Collection Framework

Collection : It is the Single entity or object which can store multiple data.
Framework : represents the Library.
Collection Framework : it is the set of predefined classes and interface which is used to store multiple data.
It contains 2 main parts :
1. Java.util.Collection
2. java.util.Map

- In Collection we can store data directly
-in Map we store data in ky value pair

What is Collection Framework, Collection, Collections

1. Collection Framework : it is an API which Contains predefined class and interface.
2. Collection(interface) : It is the root interface(present in java.util.package) of all the collection objects.
3. Collections(utility class) : it is the utility class which contains only static method .

Syntax of Collection :

```
public interface Collection<E>
extends Iterable<E>{

}
```

#Methods in Collection Interface :

- public boolean add(Object obj) ;
- public boolean addAll(Collection C) ;

ex:

```
public class Methods {
    public static void main(String[] args) {
        ArrayList list1 = new ArrayList() ;
        list1.add(10)           //if this line is written in S.o.p(), then this line will give “true”
        list1.add("Arun") ;
        list1.add('C') ;
        System.out.println(list1);

        ArrayList list2 = new ArrayList() ;
        list2.add("aaa") ;
        list2.add("bbb") ;
        list2.add("ccc") ;
        System.out.println(list2);

        list1.addAll(list2) ;
        System.out.println(list1);
    }
}
```

- boolean contains() ;

- boolean isEmpty() ;
- int size() ;

boolean remove(Object O) ;

eg:

list1.remove(100) ; //100 will be treated as index, not value but if we put any other input like String then it will take as object.

sout(list1) ;

list2.remove(“aaa”) ;

sout(list2) ;

-list1.removeAll(list2) ;

-void clear() ;   //list1.clear() ;

Difference Between Array and Collection

<—

—>

Difference Between list and set
<—

—>

Iterable(interface)   VS
Iterator(interface)

- Iterable is the parent interface of Collection
- All the Collection framework are implementing Iterable interface

- Iterable has only 1 method iterator() which return Iterator Object .

-Iterator interface has 3 method —> hasNext(), next() and remove() ;

```
public Iterable{

            Iterator<> iterator() ;

}

public Interator{
        boolean hasNext() ;
        Object next() ;

}
```

```
public class Methods {
    public static void main(String[] args) {
        List<Integer> list = new
ArrayList<>() ;
        list.add(10) ;
        list.add(20) ;
        list.add(30) ;
        list.add(40) ;
```

```java
        System.out.println(list);

        Iterator itr = list.iterator() ;
        while(itr.hasNext()){
            System.out.println(itr.next());
        }
    }
}
```

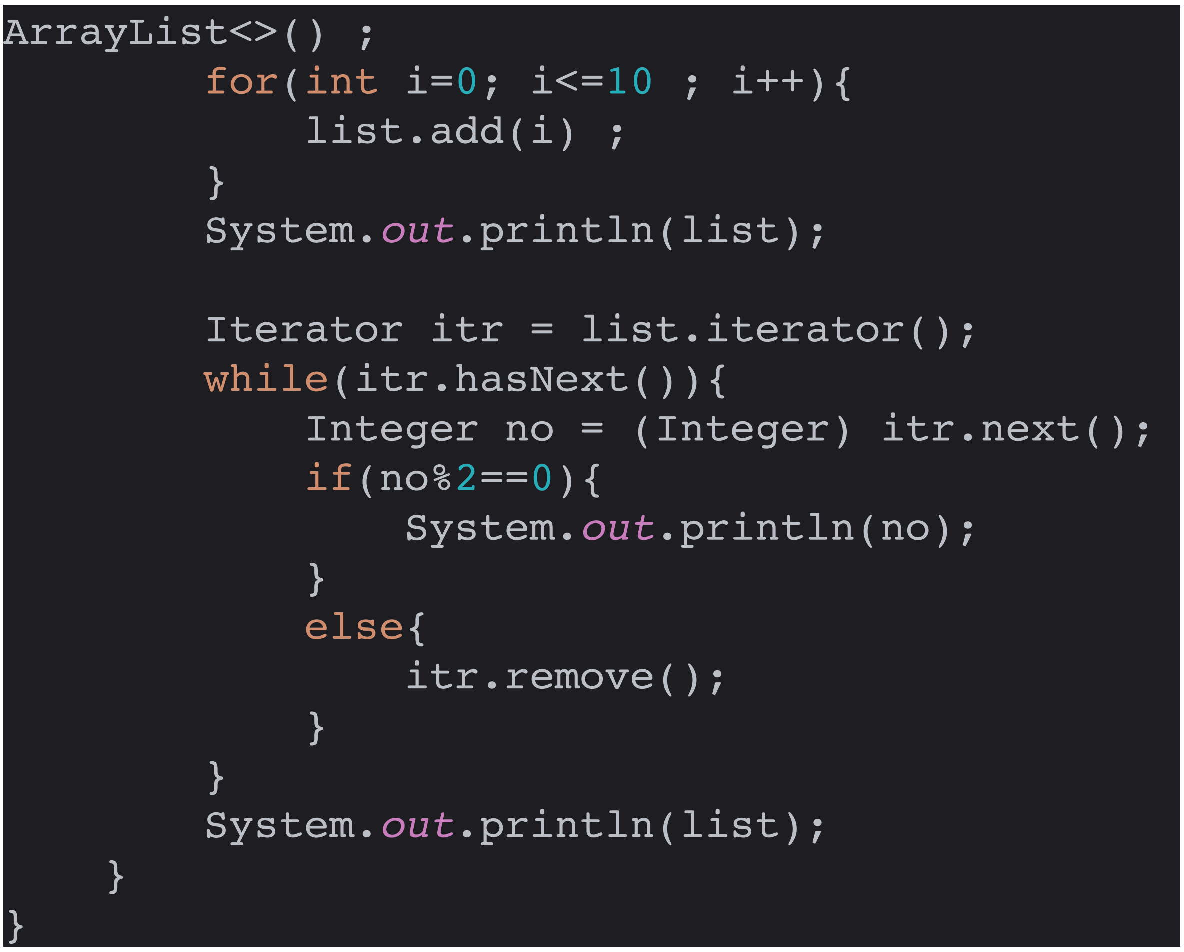
===
==========================

```java
public class Methods {
    public static void main(String[] args) {
        Stack<String> s = new Stack<>() ;
        s.push("abc") ;
        s.push("def") ;
        s.push("ghi") ;

        Iterable<String> iterable = s ;
        for(String str : iterable){
            System.out.println(str);
        }
    }
}
```

===
==========================

```java
public class Methods {
    public static void main(String[] args) {
        ArrayList<Integer> list = new
```



```
ArrayList<>() ;
        for(int i=0; i<=10 ; i++){
            list.add(i) ;
        }
        System.out.println(list);

        Iterator itr = list.iterator();
        while(itr.hasNext()){
            Integer no = (Integer) itr.next();
            if(no%2==0){
                System.out.println(no);
            }
            else{
                itr.remove();
            }
        }
        System.out.println(list);
    }
}
```

Limitation of Iterator :

1. we can only move in forward direction.
2. By using Iterator we can perform only read and remove operation, we can't perform replacement of new object .

ArrayList :

- Inherit list Interface
- present in java.util package

Syntax:

```
 class ArrayList implements List {

          //Constructor
          //methods

}
```

-ArrayList is created on the basis of growable or resizable array.

Properties:
1. Index based data structure.

2. can store different datatypes or heterogeneous data.
3. can store duplicates
4. can store any no. of null values.
5. follows the insertion order.
6. does not follows the sorting order.
7. AL are non Synchronised.

Syntax:

```
List list1= new ArrayList();
              Or
ArrayList list1=new ArrayList();
```

Some of the methods in array list are listed below:
1) add(Object o):
This method adds an object o to the

arraylist.
-obj.add("hello");
This statement would add a string hello in the arraylist at last position.

2) add(int index, Object o): It adds the object o to the array list at the given index.
-obj.add(2, "bye");
It will add the string bye to the 2nd index (3rd position as the array list starts with index 0) of the array list.
3) remove(Object o): Removes the object o from the ArrayList.
-obj.remove("Chaitanya");
This statement will remove the string “Chaitanya” from the ArrayList.

4) remove(int index): Removes

element from a given index.
-obj.remove(3);
-It would remove the element of index 3 (4th element of the list – List starts with o).

5) set(int index, Object o): Used for updating an element.
-It replaces the element present at the specified index with the object o.
-obj.set(2, "Tom");
It would replace the 3rd element (index =2 is 3rd element) with the value Tom.

6) int indexOf(Object o): Gives the index of the object o.
-If the element is not found in the list then this method returns the value -1.

-int pos = obj.indexOf("Tom");
This would give the index (position) of the string Tom in the list.

7) Object get(int index): It returns the object of the list which is present at the specified index.
-String str= obj.get(2);
Function get would return the string stored at 3rd position (index 2) and would be assigned to the string “str”. We have stored the returned value in the string variable because in our example we have defined the ArrayList is of String type. If you are having an integer array list then the returned value should be stored in an integer variable.

8) int size(): It gives the size of the ArrayList – Number of elements of the list.
-int numberOfItems = obj.size();

9) boolean contains(Object o): It checks whether the given object o is present in the array list if it's there then it returns true else it returns false.
-obj.contains("Steve");
It would return true if the string “Steve” is present in the list else we would get false.

10) clear(): It is used for removing all the elements of the array list in one go. The below code will remove all the elements of ArrayList whose

object is obj.
-obj.clear();

==
=========================

Iterate through an ArrayList using various loops and Iterator;

```
import java.util.ArrayList;
import java.util.Iterator;
import java.util.List;
import java.util.Stack;

public class Methods {
    public static void main(String[] args) {
        List<Integer> numbers = new
ArrayList<>();
        numbers.add(5);
        numbers.add(2);
        numbers.add(8);
        numbers.add(1);
        System.out.println("Loop through an
ArrayList using a for loop");
        for(int i=0;i<numbers.size();i++)
        {

System.out.println(numbers.get(i));
        }
        System.out.println("Loop through an
ArrayList using a while loop");
```

```
        int i=0;
        while(numbers.size()>i)
        {

System.out.println(numbers.get(i));
            i++;
        }
        System.out.println("Loop through an 
ArrayList using a "
                + "advanced for loop");
        for(Integer in:numbers)
        {
            System.out.println(in);
        }
        System.out.println("Loop through an 
ArrayList using iterator");
        Iterator it=numbers.iterator();
        while(it.hasNext())
        {
            System.out.println(it.next());
        }
    }
}
```

==
=========================

```
public class Employee {
    private int Id;
    private String name;
    private String address;
    public int getId() {
        return Id;
    }
    public void setId(int id) {
```

```java
        Id = id;
    }
    public String getName() {
        return name;
    }
    public void setName(String name) {
        this.name = name;
    }
    public String getAddress() {
        return address;
    }
    public void setAddress(String address) {
        this.address = address;
    }
    public Employee(int id, String name,
String address) {
        Id = id;
        this.name = name;
        this.address = address;
    }
    @Override
    public String toString() {
        return "Employee [Id=" + Id + ",
name=" + name + ", address=" + address + "]";
    }
}
```

```java
public class Methods {
    public static void main(String[] args) {
        ArrayList<Employee> employees=new
ArrayList<Employee>();
        employees.add(new Employee(1, "Steve",
"Newyork"));
        employees.add(new Employee(2, "john",
"London"));
```

```
        employees.add(new Employee(3, "Tim",
"India"));
        for(Employee e:employees) {
            System.out.println(e);
        }
    }
}
```