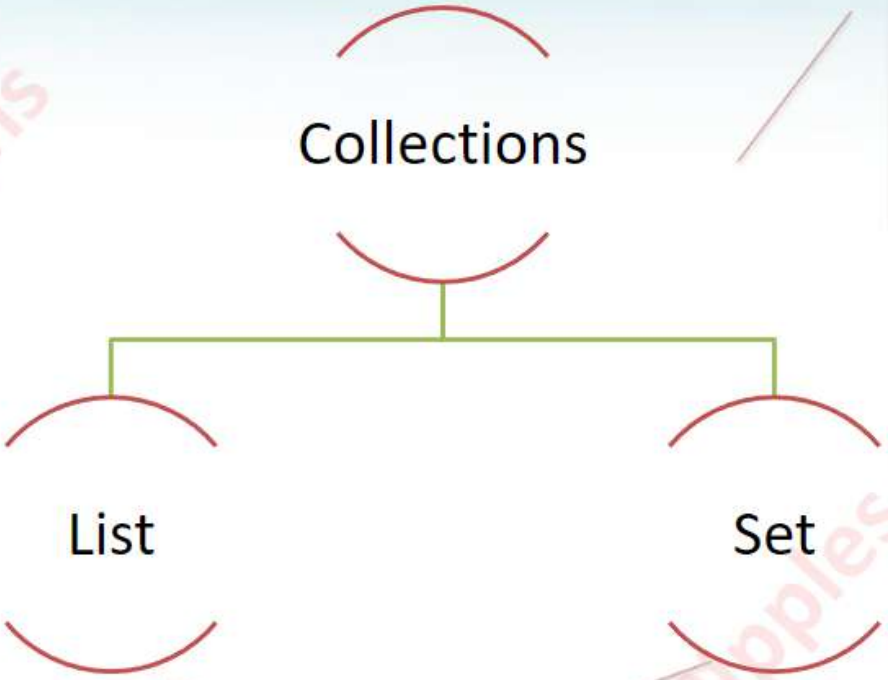
Interface Collection

Click to Continue



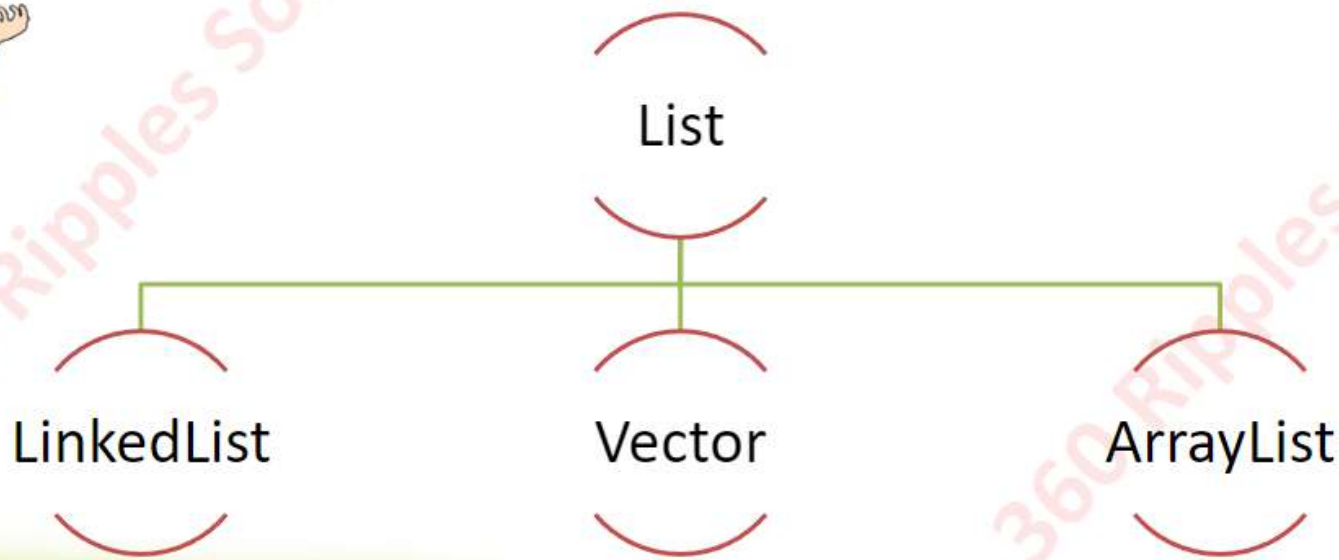
Collection is the root interface of collection framework

List and Set are two interfaces, they extend the root interface Collection.

List Interface

Click to Continue

- List stores elements in a ordered way.
- It can store duplicate entries.
- Allows to add and retrieve elements using index.



Array List

Click to Continue

- This implements the ***List*** interface
- ***ArrayList*** is a variable array it can grow and shrink its size dynamically based on the elements stored.
- Initially it is created with an initial capacity , when full the list automatically grows.
- Can store only reference data types – Cannot store primitive type values.
- Allows duplicate entries.

ArrayList API's

Click to Continue

Method	Description
void add(int index, Object element)	Inserts the specified element at the specified posit this list.
boolean add(Object o)	Appends the specified element to the end of this list.
boolean addAll(Collection)	Appends all of the elements in the specified Collection to the end of this list.
int size()	Returns the number of elements in this list.
void clear()	Removes all of the elements from this list.
boolean contains(Object elem)	Returns true if this list contains the specified element.
Object get(int index)	Returns the element at the specified position in this list.

Creating an Array List

Click to Continue

ArrayList can be created by using one of the below constructors,

1. **ArrayList() :** Constructs an empty list.
2. **ArrayList(**Collection c**) :** Constructs a list from an existing collection, containing the elements of the specified collection.
3. **ArrayList(**int initialCapacity**)** : Constructs an empty list with the specified initial capacity.

Creating an Array List

Click to Continue

ArrayList can be created by using one of the below construc

1. **ArrayList() :** Constructs an empty list.
2. **ArrayList(**Collection c**) :** Constructs a list from an existin elements of the specified collection.
3. **ArrayList(**int initialCapacity**)** : Constructs an empty list with the specified initial capacity.

Let us look at the syntax.

Syntax :

List objectName=new ArrayList();

Illustration :

List primeNumberList =new ArrayList();

Creating an Array List

Click to Continue

ArrayList can be created by using one of the below constru

1. **ArrayList() :** Constructs an empty list.
2. **ArrayList(**Collection c**) :** Constructs a list from an existin elements of the specified collection.
3. **ArrayList(**int initialCapacity**)** : Constructs an empty list with the specified initial capacity.

Let us look at the syntax.

Syntax :

List objectName=new ArrayList();

Illustration :

List primeNumberList =new ArrayList();

Now for the big question:

Why **ArrayList** object is declared with interface ***List***?

Declaration using interface

Click to Continue

In the below code assume that Java deprecates the Array List Class

What will be the impact?

```
public class ArrayListDemo {

    ArrayList primeNumber = new ArrayList();

    public ArrayList getPrimeNumbers() {
        // logic for identifying prime number
        // between 1 and 100
        return primeNumber;
    }

    public void populateOddNumber(ArrayList oddNumber) {
        // logic for add odd numbers
        // in the provided Array Lis oddNumber

    }
}
```

The code implemented in red box will have to change

Let us now reduce the impact

Use interface for declaration

Click to Continue

The solution is to declare the using **List** interface rather than the implementation **ArrayList**. This way we can ensure that even if we change the **ArrayList** to some other implementation (say **DemoList**) the impact would be less..

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

public class ArrayListDemoGood {

    List primeNumber = new ArrayList();

    public List getPrimeNumbers() {
        // logic for identifying prime number
        // between 1 and 100
        return primeNumber;
    }

    public void populateOddNumber(List oddNumber) {
        // logic for add odd numbers
        // in the provided Array Lis oddNumber

    }
}
```

Here the change needs to be done only in one place.

Adding elements in Array List

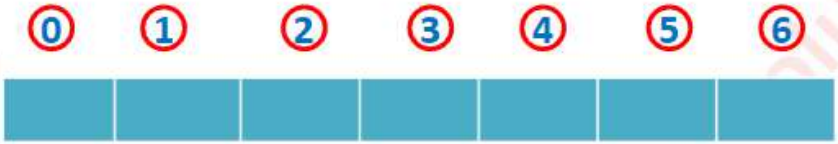
Click to Continue

Syntax :

```
listObject.add(element);
```

Illustration: Let us create a array list and add the colors in it.

```
List myColorList=new ArrayList();
```
Creates a empty arraylist.



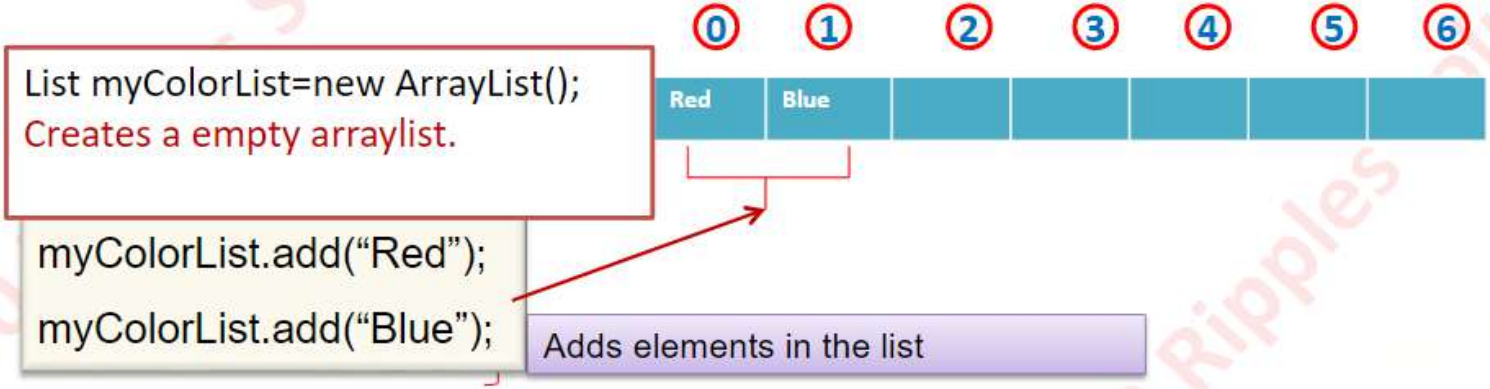
Adding elements in Array List

Click to Continue

Syntax :

listObject.add(element);

Illustration: Let us create a array list and add the colors in it.



Adding elements in Array List

Click to Continue

Syntax :

listObject.add(element);

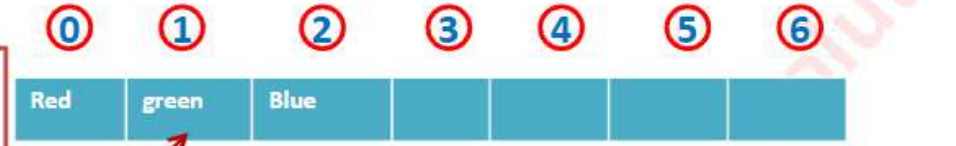
Illustration: Let us create a array list and add the colors in it.

List myColorList=new ArrayList();
Creates a empty arraylist.

myColorList.add("Red");

myColorList.add("Blue");

myColorList.add(1,"green");



Adds the element green at position 1 moving blue to position 2.

Adding elements in Array List

Click to Continue

Syntax :

listObject.add(element);

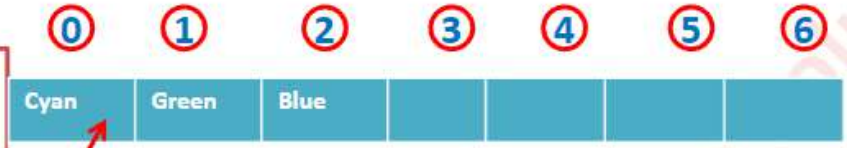
Illustration: Let us create a array list and add the colors in it.

```
List myColorList=new ArrayList();
```
Creates a empty arraylist.

```
myColorList.add("Red");
myColorList.add("Blue");
```

```
myColorList.add(1,"green");
```

```
myColorList.set(0,"Cyan");
```



When set is used the item in the particular index will be replaced by the new item.

Try it out - ArrayList

Click to Continue

We are going to learn about creating and adding element in an array List..

Develop all the below method in a class ***ArrayListDemo***

Scenario 1: Develop a method **loadStudentNames** that accepts the names of three students as three string parameter and add them to an ArrayList.

Scenario 2 : Create an method ***loadEvenNumbers*** which accepts a int N and iterates through 'N' even numbers add each number in the ArrayList and returns the list..

Scenario 3 : Create a method ***copyEvenNumbers*** which needs to copy the even numbers created in scenario # 2 in another List.

Click to Continue

Creates an array list and adds the three student names.

```
public void loadStudentNames(String name1, String name2, String name3) {
    List studentNames = new ArrayList();
    studentNames.add(name1);
    studentNames.add(name2);
    studentNames.add(name3);
    }
```

Always remember to declare the list using ***List*** interface.

Important NOTE:

1. When primitives are added in a collection it gets automatically converted to its equivalent wrapper object.
2. Always use List interface when returning a collection.
3. EvenNumber is a list declared as a class level variable.

```
public List populateEvenNumber(int N) {
    for(int i=0;i<=N;i++)
    {
        if(i%2==0)
        {
            evenNumber.add(i);
        }
    }
    return evenNumber;
}
```

Click to Continue

Important NOTE:

1. In the previous scenario the list was populated . The **evenNumber** should have been a class level variable. This will copy this list into another new list.
2. Before executing this method the populate even number method needs to be invoked and then this method needs to be triggered.

```
public List copyEvenNumbers() {
    List evenNumberCopy = new ArrayList();
    evenNumberCopy.addAll(evenNumber);
    return evenNumberCopy ;
}
```

addAll () method copies the list to another list.

Try it out Solution – Execute Program

Click to Continue

Create an object of the class and execute the three methods

```
public class ArrayListMain {

    /**
     * @param args
     */
    public static void main(String[] args) {

        ArrayListExercise exc = new ArrayListExercise();
        exc.loadStudentNames("Ram", "Shyam", "Won");
        List even = exc.populateEvenNumber(10);
        System.out.println("List" + even);

        List evenCopy = exc.copyEvenNumbers();
        System.out.println("List Copy" + evenCopy);
    }

}
```

If the list variable exc is printed using Println statement the names will be printed.

The populateEvenNumber method should be invoked which populates the evenNumber class variable.

Copy then copies the elements of the variable evenNumber into a new list and returns it. The returned list is stored in a new variable evenCopy and printed.