

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

<b>1 Aim and Objectives</b>	<b>1</b>
<b>2 Problem Statement</b>	<b>1</b>
<b>3 Theory</b>	<b>1</b>
3.1 Concept of collection framework in Java . . . . .	1
3.2 Benefit of Generics in Collections Framework . . . . .	2
3.3 Basic interfaces of Java Collections Framework . . . . .	2
3.4 Types of collections allowed by the Java collections framework, Explain with suitable syntax and examples . . . . .	2
<b>4 Platform</b>	<b>2</b>
<b>5 Input</b>	<b>2</b>
<b>6 Output</b>	<b>3</b>
<b>7 Code</b>	<b>3</b>
7.0.1 Output . . . . .	5
<b>8 Conclusion</b>	<b>8</b>
<b>9 FAQs</b>	<b>9</b>

# 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.

```
1 import java.util.*;
2 class TestJavaCollection1{
3     public static void main(String args[]){
4         ArrayList<String> list=new ArrayList<String>();//Creating arraylist
5         list.add("Ravi");//Adding object in arraylist
6         list.add("Vijay");
7         list.add("Ravi");
8         list.add("Ajay");
9         //Traversing list through Iterator
10        Iterator itr=list.iterator();
11        while(itr.hasNext()){
12            System.out.println(itr.next());
13        }
14    }
15 }
```

### **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

```
1 // Write a Java program to
2 // create a new array list and print out the collection by position
3 // add some elements (string)
4 // search an element in an array list
5 // reverse elements in an array list
6 // test an array list is empty or not
7 // join two array lists
8
9 // Krishnaraj Thadesar
10 // PA20
11 // Batch A1
12 // Assignment 6 in OOPJC
13
14 import java.util.*;
15
16 import javax.lang.model.util.ElementScanner14;
17
18 import java.lang.*;
19
20 public class assignment_6 {
21     static Scanner input = new Scanner(System.in);
22     static ArrayList<String> arrayList, list1, list2;
23     static String list1_input, list2_input, main_list_input;
24
25     public static void main(String[] args) {
26         System.out.println("Welcome to Assignment 6! What do you want to do? ");
27         int choice = 0;
28         arrayList = new ArrayList<String>();
29         while (choice != 7) {
```

```
30         System.out.println(
31             "\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");
32         choice = input.nextInt();
33         switch (choice) {
34             case 1:
35                 System.out.println("Enter the Element that you want to enter:
           ");
36                 main_list_input = input.next();
37                 arrayList.add(main_list_input);
38                 break;
39             case 2:
40                 System.out.println("Enter the Element that you want to Search:
           ");
41                 main_list_input = input.next();
42                 // search an element in ArrayList
43                 if (arrayList.contains(main_list_input)) {
44                     System.out.println("Element found");
45                 } else {
46                     System.out.println("Element not found");
47                 }
48                 break;
49             case 4:
50                 System.out.println("Checking if Arraylist is Empty: ");
51                 if (arrayList.isEmpty()) {
52                     System.out.println("Array list is empty!");
53                 } else {
54                     System.out.println("Array list isnt empty!");
55                 }
56                 break;
57             case 5:
58                 System.out.println("Joining 2 Array lists, so enter them: ");
59                 System.out.println("Enter List 1, enter \"done\" to stop
           entering");
60                 list1 = new ArrayList<String>();
61                 list1_input = new String();
62                 while (!list1_input.equals("done")) {
63                     list1_input = input.next();
64                     if (!list1_input.equals("done")) {
65                         list1.add(list1_input);
66                     }
67                 }
68                 System.out.println("Enter List 2, enter \"done\" to stop
           entering");
69                 list2 = new ArrayList<String>();
70                 list2_input = new String();
71                 while (!list2_input.equals("done")) {
72                     list2_input = input.next();
73                     if (!list2_input.equals("done")) {
74                         list2.add(list2_input);
75                     }
76                 }
77                 // merge 2 Arraylists
78                 list1.addAll(list2);
79                 System.out.println("Displaying all the elements");
80                 for (String iterable_element : list1) {
81                     System.out.println(iterable_element);
82                 }
```

```
83         break;
84     case 6:
85         System.out.println("Displaying all the elements");
86         for (String iterable_element : arrayList) {
87             System.out.println(iterable_element);
88         }
89         break;
90     case 3:
91         System.out.println("Reversing the Elements of the List");
92         Collections.reverse(arrayList);
93         System.out.println("Displaying all the elements");
94         for (String iterable_element : arrayList) {
95             System.out.println(iterable_element);
96         }
97         break;
98     case 7:
99         System.out.println("Quitting!");
100        System.exit(0);
101    default:
102        break;
103    }
104 }
105 }
106 }
```

Listing 1: Output

### 7.0.1 Output

```
1
2 Welcome to Assignment 6! What do you want to do?
3
4
5 1. Add some elements (string)
6 2. Search an element in an array list
7 3. Reverse elements in an array list
8 4. Test an array list is empty or not
9 5. Join two array lists
10 6. Display All Elements
11 7. Quit
12
13 1
14 Enter the Element that you want to enter:
15 a
16
17
18 1. Add some elements (string)
19 2. Search an element in an array list
20 3. Reverse elements in an array list
21 4. Test an array list is empty or not
22 5. Join two array lists
23 6. Display All Elements
24 7. Quit
25
26 1
27 Enter the Element that you want to enter:
28 b
29
30
```

```
31 1. Add some elements (string)
32 2. Search an element in an array list
33 3. Reverse elements in an array list
34 4. Test an array list is empty or not
35 5. Join two array lists
36 6. Display All Elements
37 7. Quit
38
39 1
40 Enter the Element that you want to enter:
41 c
42
43
44 1. Add some elements (string)
45 2. Search an element in an array list
46 3. Reverse elements in an array list
47 4. Test an array list is empty or not
48 5. Join two array lists
49 6. Display All Elements
50 7. Quit
51
52 1
53 Enter the Element that you want to enter:
54 4
55
56
57 1. Add some elements (string)
58 2. Search an element in an array list
59 3. Reverse elements in an array list
60 4. Test an array list is empty or not
61 5. Join two array lists
62 6. Display All Elements
63 7. Quit
64
65 2
66 Enter the Element that you want to Search:
67 a
68 Element found
69
70
71 1. Add some elements (string)
72 2. Search an element in an array list
73 3. Reverse elements in an array list
74 4. Test an array list is empty or not
75 5. Join two array lists
76 6. Display All Elements
77 7. Quit
78
79 3
80 Reversing the Elements of the List
81 Displaying all the elements
82 4
83 c
84 b
85 a
86
87
88 1. Add some elements (string)
89 2. Search an element in an array list
```



```
90 3. Reverse elements in an array list
91 4. Test an array list is empty or not
92 5. Join two array lists
93 6. Display All Elements
94 7. Quit
95
96 4
97 Checking if ArrayList is Empty:
98 Array list isnt empty!
99
100
101 1. Add some elements (string)
102 2. Search an element in an array list
103 3. Reverse elements in an array list
104 4. Test an array list is empty or not
105 5. Join two array lists
106 6. Display All Elements
107 7. Quit
108
109 5
110 Joining 2 Array lists, so enter them:
111 Enter List 1, enter "done" to stop entering
112 1
113 2
114 3
115 done
116 Enter List 2, enter "done" to stop entering
117 33
118 4
119 55
120 done
121 Displaying all the elements
122 1
123 2
124 3
125 33
126 4
127 55
128
129
130 1. Add some elements (string)
131 2. Search an element in an array list
132 3. Reverse elements in an array list
133 4. Test an array list is empty or not
134 5. Join two array lists
135 6. Display All Elements
136 7. Quit
137
138
139 6
140 Displaying all the elements
141 4
142 c
143 b
144 a
145
146
147 1. Add some elements (string)
148 2. Search an element in an array list
```

```
149 3. Reverse elements in an array list
150 4. Test an array list is empty or not
151 5. Join two array lists
152 6. Display All Elements
153 7. Quit
154
155 7
156 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.