



Java Collection

Press Space for next page →



Agenda

1. Important methods of Collection Interface
2. More methods of Collection Interface
3. ArrayList
4. ArrayList Code Example
5. LinkedList
6. LinkedList Code Example
7. HashSet
8. HashSet Code Example
9. Java Cursors
10. Enumeration
11. Iterator
12. ListIterator

Important methods of Collection Interface

METHOD	DESCRIPTION
<code>boolean add(Object o)</code>	Adds the specified element to the collection.
<code>boolean contains(Object o)</code>	Checks if the collection contains the specified element.
<code>boolean addAll(Collection c)</code>	Adds all elements from the specified collection to this one.
<code>boolean containsAll(Collection c)</code>	Checks if the collection contains all elements from another collection.
<code>boolean remove(Object o)</code>	Removes the specified element from the collection.

More methods of Collection Interface

METHOD	DESCRIPTION
<code>boolean isEmpty()</code>	Checks if the collection is empty.
<code>boolean removeAll(Collection c)</code>	Removes all elements from the collection that are contained in another collection.
<code>int size()</code>	Returns the number of elements in the collection.
<code>boolean retainAll(Collection c)</code>	Retains only the elements in this collection that are contained in another collection.
<code>Object[] toArray()</code>	Returns an array containing all of the elements in the collection.
<code>void clear()</code>	Removes all elements from the collection.

ArrayList

- The underlined data structure Resizable Array or Growable Array
- Duplicates are allowed.
- Insertion order is preserved.
- Heterogeneous objects are allowed except TreeSet & TreeMap everywhere heterogeneous objects are allowed.
- Null insertion is possible.

ArrayList Constructors

1. `ArrayList al = new ArrayList();`
2. `ArrayList al = new ArrayList(int initialCapacity);`
3. `ArrayList al = new ArrayList(Collection c);`

ArrayList Code Example

```
1  import java.util.ArrayList;
2
3  public class ArrayListExample {
4      public static void main(String[] args) {
5          ArrayList<String> programmingLanguages = new ArrayList<>();
6          programmingLanguages.add("Java");
7          programmingLanguages.add("Python");
8          programmingLanguages.add("JavaScript");
9          programmingLanguages.add("C#");
10         programmingLanguages.add("Ruby");
11
12         System.out.println(programmingLanguages);
13
14         ArrayList<Object> mixedList = new ArrayList<>(programmingLanguages);
15         mixedList.add("Java");
16         mixedList.add(10);
17         mixedList.add(10.5);
18         mixedList.add(true);
19         mixedList.add(null);
20
21         mixedList.remove(10.5);
22
23         System.out.println(mixedList);
24     }
25 }
```

LinkedList

- The underlying data structure is Double Linked List.
- Insertion order is preserved.
- Duplicates are allowed.
- Heterogeneous Objects are allowed.
- Null insertion is possible.
- Linked List is the best choice if our requirement is **insertion and deletion in the middle**.
- Worst choice if our frequent operation is **retrieval of data**.

ArrayList Constructors

1. `LinkedList l1 = new LinkedList();`
2. `LinkedList l2 = new LinkedList(Collection c);`

LinkedList Code Example

```
1  import java.util.LinkedList;
2
3  public class LinkedListExample {
4      public static void main(String[] args) {
5          LinkedList<String> programmingLanguages = new LinkedList<>();
6          programmingLanguages.add("Java");
7          programmingLanguages.add("Python");
8          programmingLanguages.add("JavaScript");
9          programmingLanguages.add("C#");
10         programmingLanguages.add("Ruby");
11
12         System.out.println(programmingLanguages);
13
14         LinkedList<Object> mixedList = new LinkedList<>(programmingLanguages);
15         mixedList.add("Java");
16         mixedList.add(10);
17         mixedList.add(10.5);
18         mixedList.add(true);
19         mixedList.add(null);
20
21         mixedList.remove(10.5);
22
23         System.out.println(mixedList.getFirst());
24         System.out.println(mixedList.getLast());
25         System.out.println(mixedList.get(2));
26         System.out.println(mixedList);
```



```
27     }  
28 }
```

HashSet

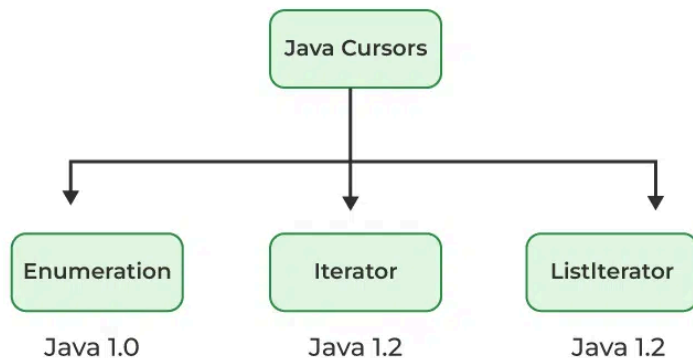
- The underlying data structure is Hashtable.
- **Duplicates are not allowed**. If we are trying to insert duplicates, we won't get any compile time or runtime errors. add() method simply returns false.
- **Insertion order is not preserved** and all objects will be inserted based on hash-code of objects.
- Heterogeneous objects are allowed.
- Null insertion is possible.
- Implements Serializable and Clonable interfaces but not RandomAccess.
- HashSet is the best choice, if our frequent operation is Search operation.

HashSet Code Example

```
1  import java.util.HashSet;
2  public class HashSetExample {
3      public static void main(String[] args) {
4
5          HashSet<Object> hashSet = new HashSet<>(); // Create a HashSet
6
7          hashSet.add("Java");           // String
8          hashSet.add(42);               // Integer
9          hashSet.add(3.14);             // Double
10         hashSet.add(null);              // Null value
11         hashSet.add("HashSet");         // String
12
13         boolean isAdded = hashSet.add("Java"); // Duplicates not allowed
14         System.out.println("Was 'Java' added again? " + isAdded); // Output: false
15
16         System.out.println("HashSet elements: " + hashSet);
17
18         boolean containsElement = hashSet.contains(42);
19         System.out.println("Does HashSet contain '42'? " + containsElement); // Output: true
20
21         hashSet.remove(3.14);
22         System.out.println("After removing 3.14: " + hashSet);
23
24         System.out.println("Does HashSet contain 'HashSet'? " + hashSet.contains("HashSet")); // Output: true
25     }
26 }
```

Java Cursors

in Java, cursors are a set of interfaces and classes that allow you to traverse or iterate through the elements of a collection. These cursors provide a way to access and manipulate the elements of a data structure, such as lists, sets, and maps, one at a time.



Java Cursors

Enumeration

Available methods:

- `boolean hasMoreElements()` `Object nextElement()`

Create an Enumeration object by calling the `elements()` method of the `Vector` class.

```
1  import java.util.Enumeration;
2  import java.util.Vector;
3
4  public class EnumerationExample {
5      public static void main(String[] args) {
6          Vector<String> vector = new Vector<>();
7          vector.add("Java");
8          vector.add("Python");
9          vector.add("JavaScript");
10         vector.add("C#");
11         vector.add("Ruby");
12
13         Enumeration<String> enumeration = vector.elements();
14         while (enumeration.hasMoreElements()) {
15             System.out.println(enumeration.nextElement());
16         }
17     }
18 }
```


Available methods:

```
1  import java.util.ArrayList;
2  import java.util.Iterator;
3
4  public class IteratorExample {
5      public static void main(String[] args) {
6          ArrayList<String> programmingLanguages = new ArrayList<>();
7          programmingLanguages.add("Java");
8          programmingLanguages.add("Python");
9          programmingLanguages.add("JavaScript");
10         programmingLanguages.add("C#");
11         programmingLanguages.add("Ruby");
12
13         Iterator<String> iterator = programmingLanguages.iterator();
14         while (iterator.hasNext()) {
15             // if python remove it else print all the available programming languages
16             String language = iterator.next();
17             if (language == "Python" ) {
18                 iterator.remove();
19             } else {
20                 System.out.println(language);
21             }
22         }
23     }
```

Available methods:

```
1  import java.util.LinkedList;
2  import java.util.ListIterator;
3
4  public class ListIteratorExample {
5      public static void main(String[] args) {
6          LinkedList<String> programmingLanguages = new LinkedList<>();
7          programmingLanguages.add("Java");
8          programmingLanguages.add("Python");
9          programmingLanguages.add("JavaScript");
10         programmingLanguages.add("C#");
11         programmingLanguages.add("Ruby");
12
13         ListIterator<String> listIterator = programmingLanguages.listIterator();
14         while (listIterator.hasNext()) {
15             String language = listIterator.next();
16             if (language == "Python") {
17                 listIterator.remove();
18             } else {
19                 System.out.println(language);
20             }
21         }
```

Thank you 

 [qa-june-2024-automation-with-java-slides](#)