4:
                    emp.delete_by_id();
                    break;
                case 5:
                    emp.display();
                    break;
                case 6:
                    System.out.println("Exited");
                    break;
                default:
                    System.out.println("Invalid choice. Try again.");
            }
        } while (choice != 6);
    }
```

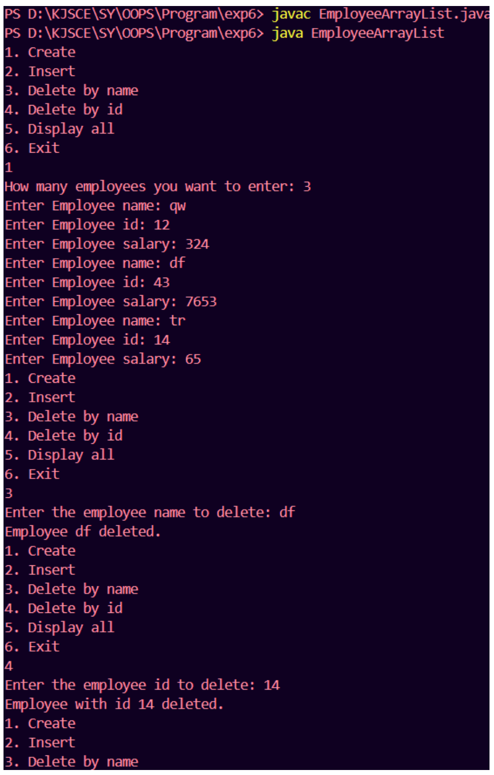
Output:



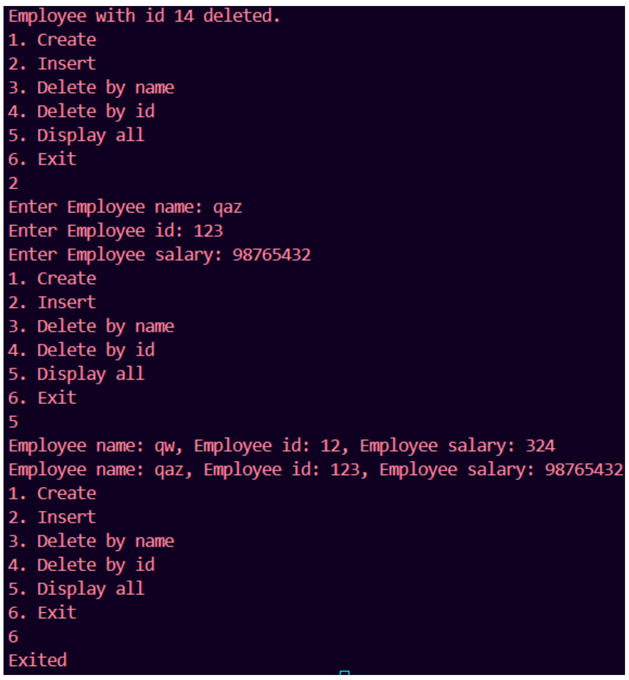
```
PS D:\KJSCE\SY\OOPS\Program\exp6> javac EmployeeVector.java
PS D:\KJSCE\SY\OOPS\Program\exp6> java EmployeeVector
1. Create
2. Insert
3. Delete by name
4. Delete by id
5. Display all
6. Exit
1
How many employees do you want to enter: 3
Enter Employee name: aaryan
Enter Employee id: 12
Enter Employee salary: 999999
Enter Employee name: b
Enter Employee id: 17
Enter Employee salary: 1234
Enter Employee name: c
Enter Employee id: 24
Enter Employee salary: 4323
1. Create
2. Insert
3. Delete by name
4. Delete by id
5. Display all
6. Exit
5
Employee name: b, Employee id: 17, Employee salary: 1234
Employee name: c, Employee id: 24, Employee salary: 4323
Employee name: aaryan, Employee id: 12, Employee salary: 999999
1. Create
2. Insert
3. Delete by name
4. Delete by id
5. Display all
6. Exit
2
Enter Employee name: x
Enter Employee id: 99
Enter Employee salary: 987654
1. Create
2. Insert
```



```
1. Create
2. Insert
3. Delete by name
4. Delete by id
5. Display all
6. Exit
3
Enter the employee name to delete: b
Employee b deleted.
1. Create
2. Insert
3. Delete by name
4. Delete by id
5. Display all
6. Exit
4
Enter the employee id to delete: 24
Employee with id 24 deleted.
1. Create
2. Insert
3. Delete by name
4. Delete by id
5. Display all
6. Exit
5
Employee name: x, Employee id: 99, Employee salary: 987654
Employee name: aaryan, Employee id: 12, Employee salary: 999999
1. Create
2. Insert
3. Delete by name
4. Delete by id
5. Display all
6. Exit
6
Exited
```



```
PS D:\KJSCE\SY\OOPS\Program\exp6> javac EmployeeArrayList.java
PS D:\KJSCE\SY\OOPS\Program\exp6> java EmployeeArrayList
1. Create
2. Insert
3. Delete by name
4. Delete by id
5. Display all
6. Exit
1
How many employees you want to enter: 3
Enter Employee name: qw
Enter Employee id: 12
Enter Employee salary: 324
Enter Employee name: df
Enter Employee id: 43
Enter Employee salary: 7653
Enter Employee name: tr
Enter Employee id: 14
Enter Employee salary: 65
1. Create
2. Insert
3. Delete by name
4. Delete by id
5. Display all
6. Exit
3
Enter the employee name to delete: df
Employee df deleted.
1. Create
2. Insert
3. Delete by name
4. Delete by id
5. Display all
6. Exit
4
Enter the employee id to delete: 14
Employee with id 14 deleted.
1. Create
2. Insert
3. Delete by name
```



```
Employee with id 14 deleted.
1. Create
2. Insert
3. Delete by name
4. Delete by id
5. Display all
6. Exit
2
Enter Employee name: qaz
Enter Employee id: 123
Enter Employee salary: 98765432
1. Create
2. Insert
3. Delete by name
4. Delete by id
5. Display all
6. Exit
5
Employee name: qw, Employee id: 12, Employee salary: 324
Employee name: qaz, Employee id: 123, Employee salary: 98765432
1. Create
2. Insert
3. Delete by name
4. Delete by id
5. Display all
6. Exit
6
Exited
```

Conclusion:

We covered Vector and ArrayList, explored the differences between them, and applied them in our code.

Date: 10 / 09 / 2024 **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 exampl

The Vector class in Java is a legacy class that implements a dynamic array, similar to the ArrayList class.

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.

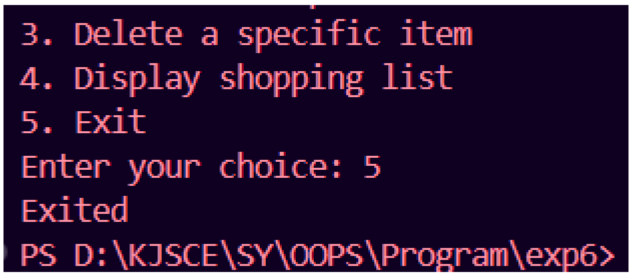
To delete specific item (given by user) in the vector

Add item at the end of the vector

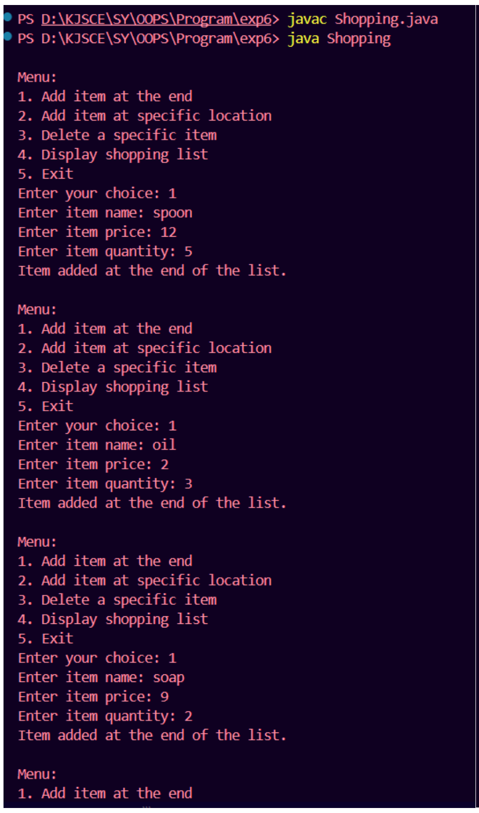
Add item at specific location

Print the contents of vector using the enumeration interface.

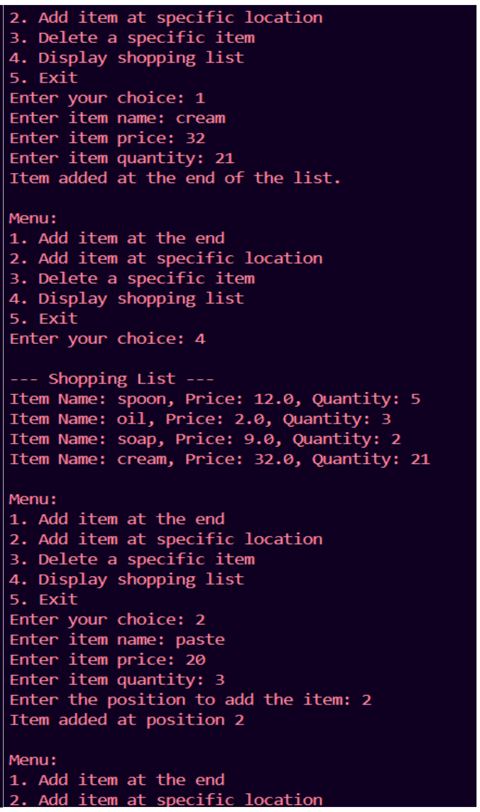
```java
import java.util.Scanner;
import java.util.Vector;

class ShopItem {
    private String name;
    private float price;
    private int quantity;

    public ShopItem() {
    }

    public ShopItem(String name, float price, int quantity) {
        this.name = name;
        this.price = price;
        this.quantity = quantity;
    }

    public String getName() {
        return name;
    }

    public float getPrice() {
        return price;
    }

    public int getQuantity() {
        return quantity;
    }

    @Override
    public String toString() {
        return "Item Name: " + name + ", Price: " + price + ", Quantity:
" + quantity;
    }
}

class ShoppingList {
    private Vector<ShopItem> shoppingList = new Vector<>();
    private Scanner sc = new Scanner(System.in);

    public void display() {
        if (shoppingList.isEmpty()) {
            System.out.println("The shopping list is empty.");
        } else {
            System.out.println("\n--- Shopping List ---");
            for (ShopItem item : shoppingList) {
                System.out.println(item);
            }
        }
    }

    public void addEnd() {
        ShopItem item = getItemDetailsFromUser();
        shoppingList.add(item);
        System.out.println("Item added at the end of the list.");
    }

    public void addSpecific() {
        ShopItem item = getItemDetailsFromUser();
        System.out.print("Enter the position to add the item: ");
        int position = sc.nextInt();
        sc.nextLine();
        if (position >= 0 && position <= shoppingList.size()) {
            shoppingList.add(position, item);
            System.out.println("Item added at position " + position);
        } else {
            System.out.println("Invalid position! Item not added.");
        }
    }

    public void removeItem() {
        System.out.print("Enter the name of the item to delete: ");
        String nameToDelete = sc.next();
        boolean removed = shoppingList.removeIf(item ->
item.getName().equalsIgnoreCase(nameToDelete));
        if (removed) {
            System.out.println("Item deleted successfully.");
        } else {
            System.out.println("Item not found in the list.");
        }
    }

    private ShopItem getItemDetailsFromUser() {
        System.out.print("Enter item name: ");
        String name = sc.next();
        System.out.print("Enter item price: ");
        float price = sc.nextFloat();
        System.out.print("Enter item quantity: ");
        int quantity = sc.nextInt();
        sc.nextLine();
        return new ShopItem(name, price, quantity);
    }
}

public class Shopping {
    public static void main(String[] args) {
        Scanner sc = new Scanner(System.in);
        ShoppingList shoppingList = new ShoppingList();
        int choice;
        do {
            System.out.println("\nMenu:");
            System.out.println("1. Add item at the end");
            System.out.println("2. Add item at specific location");
            System.out.println("3. Delete a specific item");
            System.out.println("4. Display shopping list");
            System.out.println("5. Exit");
            System.out.print("Enter your choice: ");
            choice = sc.nextInt();
            sc.nextLine();
            switch (choice) {
                case 1:
                    shoppingList.addEnd();
                    break;
                case 2:
                    shoppingList.addSpecific();
                    break;
                case 3:
                    shoppingList.removeItem();
                    break;
                case 4:
                    shoppingList.display();
                    break;
                case 5:
                    System.out.println("Exited");
                    break;
                default:
                    System.out.println("Invalid choice! Please try
again.");
            }
        } while (choice != 5);
        sc.close();
    }
}
```



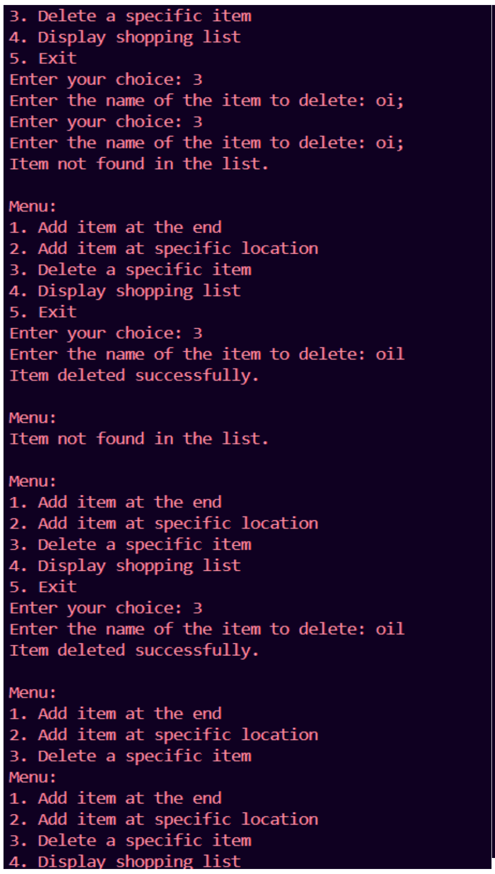
```
3. Delete a specific item
4. Display shopping list
5. Exit
Enter your choice: 5
Exited
PS D:\KJSCE\SY\OOPS\Program\exp6>
```



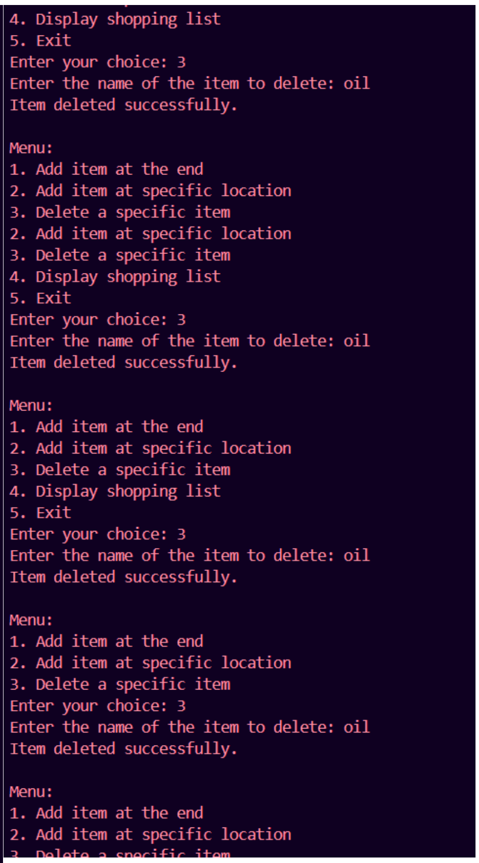
```
PS D:\KJSCE\SY\OOPS\Program\exp6> javac Shopping.java
PS D:\KJSCE\SY\OOPS\Program\exp6> java Shopping

Menu:
1. Add item at the end
2. Add item at specific location
3. Delete a specific item
4. Display shopping list
5. Exit
Enter your choice: 1
Enter item name: spoon
Enter item price: 12
Enter item quantity: 5
Item added at the end of the list.

Menu:
1. Add item at the end
2. Add item at specific location
3. Delete a specific item
4. Display shopping list
5. Exit
Enter your choice: 1
Enter item name: oil
Enter item price: 2
Enter item quantity: 3
Item added at the end of the list.

Menu:
1. Add item at the end
2. Add item at specific location
3. Delete a specific item
4. Display shopping list
5. Exit
Enter your choice: 1
Enter item name: soap
Enter item price: 9
Enter item quantity: 2
Item added at the end of the list.

Menu:
1. Add item at the end
```



```
2. Add item at specific location
3. Delete a specific item
4. Display shopping list
5. Exit
Enter your choice: 1
Enter item name: cream
Enter item price: 32
Enter item quantity: 21
Item added at the end of the list.

Menu:
1. Add item at the end
2. Add item at specific location
3. Delete a specific item
4. Display shopping list
5. Exit
Enter your choice: 4

--- Shopping List ---
Item Name: spoon, Price: 12.0, Quantity: 5
Item Name: oil, Price: 2.0, Quantity: 3
Item Name: soap, Price: 9.0, Quantity: 2
Item Name: cream, Price: 32.0, Quantity: 21

Menu:
1. Add item at the end
2. Add item at specific location
3. Delete a specific item
4. Display shopping list
5. Exit
Enter your choice: 2
Enter item name: paste
Enter item price: 20
Enter item quantity: 3
Enter the position to add the item: 2
Item added at position 2

Menu:
1. Add item at the end
2. Add item at specific location
```



```
3. Delete a specific item
4. Display shopping list
5. Exit
Enter your choice: 3
Enter the name of the item to delete: oi;
Enter your choice: 3
Enter the name of the item to delete: oi;
Item not found in the list.

Menu:
1. Add item at the end
2. Add item at specific location
3. Delete a specific item
4. Display shopping list
5. Exit
Enter your choice: 3
Enter the name of the item to delete: oil
Item deleted successfully.

Menu:
Item not found in the list.

Menu:
1. Add item at the end
2. Add item at specific location
3. Delete a specific item
4. Display shopping list
5. Exit
Enter your choice: 3
Enter the name of the item to delete: oil
Item deleted successfully.

Menu:
1. Add item at the end
2. Add item at specific location
3. Delete a specific item
Menu:
1. Add item at the end
2. Add item at specific location
3. Delete a specific item
4. Display shopping list
```



```
4. Display shopping list
5. Exit
Enter your choice: 3
Enter the name of the item to delete: oil
Item deleted successfully.

Menu:
1. Add item at the end
2. Add item at specific location
3. Delete a specific item
2. Add item at specific location
3. Delete a specific item
4. Display shopping list
5. Exit
Enter your choice: 3
Enter the name of the item to delete: oil
Item deleted successfully.

Menu:
1. Add item at the end
2. Add item at specific location
3. Delete a specific item
4. Display shopping list
5. Exit
Enter your choice: 3
Enter the name of the item to delete: oil
Item deleted successfully.

Menu:
1. Add item at the end
2. Add item at specific location
3. Delete a specific item
Enter your choice: 3
Enter the name of the item to delete: oil
Item deleted successfully.

Menu:
1. Add item at the end
2. Add item at specific location
```