MIT World Peace University

Object Oriented Programming with Java and C++
Second Year B. Tech, Semester 1

Collection Frameworks in Java

Practical Report
Assignment 6

Prepared By

Krishnaraj Thadesar
Cyber Security and Forensics
Batch A1, PA 20

November 25, 2022

Contents

1 Aim and Objectives

Aim

Demonstrating the use of Collection Frameworks in Java.

Objective

- To study the concept of collection framework
- To get familiar with features of collection framework in Java

2 Problem Statement

Write a Java Program to:

- Create a new array list and print out the collection by position
- Add some elements (string)
- Search an element in an array list
- Reverse elements in an array list
- Test an array list is empty or not
- Join two array lists

3 Theory

3.1 Concept of collection framework in Java

The Collection in Java is a framework that provides an architecture to store and manipulate the group of objects.

Java Collections can achieve all the operations that you perform on a data such as searching, sorting, insertion, manipulation, and deletion.

Here is an Example using the ArrayList interface.

```java
import java.util.*;
class TestJavaCollection1{
  public static void main(String args[]){
    ArrayList<String> list=new ArrayList<String>();//Creating arraylist
    list.add("Ravi");//Adding object in arraylist
    list.add("Vijay");
    list.add("Ravi");
    list.add("Ajay");
    //Traversing list through Iterator
    Iterator itr=list.iterator();
    while(itr.hasNext()){
      System.out.println(itr.next());
    }
  }
}
```

3.2 Benefit of Generics in Collections Framework

The Java Generics programming is introduced in J2SE 5 to deal with type-safe objects. It makes the code stable by detecting the bugs at compile time.

Before generics, we can store any type of objects in the collection, i.e., non-generic. Now generics force the java programmer to store a specific type of objects.

The Advantages are:

- Type-safety : We can hold only a single type of objects in generics. It doesn?t allow to store other objects.
- Type casting is not required: There is no need to typecast the object.
- Compile-time checking: It is checked at compile time so problem will not occur at runtime. It is far better to handle the problem at compile time than runtime.

3.3 Basic interfaces of Java Collections Framework

Java Collection means a single unit of objects. Java Collection framework provides many basic interfaces.

- ArrayList: It is used to create a dynamic array. It is similar to vector.
- LinkedList: It is used to create a linked list. It is the implementation of the List and Deque interfaces.
- Vector: It is similar to ArrayList. It is also used to create a dynamic array.
- HashSet: It is used to create a collection that uses a hash table for storage.
- TreeSet: It is used to create a collection that uses a tree for storage.
- HashMap: It is used to create a map that uses a hash table for storage.
- TreeMap: It is used to create a map that uses a tree for storage.

3.4 Types of collections allowed by the Java collections framework, Explain with suitable syntax and examples

4 Platform

Operating System: Arch Linux x86-64
IDEs or Text Editors Used: Visual Studio Code
Compilers : g++ and gcc on linux for C++, and javac, with JDK 18.0.2 for Java

5 Input

For C++

1. Number of Each Type of Employee
2. Name, Age, Address City, and Salary of Each Employee

For Java

1. The Side of the Square
2. The Radius of the Circle
3. The Length and Breadth of the Rectangle.

6 Output

For C++

1. General Information about Each Employee
2. The Weekly, hourly and commisioned Salary for Respective Employees.

For Java

1. The Area of the Shapes
2. The Location of the Hill Stations
3. The Reason the Hill stations are Famous for.

7 Code

```java
// Write a Java program to
// create a new array list and print out the collection by position
// add some elements (string)
// search an element in an array list
// reverse elements in an array list
// test an array list is empty or not
// join two array lists

// Krishnaraj Thadesar
// PA20
// Batch A1
// Assignment 6 in OOPCJ

import java.util.*;

import javax.lang.model.util.ElementScanner14;

import java.lang.*;

public class assignment_6 {
    static Scanner input = new Scanner(System.in);
    static ArrayList<String> arrayList, list1, list2;
    static String list1_input, list2_input, main_list_input;

    public static void main(String[] args) {
        System.out.println("Welcome to Assignment 6! What do you want to do? ");
        int choice = 0;
        arrayList = new ArrayList<String>();
        while (choice != 7) {
            System.out.println(
                    "\n\n1. Add some elements (string)\n2. Search an element in an array list\n3. Reverse elements in an array list\n4. Test an array list is empty or not\n5. Join two array lists\n6. Display All Elements\n7. Quit\n");
            choice = input.nextInt();
            switch (choice) {
                case 1:
                    System.out.println("Enter the Element that you want to enter: ");
                    main_list_input = input.next();
                    arrayList.add(main_list_input);
                    break;
                case 2:
                    System.out.println("Enter the Element that you want to Search: ");
                    main_list_input = input.next();
                    // search an element in ArrayList
                    if (arrayList.contains(main_list_input)) {
                        System.out.println("Element found");
                    } else {
                        System.out.println("Element not found");
                    }
                    break;
                case 4:
                    System.out.println("Checking if Arraylist is Empty: ");
                    if (arrayList.isEmpty()) {
                        System.out.println("Array list is empty!");
                    } else {
                        System.out.println("Array list isnt empty!");
                    }
                    break;
                case 5:
                    System.out.println("Joining 2 Array lists, so enter them: ");
                    System.out.println("Enter List 1, enter \"done\" to stop entering");
                    list1 = new ArrayList<String>();
                    list1_input = new String();
                    while (!list1_input.equals("done")) {
                        list1_input = input.next();
                        if (!list1_input.equals("done")) {
                            list1.add(list1_input);
                        }
                    }
                    System.out.println("Enter List 2, enter \"done\" to stop entering");
                    list2 = new ArrayList<String>();
                    list2_input = new String();
                    while (!list2_input.equals("done")) {
                        list2_input = input.next();
                        if (!list2_input.equals("done")) {
                            list2.add(list2_input);
                        }
                    }
                    // merge 2 Arraylists
                    list1.addAll(list2);
                    System.out.println("Displaying all the elements");
                    for (String iterable_element : list1) {
                        System.out.println(iterable_element);
                    }
```

```java
                        break;
                    case 6:
                        System.out.println("Displaying all the elements");
                        for (String iterable_element : arrayList) {
                            System.out.println(iterable_element);
                        }
                        break;
                    case 3:
                        System.out.println("Reversing the Elements of the List");
                        Collections.reverse(arrayList);
                        System.out.println("Displaying all the elements");
                        for (String iterable_element : arrayList) {
                            System.out.println(iterable_element);
                        }
                        break;
                    case 7:
                        System.out.println("Quitting!");
                        System.exit(0);
                    default:
                        break;
                }
            }
        }
}
```

Listing 1: Output

7.0.1 Output

```

Welcome to Assignment 6! What do you want to do?


1. Add some elements (string)
2. Search an element in an array list
3. Reverse elements in an array list
4. Test an array list is empty or not
5. Join two array lists
6. Display All Elements
7. Quit

1
Enter the Element that you want to enter:
a


1. Add some elements (string)
2. Search an element in an array list
3. Reverse elements in an array list
4. Test an array list is empty or not
5. Join two array lists
6. Display All Elements
7. Quit

1
Enter the Element that you want to enter:
b


```

```
1. Add some elements (string)
2. Search an element in an array list
3. Reverse elements in an array list
4. Test an array list is empty or not
5. Join two array lists
6. Display All Elements
7. Quit

1
Enter the Element that you want to enter:
c


1. Add some elements (string)
2. Search an element in an array list
3. Reverse elements in an array list
4. Test an array list is empty or not
5. Join two array lists
6. Display All Elements
7. Quit

1
Enter the Element that you want to enter:
4


1. Add some elements (string)
2. Search an element in an array list
3. Reverse elements in an array list
4. Test an array list is empty or not
5. Join two array lists
6. Display All Elements
7. Quit

2
Enter the Element that you want to Search:
a
Element found


1. Add some elements (string)
2. Search an element in an array list
3. Reverse elements in an array list
4. Test an array list is empty or not
5. Join two array lists
6. Display All Elements
7. Quit

3
Reversing the Elements of the List
Displaying all the elements
4
c
b
a


1. Add some elements (string)
2. Search an element in an array list
```

```
3. Reverse elements in an array list
4. Test an array list is empty or not
5. Join two array lists
6. Display All Elements
7. Quit

4
Checking if Arraylist is Empty:
Array list isnt empty!


1. Add some elements (string)
2. Search an element in an array list
3. Reverse elements in an array list
4. Test an array list is empty or not
5. Join two array lists
6. Display All Elements
7. Quit

5
Joining 2 Array lists, so enter them:
Enter List 1, enter "done" to stop entering
1
2
3
done
Enter List 2, enter "done" to stop entering
33
4
55
done
Displaying all the elements
1
2
3
33
4
55


1. Add some elements (string)
2. Search an element in an array list
3. Reverse elements in an array list
4. Test an array list is empty or not
5. Join two array lists
6. Display All Elements
7. Quit


6
Displaying all the elements
4
c
b
a


1. Add some elements (string)
2. Search an element in an array list
```

```
3. Reverse elements in an array list
4. Test an array list is empty or not
5. Join two array lists
6. Display All Elements
7. Quit

7
Quitting!
```

Listing 2: Output

8 Conclusion

Thus, learnt the use of collection framework in java and performed array list operations

9 FAQs

1. **Why do we use collection framework?**
   Java Collection Framework offers the capability to Java Collection to represent a group of elements in classes and Interfaces.

   Java Collection Framework enables the user to perform various data manipulation operations like storing data, searching, sorting, insertion, deletion, and updating of data on the group of elements. Followed by the Java Collections Framework, you must learn and understand the Hierarchy of Java collections and various descendants or classes and interfaces involved in the Java Collections.

   - When input size is dynamic.
   - Whenever we are required to store heterogeneous data.
   - When data grows and shrinks frequently.

2. **Which is best collection framework in Java?**
   There are several very important collection frameworks, in Java. They are:

   - ArrayList
   - LinkedList
   - Vector
   - HashSet
   - TreeSet
   - HashMap
   - TreeMap

   All of them are important, and useful in their own way. So we cannot say one of them is the best one, as it depends on the requirements.

3. **What is difference between array and collection?**

   - Arrays are fixed in size, whereas collections are dynamic in size.
   - Arrays are homogeneous, whereas collections are heterogeneous.
   - Arrays are not type safe, whereas collections are type safe.
   - Arrays are not thread safe, whereas collections are thread safe.
   - Arrays are not synchronized, whereas collections are synchronized.

4. **What is HashMap in Java?**
   HashMap is a part of the Java Collections Framework. It is a class that implements the Map interface. It is used to store key-value pairs. It is similar to Hashtable, but it is not synchronized. It allows one null key and multiple null values. It maintains no order.

   It provides the functionality of the hash table data structure. It stores elements in key/value pairs. Here, keys are unique identifiers used to associate each value on a map.