Batch: B1 **Roll No.: 1711072**

Experiment / assignment / tutorial No. 07

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

Signature of the Staff In-charge with date

TITLE : Vector and Hash Map

AIM: Create a class **ShoppingList** which stores Name of the items, item id, cost, and total no. of items. Use class Vector to maintain an array of items in the descending order of the cost.

1. Accepts a shopping list from the command line and stores them in a vector.
2. To delete an specific item (given by user) in the vector
3. Add item at the end of the vector
4. Add item at specific location
5. Print the contents of vector.
6. Sort the contents of the vector in descending order.

Provide the following functions

1) create() : this function will accept the n items records in any order and will arrange them in the sorted order. This data will be accepted from the command line and stores them in a vector.

2) insert(): to insert the given item at appropriate index in the vector depending upon the grand total.

3) sort(): to sort the items in shopping list in descending order of ID.

4) deleteById( ): to accept the id of the item and delete the record having given roll no.

In an array 1-100 many numbers are duplicates. Use hash Map. Given two arrays 1,2,3,4,5 and 2,3,1,0,5. Find which element is not present in the second array. Use hash Table.

Expected OUTCOME of Experiment:

CO2: Solve problems using Java basic constructs (like if else statement, control structures, and data types, array, string, vectors, packages, collection class).

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:

Vector implements a dynamic array. It is similar to ArrayList, but with two differences:

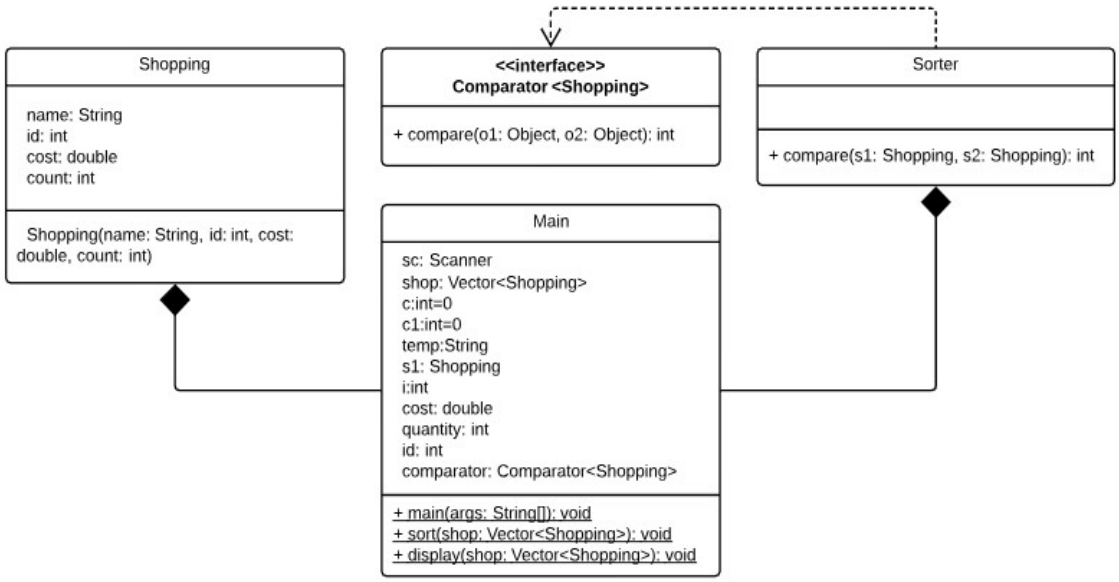
- Vector is synchronized.
- Vector contains many legacy methods that are not part of the collections framework.

Vector proves to be very useful if you don't know the size of the array in advance or you just need one that can change sizes over the lifetime of a program.

Comparators:

Java Comparator is an interface for sorting Java objects. Invoked by "java.util.comparator," Java Comparator compares two Java objects in a "compare(Object o1, Object o2)" format. Using configurable methods, Java Comparator can compare objects to return an integer based on a positive, equal or negative comparison. Since it is not limited to comparing numbers, this can allow Java Comparator to be set up to order lists alphabetically or numerically. With java.io.Serializable, Java comparator can also be used to successfully order serialized data structures. Java Comparator is similar to the Comparable interface but is intended for defining alternate sort orders where Comparable sorts by natural ordering such as lexicographic sorting.

Class Diagram:



Implementation details:

Main Class:

```
import java.util.Collections;
import java.util.Scanner;
import java.util.Vector;
import java.util.Comparator;
public class Main {

  public static void main(String[] args) {
    Scanner sc=new Scanner(System.in);
    Vector<Shopping> shop=new Vector<Shopping>();
    int c=0,c1=0;
    String temp;
    for(int i=0;i<Integer.parseInt(args[0]);i++){
      Shopping s1=new Shopping(args[c1+1],Integer.parseInt(args[c1+2]),
Double.parseDouble(args[c1+3]), Integer.parseInt(args[c1+4]));
      c1+=4;
      shop.add(s1);
    }
    while(c!=6)
    {
```

```
      System.out.print("\n1. Delete an item\n2. Add item at end\n3. Add
item at specific position\n4. Sort the list\n5. Print the shopping
list\n6. Exit\nEnter a choice: ");
      c=sc.nextInt();
      switch(c)
      {
        case 1:
          System.out.print("Enter id to remove: ");
          int id=sc.nextInt();
          for(int i=0;i<shop.size();i++){
            Shopping s=shop.get(i);
            if(id==s.id)
              shop.removeElement(s);
          }
          break;

        case 2:
          System.out.print("Enter item, item ID, cost and quantity to add:
");
          temp=sc.next();
          int id=sc.nextInt();
          double cost=sc.nextDouble();
          int quantity=sc.nextInt();
          Shopping s=new Shopping(temp, id, cost, quantity);
          shop.addElement(s);
          break;

        case 3:
          System.out.print("Enter item, item ID, cost, quantity and
location to add: ");
          temp=sc.next();
          id=sc.nextInt();
          cost=sc.nextDouble();
          quantity=sc.nextInt();
          int pos=sc.nextInt();
          Shopping sh=new Shopping(temp, id, cost, quantity);
          shop.add(pos-1,sh);
          break;

        case 4:
          sort(shop);
          break;

        case 5:
```

```
            display(shop);
            break;
        }
      }
      sc.close();
    }
    public static void sort(Vector<Shopping> shop){
      Comparator<Shopping> comparator=(Comparator<Shopping>) new Sorter();
      Collections.sort(shop, comparator);
      display(shop);
    }
    public static void display(Vector<Shopping> shop){
      for(int i=0;i<shop.size();i++){
        Shopping s=shop.get(i);
        System.out.println("Name: "+s.name+"\nID: "+(int)s.id+"\nPrice:
"+(double)s.cost+"\nQuantity: "+(int)s.count);
      }
    }
}
```

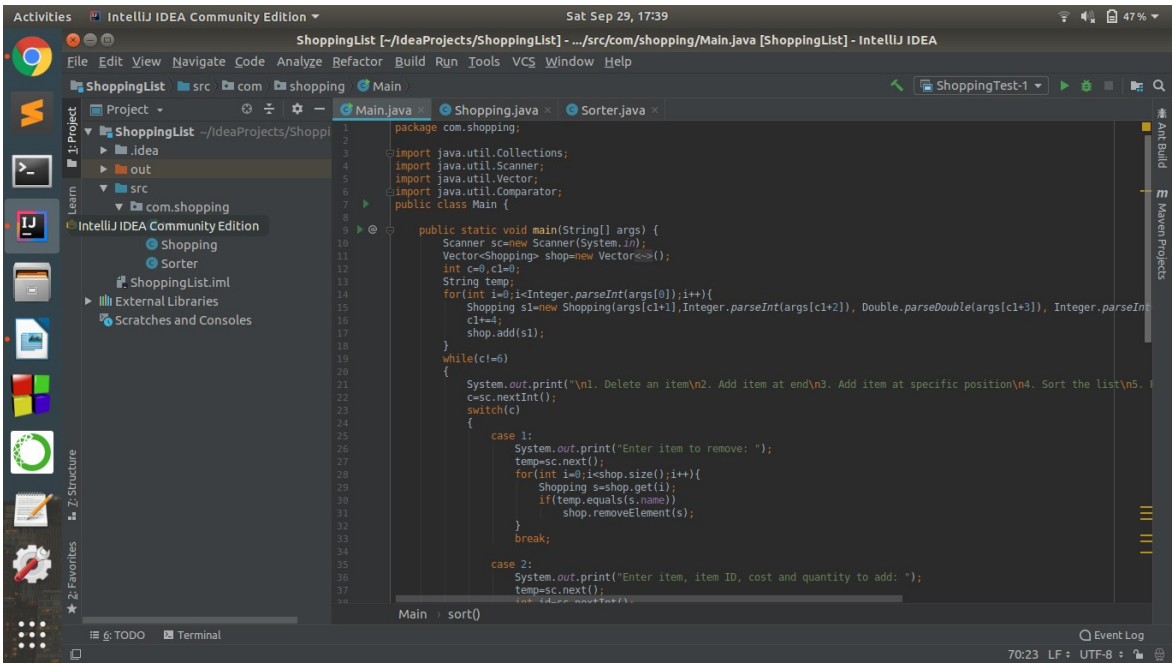
Shopping Class:

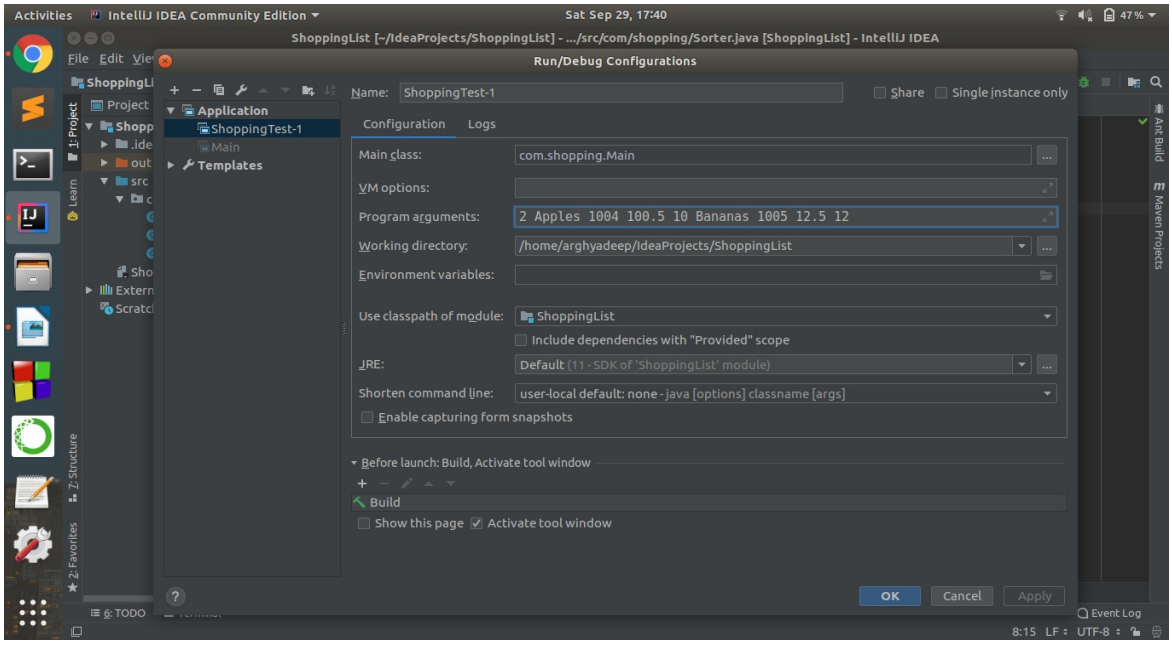
```
public class Shopping {
    String name;
    int id;
    double cost;
    int count;
    Shopping(String name, int id, double cost,int count){
      this.name=name;
      this.id=id;
      this.cost=cost;
      this.count=count;
    }
}
```

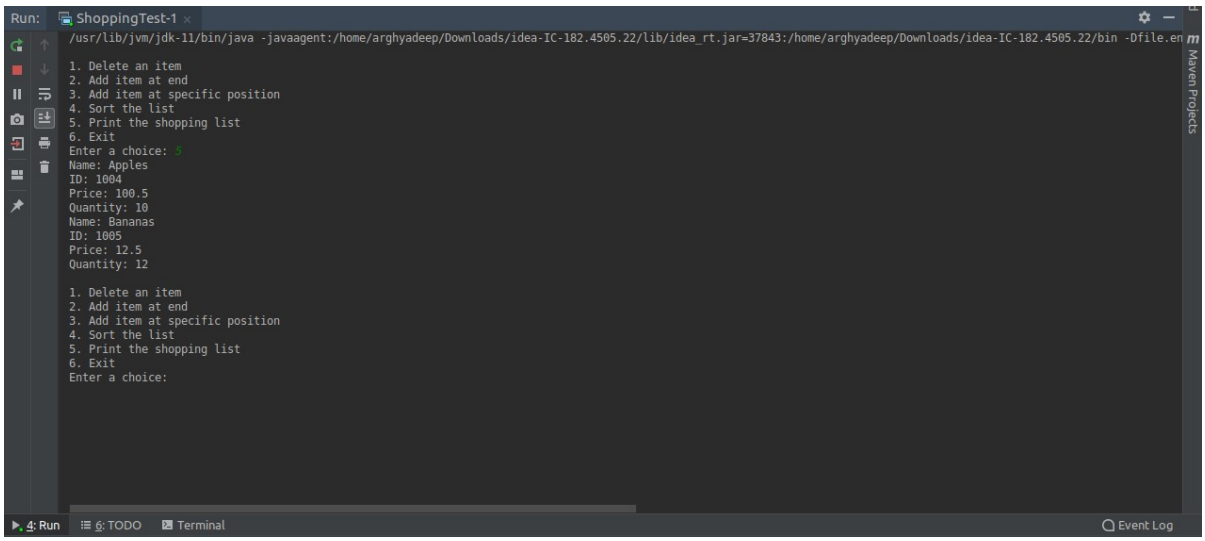
Sorter Class:

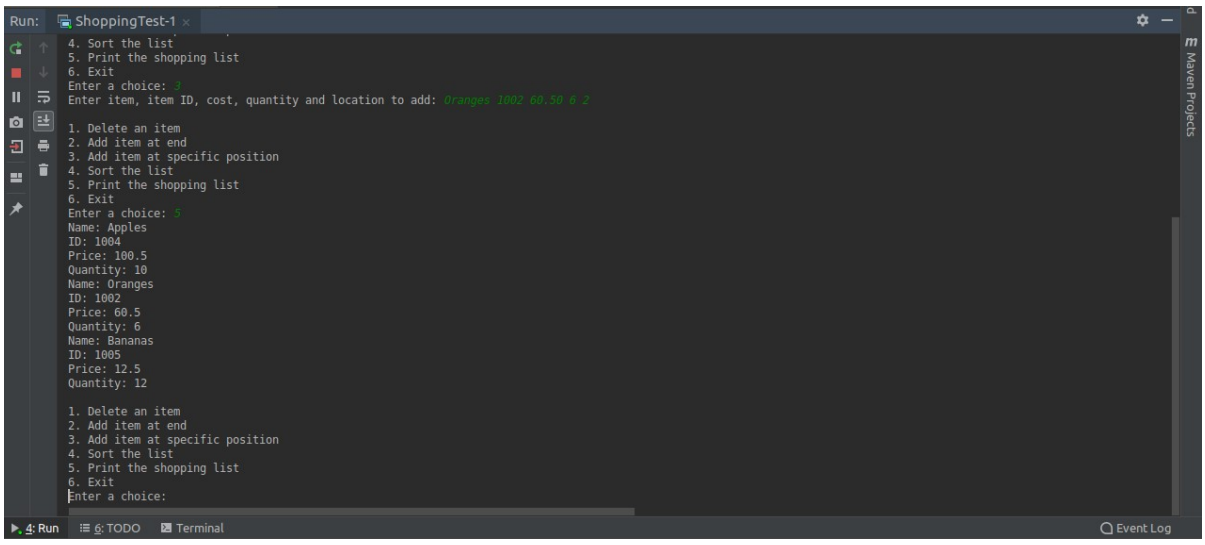
```
import java.util.Comparator;
class Sorter implements Comparator<Shopping> {

  @Override
  public int compare(Shopping s1, Shopping s2) {
    return s2.id-s1.id;
  }
}
```

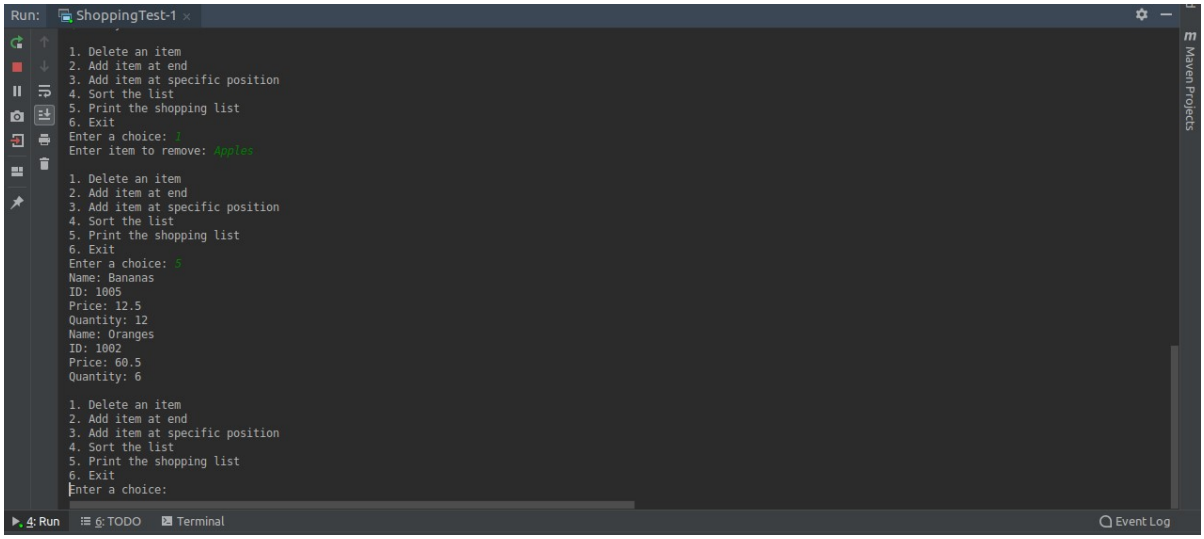
Output Screens:

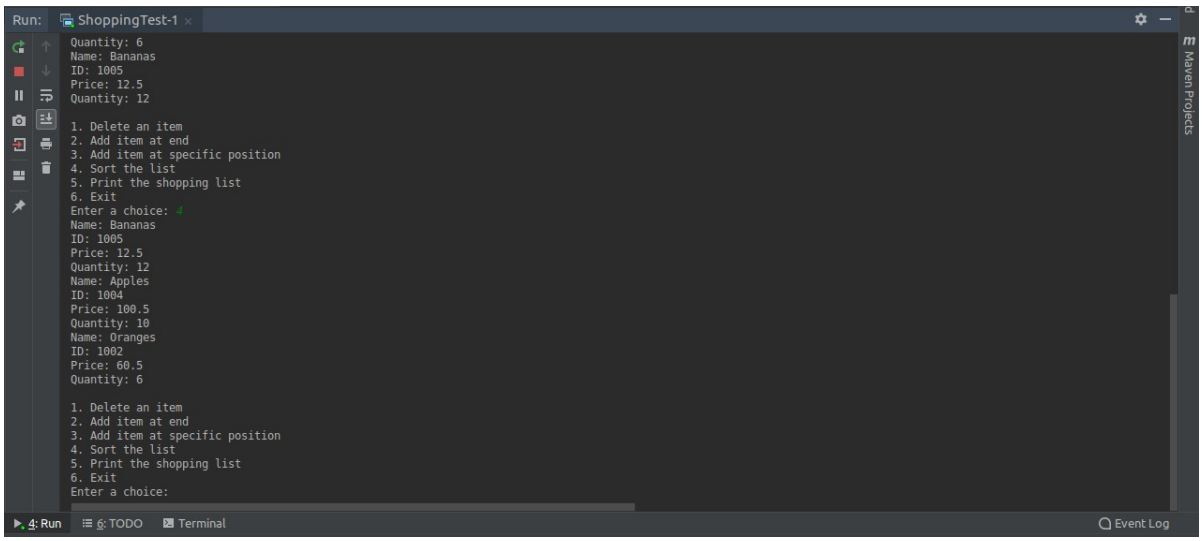



Activities
IntelliJ IDEA Community Edition
Sat Sep 29, 17:40
ShoppingList [~/IdeaProjects/ShoppingList] - .../src/com/shopping/Sorter.java [ShoppingList] - IntelliJ IDEA
Run/Debug Configurations
Name: ShoppingTest-1
Share
Single instance only
Application
ShoppingTest-1
Main
Templates
Configuration
Logs
Main class: com.shopping.Main
VM options:
Program arguments: 2 Apples 1004 100.5 10 Bananas 1005 12.5 12
Working directory: /home/arghyadeep/IdeaProjects/ShoppingList
Environment variables:
Use classpath of module: ShoppingList
Include dependencies with "Provided" scope
JRE: Default (11 - SDK of 'ShoppingList' module)
Shorten command line: user-local default: none - java [options] classname [args]
Enable capturing form snapshots
Before launch: Build, Activate tool window
Build
Show this page
Activate tool window
OK
Cancel
Apply


Run: ShoppingTest-1
/usr/lib/jvm/jdk-11/bin/java -javaagent:/home/arghyadeep/Downloads/idea-IC-182.4505.22/lib/idea_rt.jar=37843:/home/arghyadeep/Downloads/idea-IC-182.4505.22/bin -Dfile.en
1. Delete an item
2. Add item at end
3. Add item at specific position
4. Sort the list
5. Print the shopping list
6. Exit
Enter a choice: 5
Name: Apples
ID: 1004
Price: 100.5
Quantity: 10
Name: Bananas
ID: 1005
Price: 12.5
Quantity: 12
1. Delete an item
2. Add item at end
3. Add item at specific position
4. Sort the list
5. Print the shopping list
6. Exit
Enter a choice:
4: Run
6: TODO
Terminal
Event Log


Run: ShoppingTest-1
4. Sort the list
5. Print the shopping list
6. Exit
Enter a choice: 3
Enter item, item ID, cost, quantity and location to add: Oranges 1002 60.50 6 2
1. Delete an item
2. Add item at end
3. Add item at specific position
4. Sort the list
5. Print the shopping list
6. Exit
Enter a choice: 5
Name: Apples
ID: 1004
Price: 100.5
Quantity: 10
Name: Oranges
ID: 1002
Price: 60.5
Quantity: 6
Name: Bananas
ID: 1005
Price: 12.5
Quantity: 12
1. Delete an item
2. Add item at end
3. Add item at specific position
4. Sort the list
5. Print the shopping list
6. Exit
Enter a choice:
4: Run
6: TODO
Terminal
Event Log





Conclusion:

The program was developed on ***IntelliJ IDEA Community 2018.2.3 IDE*** and the program ran successfully as we were able to accept arguments for vector through command line and were able to handle all test and edge cases.

Date: 28/09/2018 **Signature of faculty in-charge**

Post Lab Descriptive Questions (Add questions from examination point view)

1. **Explain methods of Vector class in detail with the help of examples.**

Ans. The various methods of Vector class are:

1. **void add(int index, Object element)**

   Inserts the specified element at the specified position in this Vector.

   Example: v.add(1, "Adam");

2. **boolean add(Object o)**

   Appends the specified element to the end of this Vector.

   Example: v.add("John");

3. **boolean addAll(int index, Collection c)**

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

   Example: v.addAll(3, arrlist);

4. **void addElement(Object obj)**

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

   Example: v.addElement("Henry");

5. **int capacity()**

   Returns the current capacity of this vector.

   Example: v.capacity();

6. **void clear()**

   Removes all of the elements from this vector.

   Example: v.clear();

7. **Object clone()**

   Returns a clone of this vector.

   Example: Vector vec=v.clone();

8. **boolean contains(Object elem)**

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

   Example: v.contains("Adam");

9. **void copyInto(Object[] anArray)**

   Copies the components of this vector into the specified array.

   Example: v.copyInto(arr);

10. **Object elementAt(int index)**

    Returns the component at the specified index.

    Example: v.elementAt(2);

11. **void ensureCapacity(int minCapacity)**

    Increases the capacity of this vector, if necessary, to ensure that it can hold at least the number of components specified by the minimum capacity argument.

    Example: v.ensureCapacity(v.size());

12. **int indexOf(Object elem, int index)**

    Searches for the first occurence of the given argument, beginning the search at index, and testing for equality using the equals method.

    Example: v.indexOf("John",8);

13. **boolean isEmpty()**

    Tests if this vector has no components.

    Example: v.isEmpty();

14. **Object remove(int index)**

    Removes the element at the specified position in this vector.

    Example: v. remove(3);

15. **int size()**

    Returns the number of components in this vector. Example: v.size();