Experiment No. 15

Title: Write a program to demonstrate Collection and Generics. .

Objectives: 1 To learn Collections and Generics

Theory:

Collections in Java

Collections in java are a framework that provides architecture to store and manipulate the group of objects. All the operations that you perform on a data such as searching, sorting, insertion, manipulation, deletion etc. can be performed by Java Collections.

Java Collection simply means a single unit of objects. Java Collection framework provides many interfaces (Set, List, Queue, Deque etc.) and classes (ArrayList, Vector, LinkedList, PriorityQueue, HashSet, LinkedHashSet, TreeSet etc).

Hierarchy of Collection Framework

The java.util package contains all the classes and interfaces for Collection framework.

Generic Programming:

Generic programming means to write code that can be reused for objects of many different types.Example, you don't want to program separate classes to collect String & File objects.

A generic class is a class with one or more type variables. Following class allows us to focus on generics.

```
public class Pair<T>
{
        public Pair() { first = null; second = null; }
        public Pair(T first, T second) { this.first = first; this.second = second; }

        public T getFirst() { return first; }
        public T getSecond() { return second; }

        public void setFirst(T newValue) { first = newValue; }
        public void setSecond(T newValue) { second = newValue; }

        private T first;
        private T second;
}
```

The Pair class introduces a type variable T,enclosed in < >,after class name. The type variable is used through class defination to specify method return types & types of fields & local variables.

Example: In above class, **private T first**;

You can instantiate generic type by substituting types for the type variable,

Example: Pair<String> **T** will have type String.

Generic Methods:
You can define generic method inside ordinary classes or generic classes.

Example: Following class defines generic method in ordinary class.

```
Class ArrayAlgo
{
        Public static <T> T getMiddle(T[] a)
        {
                Return a[a.length/2]
        }
}
```

When you call generic methods, you place the actual types, enclosed in angle bracket, before method name. As below:

```
String[] names={"John","Q","Public"};
String middle=ArrayAlgo.<String>getMiddle(names);
```

Compiler matches type of names against generic type t[] and deduces T must be String so you can omit the <String > type parameter from method call.

Java ArrayList Class

An ArrayList is like ail array, which can grow in memory dynamically. It means that when we store elements into the ArrayList, depending on the number of elements, the memory is dynamically allotted and re-allotted to accommodate all the elements. ArrayList is not synchronized. This means that when more than one thread acts simultaneously on the ArrayList object, the results may be incorrect in some cases.
The ArrayList class can be written as:

class ArrayList <E>

where E represents the type of elements to be stored into the ArrayList. For example, to store string type elements, we can create an object to ArrayList as:

ArrayList <String>arl=new ArrayList<String>();

The preceding statement constructs an ArrayList with a default initial capacity of 10. We can also mention the capacity at the time of creating ArrayList object as:

ArrayList <Double>arl= new ArrayList<Double>(101);

The preceding ArrayList can store Double type objects and initial capacity is declared to be 101.
ArrayList Class Methods-
1]boolean add (element obj): This method appends the specified element to the end of the ArrayList. If the element is added successfully then the preceding method returns true.
2] **void add (int position, element obj) :** This method inserts the specified element at the specified position in the ArrayList.
3] **element remove(int position)-:** This method removes the element at the specified position

in the ArrayList. This method also returns the element which was removed from the ArrayList.
4] **boolean remove (Object obj):** This method removes the first occurrence of the specified element obj from the ArrayList, if it is present.
5] **void clear ()** : This method removes all the elements from the ArrayList.
6] **element set (int pos i tion, element obj):** This method replaces an element at the specified position.in the ArrayList with the specified element obj .
7] **boolean contains (Object obj):** This method returns true if the ArrayList contains the specified element obj.
8] **element get (int position):** This method returns the element available at the specified position in the ArrayList.
9] **int indexOf (Object obj):** This method returns the position of the first occurrence of the specified element obj in the list, or -1 if the element is not found in the list.
10**] int lastIndexOf (Object obj):** This method returns the position of the last occurrence of the specified element obj in the list, or -1 if the element is not found in the list.
11**] int size ()** : This method returns the number of elements present in the ArrayList.
12] **Object [ ] toArray ():** This method returns an Object class type array containing all the elements in the ArrayList in proper sequence..

Example:

```
ArrayList<String> arl=new ArrayList<String>();      //create Arralylist
arl.add("Apple");         //add 4 objects
arl.add("Mango");
arl.add("Grapes");
arl.add("Guava");
System.out.println("Contents:"+arl);
arl.remove(3);     //remove object
arl.remove("Apple");
System.out.println("Contents after removing:"+arl);
System.out.println("Size of arraylist"+arl.size());
System.out.println("Extracting using Iterator:");
Iterator it=arl.iterator();    //add an Iterator to Arraylist to retrieve elements
While(it.hasNext())
{
     System.out.println(it.next);  }
```

Key concepts: Generic, Generic class, Generic methods,ArrayList

Algorithm:

1. Create a class ArrayListDemo.
2. Using add method insert data into ArrayList i.e. BookID,BookName,Author,Publisher,Quantity.
3. Display size of ArrayList.
4. Display All elements from ArrayList.
5. Remove one element from ArrayList.
6. Display elements after removing.