

ASSINGMENT

PROGRAMMING IN JAVA



Submitted To

Puneet Kumar , Assistant Professor


Lovely Professional University

Jalandhar, Punjab, India.

Submitted By

SL.NO	Registration Number	Student Name	Roll No	Obtain Mark	Signature
01.	11815813	Shykat Roy	RK18PTB42		Shykat

Q1. Write a Complete chapter on Collections in java which includes Definition, importance and uses.

Date.....	Page No.....
<u>Collection in Java</u>	
The collection in Java is a framework that provides an architecture to store and manipulate the group of object.	
Java Collections can achieve all the operations that you perform on a data such as searching, sorting, manipulation and deletion.	
Java collection means a single unit of object. Java collection framework provides many interfaces (Set, List, Queue, Deque) and classes (ArrayList, Vector, LinkedList, PriorityQueue, HashSet, LinkedHashSet, TreeSet).	
⇒ <u>What is collection in Java?</u> A collection represents a single unit of object, i.e., a group.	
⇒ <u>What is a framework in Java?</u> <ul style="list-style-type: none">① It provides readmade architecture.② It represents a set of classes and interfaces.③ It is optional.	
	Signature of Supervisor :

Methods

- 1) Public boolean add() → It is used to insert an element.
- 2) Public boolean addAll(Collection <? Expected E> c) → It is used to insert the specified collection elements.
- 3) Public boolean remove() → used to delete an element.
- 4) " removeAll() → used to remove specified collection.
- 5) Public int size() → It returns the total number of element in the collection.
- 6) Public void clear() → used to remove total number of elements from the collection.
- 7) Public boolean contains() → It is used to search an element.
- 8) Public boolean containsAll() → It is used to search the specified collection in the collection.
- 9) Public Iterator iterator() → It returns an iterator.
- 10) public Object[] toArray() → It converts collection into array.



- 11) public boolean isEmpty() → To check if collection is empty.
- 12) public boolean equals() → It matches two collection.
- 13) public int hashCode() → It returns the hash code numbers of the collection.

List Interface

To instantiate the List interface :-

- i) List <data-type> list1 = new ArrayList()
- ii) List <data-type> list2 = new LinkedList()
- iii) List <data-type> list3 = new Vector()
- iv) List <data-type> list4 = new Stack()

Ex:-

```
import java.util.*;  
class Test  
{  
    public static void main (String a[])  
    {  
        ArrayList<String> list = new ArrayList<String>();  
        list.add ("Raj");  
        list.add ("Vijai");  
        list.add ("Ajai");  
        Iterator itr = list.iterator();  
    }  
}
```



L
P
U

Signature of Supervisor :

Date.....

Page No.....

```
while(itr.hasNext())  
{  
    System.out.println(itr.next());  
}  
}
```

LinkedList

```
import java.util.*;  
public class Test  
{  
    public static void main(String a[])  
    {  
        LinkedList<String> list = new LinkedList<String>();  
        list.add("Ravi");  
        list.add("Raj");  
        list.add("Roy");  
        list.add("Aaj");  
        LinkedList<String> list;  
        Iterator<String> itr = list.iterator();  
        while(itr.hasNext())  
        {  
            System.out.println(itr.next());  
        }  
    }  
}
```



L
P
U

Signature of Supervisor :

2. Public boolean addAll(^{element}) →

Stack

```
import java.util
public class Test
{
    public static void main(String a[])
    {
        Stack<String> li = new Stack<String>();
        li.push("Ajush");
        li.push("Avi");
        li.push("Amit");
        li.push("Chani");
        li.pop();
        Iterator<String> itr = Stackli.iterator();
        while(itr.hasNext())
        {
            System.out.println(itr.next());
        }
    }
}
```



L
P
U

Signature of Supervisor :

Vector

```
import java.util.*;  
public class Test  
{  
    public static void main(String a[])  
    {  
        Vector<String> v = new Vector<String>();  
        v.add("Roy");  
        v.add("Amit");  
        v.add("Raj");  
        v.add("Ashis");  
        Iterator<String> itr = v.iterator();  
        while (itr.hasNext())  
        {  
            System.out.println(itr.next());  
        }  
    }  
}
```

ate.....

Queue Interface

i) `Queue<String> q1 = new PriorityQueue();`
 ii) `Queue<String> q2 = new ArrayDeque();`

Priority Queue

```
import java.util.*;
public class test
{
    PriorityQueue<String> queue = new PriorityQueue<String>();

    queue.add("Amit");
    queue.add("Roy");
    queue.add("Rohan");
    queue.add("Raj");
    System.out.println("head:" + queue.element());
    System.out.println("head:" + queue.element());
    System.out.println("iterating the queue elements:");
    Iterator itr = queue.iterator();
    while (itr.hasNext())
    {
        System.out.println(itr.next());
    }
    queue.remove();
    queue.poll();
}
```

L
P
U

Signature of Supervisor :


```
system.out.println("after removing two  
elements:");
```

```
Iterator<String> itr2 = queue.iterator();
```

```
while(itr2.hasNext())
```

```
{
```

```
    system.out.println(itr2.next());
```

```
}
```

```
}
```

```
}
```



L
P
U

Signature of Supervisor :

Date.....

Page No.....

ArrayDeque

```
import java.util.*;  
public class Test  
{  
    public static void main (String a[])  
    {  
        Deque<String> deque = new ArrayDeque<String>  
            ();  
  
        deque.add ("Hani");  
        deque.add ("Roy");  
        deque.add ("Ajay");  
        for (String str : deque)  
        {  
            System.out.println (str);  
        }  
    }  
}
```

Set Interface

- i) `Set<data-type> s1 = new HashSet<data-type>();`
- ii) `Set<data-type> s2 = new LinkedHashSet<data-type>();`
- iii) `Set<data-type> s3 = new TreeSet<data-type>();`

HashSet

```
import java.util.*;
public class Test
{
    public static void main(String a[])
    {
        HashSet<String> Set = new HashSet<String>();
        Set.add("Raj");
        Set.add("Ravi");
        Set.add("Raju");
        Iterator<String> itr = Set.iterator();
        while (itr.hasNext())
        {
            System.out.println(itr.next());
        }
    }
}
```



Date.....

Page No.....

Tree Set

```
import java.util.*;  
public class Test  
{  
    public static void main (String a[])  
    {  
        TreeSet<String> set = new TreeSet<String>();  
        set.add("Ravi");  
        set.add("Raja");  
        set.add("Ajay");  
        Iterator<String> itr = set.iterator();  
        while (itr.hasNext())  
        {  
            System.out.println(itr.next());  
        }  
    }  
}
```

LinkedList

```
import java.util.*;
public class Test
{
    public static void main()
    {
        LinkedList<String> Set = new LinkedList<String>();
        Set.add("Raj");
        Set.add("Ravi");
        Set.add("Raju");
        Set.add("Rajd");
        Iterator<String> itr = Set.iterator();
        while(itr.hasNext())
        {
            System.out.println(itr.next());
        }
    }
}
```



Date.....

⇒ What is the difference between Java Collection and Java Collections?

Major difference between Collection and Collections is Collection is an interface and Collections is a root level interface of the Java Collection Framework. Most of the classes in Java Collection Framework inherit from this interface.

Advantages of Collection Framework

- ⇒ It reduces the development time and the burden of designers, programmers, and users.
- ⇒ Code is easier to maintain because it provides useful data structures and interfaces which reduce programming efforts.
- ⇒ The size of the container is growable in nature.
- ⇒ It implements high-performance of useful data structures and algorithms that increase the performance.
- ⇒ It enables software reuse.



Signature of Supervisor :

Date.....

Page No.....

Use of Collection in Java

Collections are used almost in every programming language and when Java arrived, it also came with collection classes. Collections are used in situations where data is dynamic. Collections allow adding an element, deleting an element and host of other operations. There are a number of collections in Java allowing to choose the right collection for the right context. You can play with data structure and algorithms.

Q2:- Write a program to perform union, intersection and difference of two different Array-Lists.

GIT-HUB-LINK:-

<https://github.com/shykat199/java-Unioninsertion>

Code:-

[illegible]

[illegible]


```
{
    for(int k=j;k<(size1-1);k++)
    {
        arr1[k]=arr1[k+1];
    }
    size1=size1-1;
}

}

for(int i=0;i<size1;i++)
{
    System.out.print(arr1[i]+" ");
}

System.out.println
(">>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>\n");

    System.out.print("\nArray Element of Array2\nafter applying remove duplicate logic: ");
    for(int i=0;i<(size2-1);i++)
    {
        for(int j=i+1;j<size2;j++)
        {
            if(arr2[i]==arr2[j])
            {
                for(int k=j;k<(size2-1);k++)
                {
                    arr2[k]=arr2[k+1];
```

```

        }
        size2=size2-1;
    }

}

for(int i=0;i<size2;i++)
{
    System.out.print(arr2[i]+" ");
}

System.out.println
(">>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>\n");

System.out.println(" ");
System.out.println("Enter   '1' for finding\nthe Union of given Array");
System.out.println("Enter   '2' for finding the\nintersection of given Array");
int p=s.nextInt();

switch(p)
{
case 1:
{
    int flag=0;
```

System.out.print("Union of Two Arrays:

```
");
```

```
for(int i=0;i<size1;i++)
{
    System.out.print(arr1[i]+" ");
}
for(int i = 0; i < size2; i++)
{
    for(int j = 0; j < size1; j++)
    {
        if(arr2[i] != arr1[j])
        {
            flag =1;

        }
        else
        {
            flag = 0;
            break;
        }

    }
    if(flag ==1)
    {
        System.out.print(arr2[i]+" ");
    }
}
break;
}
```



```

        case 2 :
        {
            System.out.println("Intersection    of
two given Array:");

            for(int i = 0; i<size1; i++ )
            {
                for(int j = 0; j<size2; j++)
                {
                    if(arr1[i]==arr2[j])
                    {
                        System.out.println(arr2[j]);
                    }
                }
            }
            break;
        }

    }

}

```

Out-put:-

