SYNTAX
TECHNOLOGIES

JAVA

Class 29

Agenda

Wrapper Class in Java
Collection Framework in Java
List Interface

Wrapper classes

Wrapper classes provide a way to use primitive data types (int, boolean, etc..) as objects.

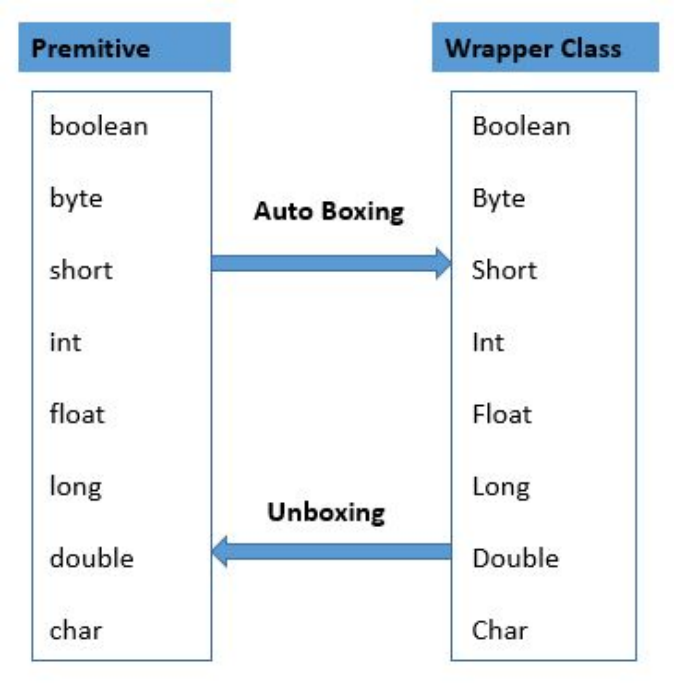
Wrapper class in java provides the mechanism to convert primitive into object and object into primitive.

Sometimes we must use wrapper classes, for example when working with Collection objects, such as ArrayList, where primitive types cannot be used.

Each Java primitive has a corresponding wrapper:

- boolean, byte, short, char, int, long, float, double
- Boolean, Byte, Short, Character, Integer, Long, Float, Double

Wrapper classes



Wrapper classes

“Boxing” refers to converting a primitive value into a corresponding wrapper object.

When a wrapper object is unwrapped into a primitive value then this is known as unboxing.

```
5      public static void main(String[] args) {
6
7          Integer i=new Integer(10);//Boxing
8          Integer int1=10;//AutoBoxing
9          System.out.println(i);
10         System.out.println(int1);
11
12         int i1=i.valueOf(i);//Unboxing
13         int int2=int1;//AutoUnboxing
14
15         System.out.println(i1);
16         System.out.println(int2);
```

Console

<terminated> WrapperClasses [Java Application] /Library/Java/JavaVirtualMachines/jdk1.8.0_144.jdk/Contents/Home/bin/java (Apr 28, 2

```
10
10
10
10
```

Collection Framework

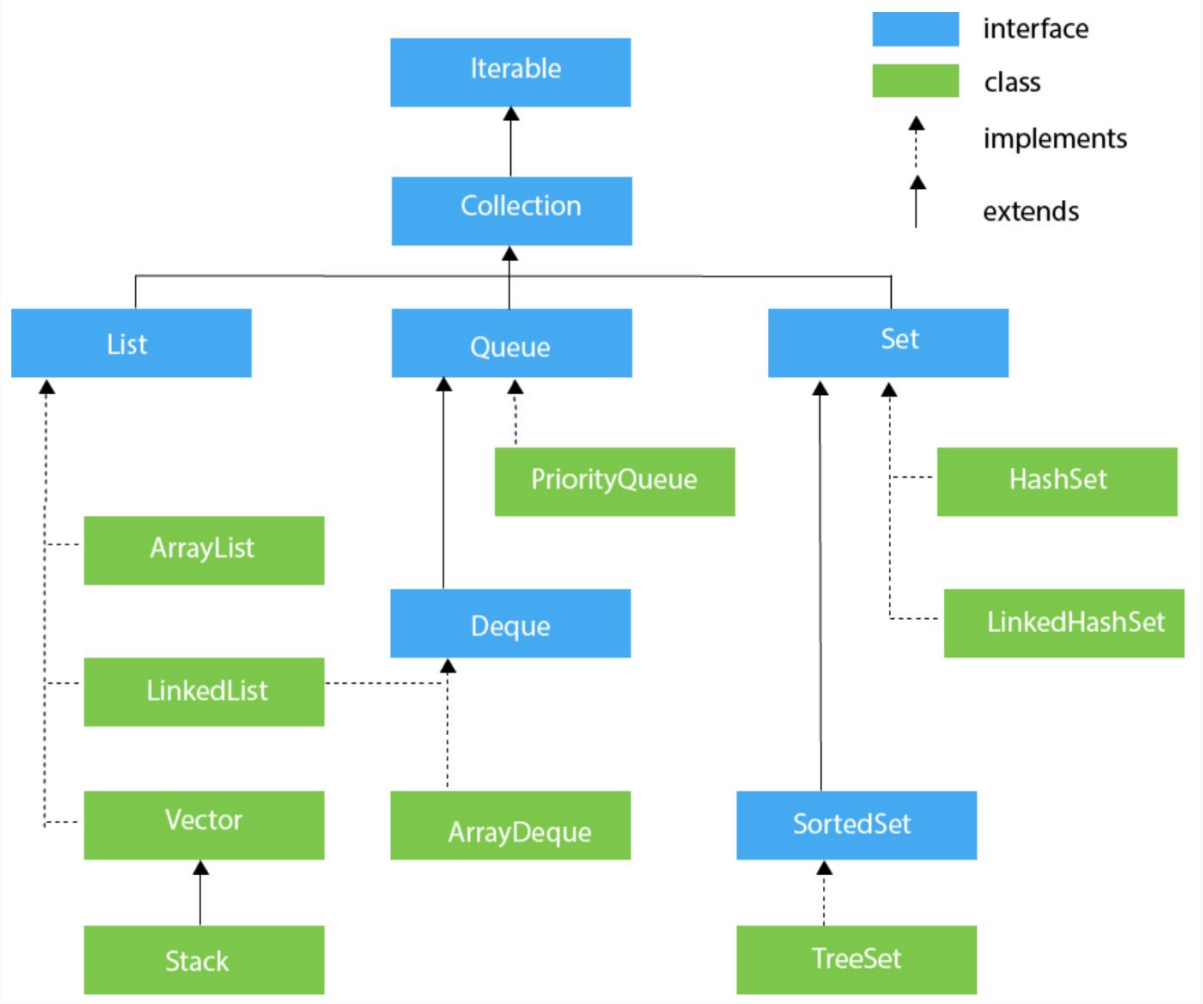
Collection framework in java provides an architecture to store and manipulate the group of objects that represented as a single unit.

This framework has several useful classes which have tons of useful functions which makes a programmer task super easy

Java Collection Framework is a collection of interfaces and classes which help in storing and processing the data efficiently.

Collections are used almost in every programming language. The most frequently used for the test automation: List, Set, Map (map is not a collection but mostly used for collections)

Collection Framework



Commonly used methods of Collection interface:

public boolean add(object element): is used to insert an element in this collection.
public boolean addAll(collection c): is used to insert the specified collection elements in the invoking collection.
public boolean remove(object element): is used to delete an element from this collection.
public boolean removeAll(Collection c): is used to delete all the elements of specified collection from the invoking collection.
public boolean retainAll(Collection c): is used to delete all the elements of invoking collection except the specified collection.
public int size(): return the total number of elements in the collection.
public void clear(): removes the total no of element from the collection.
public boolean contains(object element): is used to search an element.
public boolean containsAll(collection c): is used to search the specified collection in this collection.
public Iterator iterator(): returns an iterator.

Generic vs Non-Generic

Java collection was non-generic before JDK 1.5. Since 1.5, it is generic. Java new generic collection allows you to have only one type of object in the collection. Now it is type safe so typecasting is not required at run time.

- **Non-Generic ArrayList Example**
- ArrayList angl=new ArrayList();
- This is the example of Non-generic ArrayList here we didn't mention the type of collection

- **Generic ArrayList Example**
- ArrayList<String> agl=new ArrayList<>();
- This is the example of Generic ArrayList.
- Here have declared the Collection type (String) at the time of Initialization.

List

A List is an ordered Collection (sometimes called a sequence).

Lists may contain duplicate elements.

Elements can be inserted or accessed by their position in the list, using a zero-based index.

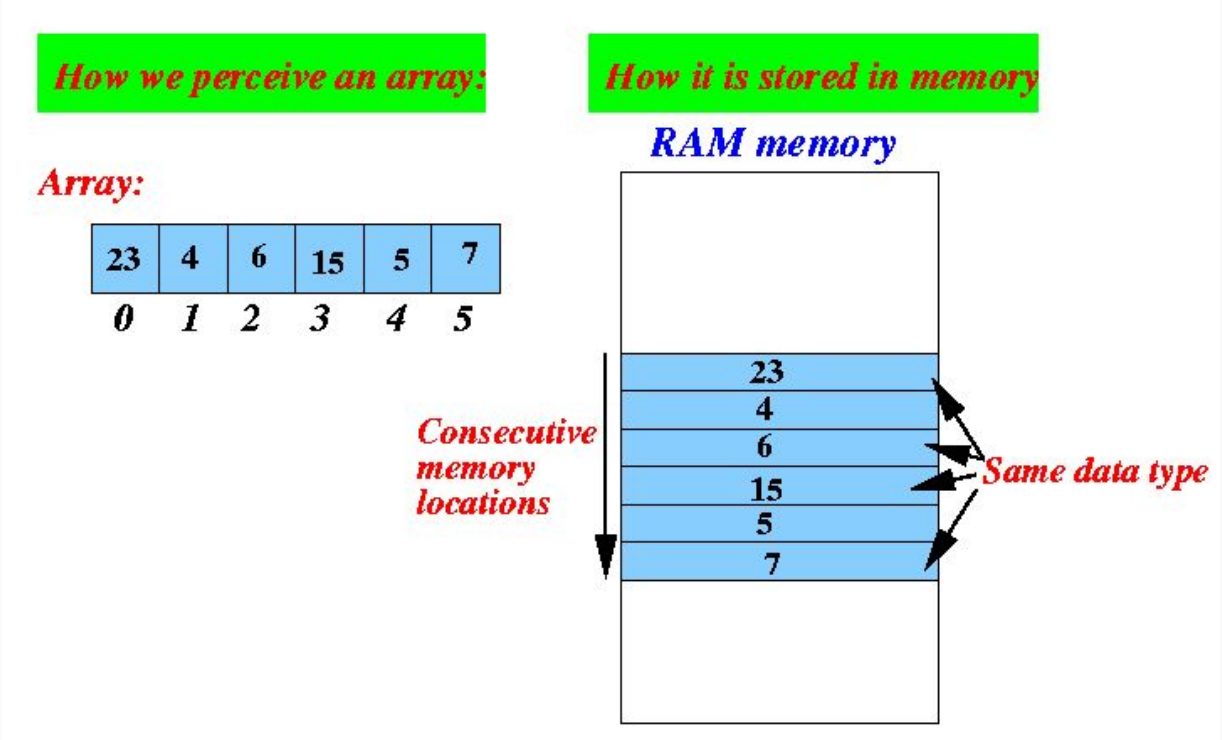
List is an order collection that can contain duplicate elements

List is one of the most used Collection type.

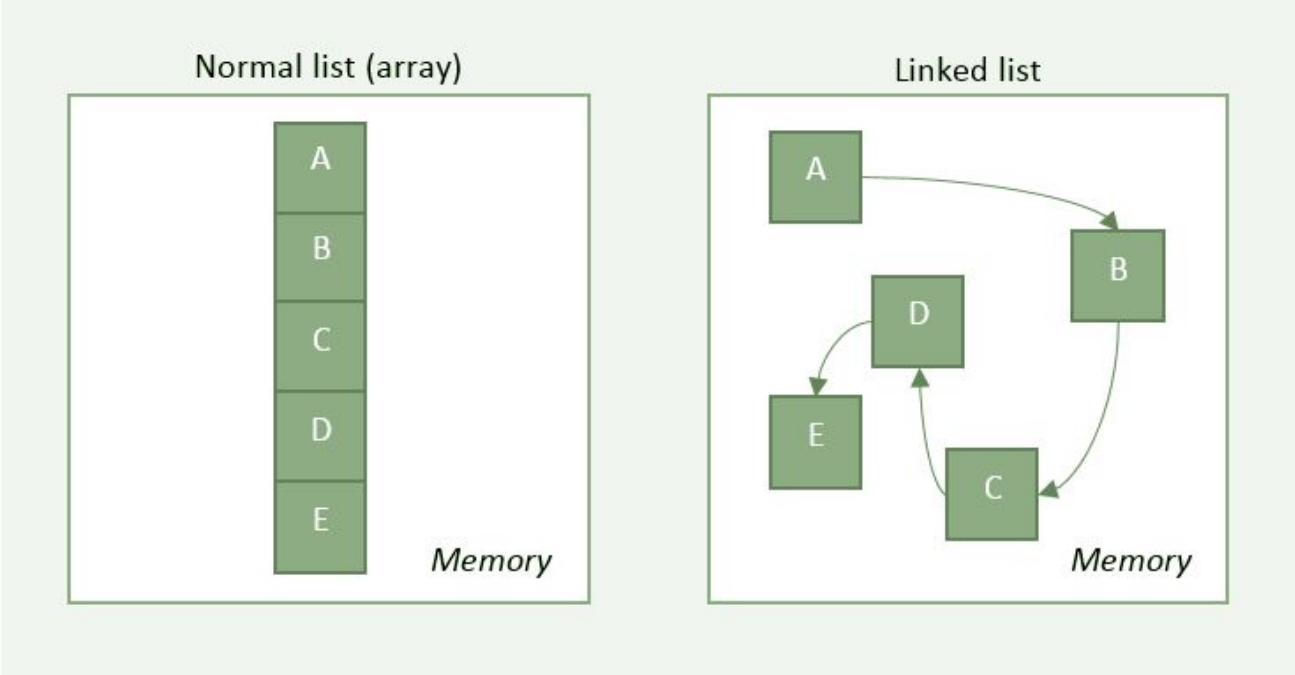
Classes that implement List interface:

- ArrayList
- LinkedList
- Vector

How Arrays Are Stored in Memory



How Linked List is stored in memory



ArrayList

ArrayList is a class which implements the List interface of collection framework.

ArrayList is dynamic data structure i.e like Arrays you don’t need to define the size of ArrayList during the declaration.

You can add and remove the elements from ArrayList and ArrayList adjust its size automatically.

ArrayList can contain the duplicate elements.

It implements all optional list operations, and permits all elements, including null.

Methods of ArrayList

add(Object o)	adds an object to the arraylist
add(int index, Object o)	adds the object to the array list at the specified index
remove(Object o)	removes the object from the ArrayList
remove(int index)	removes element from a given index
set(int index, Object o)	used for updating an element. It replaces the element present at the specified index with the object
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.
Object get(int index)	returns the object of list which is present at the specified index
int size()	gives the size of the ArrayList – Number of elements of the list
boolean contains(Object o)	checks whether the given object is present in the array list if its there then it returns true else it returns false
clear()	used for removing all the elements of the array list in one go. The below code will remove all the elements of ArrayList

ArrayList

```
public class ArrayListDemo {
    public static void main(String[] args) {

   // Create new ArrayList
    ArrayList<Integer> elements = new ArrayList<Integer>();
    elements.add(10); // Add three elements.
    elements.add(15);
    elements.add(20);

   SOP("Elements of the ArrayList are -- " + elements); // Print ArrayList

    int count = elements.size(); // Get size and display.

    SOP("Size of ArrayList after Element addition -- " + count);

    elements.remove(2); // Remove elements are using Index Number

    }
}
```

How to loop ArrayList in Java

```
public class LoopExample {
   public static void main(String[] args) {

      ArrayList<Integer> arrlist = new
ArrayList<Integer>();
      arrlist.add(14);
      arrlist.add(7);
      arrlist.add(39);
      arrlist.add(40);

      /* For Loop for iterating ArrayList */
      SOP("For Loop");
      for (int counter = 0; counter < arrlist.size();
counter++) {
         System.out.println(arrlist.get(counter));

      }
```

```
/* Advanced For Loop*/
      SOP("Advanced For Loop");

for (Integer num : arrlist) {
         System.out.println(num);
      }

      /*Looping Array List using Iterator*/
      SOP("Iterator");
      Iterator iter = arrlist.iterator();
      while (iter.hasNext()) {
        System.out.println(iter.next());
      }
   }
}
```

ITERATOR

The iterator is used for iterating the classes in Collection framework.

We use an iterator to iterate the elements of the collection classes.

The iterator is an interface.

You can iterate only in one direction.

Note: Iteration can be done only once. If you reach the end of series it's done. If we need to iterate again we should get a new Iterator.

ITERATOR

There are only three methods in the Iterator interface. They are:

boolean hasNext() – It returns true if iterator has more elements.

Object next() – It returns the element and moves the cursor pointer to the next element.

default void remove() – It removes the last elements returned by the iterator.