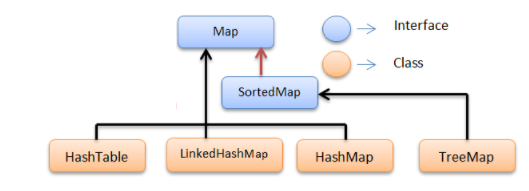
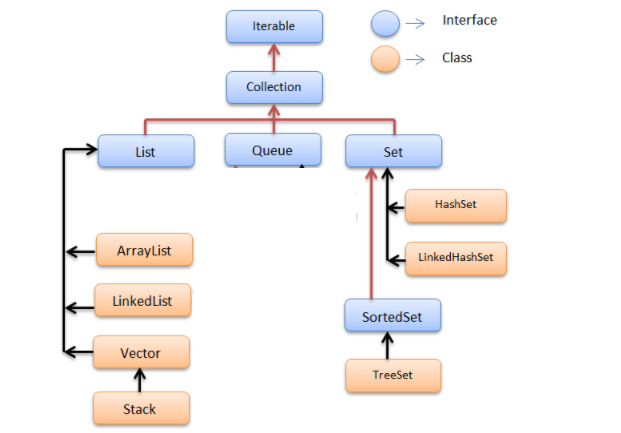
**COLLECTION**

* Collection represents a single unit of objects.
* It is an Interface.
* It Supports **Dissimilar** datatype.
* It Has Low Memory Wastage.
* It doesn't have any Fixed Length.

**Note: To overcome array we are using the collection.**

Types Of Collection:

* List [Interface]
* Set [Interface]
* Map [Interface]

concurrentHashMap

I RIA RIRAR

**List : [INSERTION ORDER]**

* It is an Interface.
* It is a index based.
* it does allow duplicate values.
* It Prints in insertion order.

**ORDER 🡪 Insertion**

**Syntax**:

**List <WrapperClass> refName = new (types of list) <WrapperClass> ();**

where

WrapperClass --> Its convert datatype into class object.

List ----> Interface

Types of List -----> Class

< > ----> Generics -----> For Type Safety.

**Array List ( Class )**

Syntax :

List <WrapperClass> refName = new ArrayList<WrapperClass> ();

Def:

* Searching & Retrieving is very easy
* Insertion & Deletion is difficult
* Asynchronized
* Thread is not safe but very fast.

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

**Linked List : ( Class )**

Syntax :

List <WrapperClass> refName = new LinkedList<WrapperClass> ();

Def:

* Insertion And Deletion is Easy
* Searching And Retrieving is difficult.
* Asynchronized
* Thread is not Safe.... But fast Process

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

**Vector List ( Class )**

Synatx :

List <WrapperClass> refName = new Vector<WrapperClass> ();

Definition :

Synchronized

Thread Safe ... But it is a slow process

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

List Methods :

|  |  |  |
| --- | --- | --- |
| add() | indexOf() | removeAll() |
| size() | contains() |  |
| get(3) | clear() |  |
| set() | addall() |  |
| remove() | retainAll() |  |

List<Integer> numbers = Arra0079s.asList(1, 2, 3, “ishu”, 5, 6, 7, 8);

OR

List<Integer> al = new ArrayList<Integer>();

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

Set ( Interface ) :

* It is a Interface.
* It does not have any specific order.
* It does not allow Duplicate values.
* It is a value based.

**ORDER 🡪 Random Insertion Ascending**

Syntax : Set <WrapperClass> refName = new TypesOfSet <WrapperClass> ();

Types :

HashSet - R

LinkedHashSet - I

TreeSet - A

HashSet :

It Prints in Random Order.

It will not allow Duplicate Values.

It will allow Single null value . not a duplicate null values.

LinkedHashSet :

It Prints in Insertion Order.

It will not allow Duplicate Values.

It will allow Single null value . not a duplicate null values.

TreeSet :

It Prints in Ascending Order.

It will not allow Duplicate Values.

It will not allow even Single null value .

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

Map [ Interface ]

It is A Key And Value Pair.

Key + Value = One Entry

Key ---->

it will **not allow duplicate** it can be override.

it will allow override null [ duplicate]

value ----->

The value will allow duplicate

The value will allow the both null.

**ORDER 🡪 Random Insertion Random Ascending Random**

Syntax :

Map<Key,Value> refName = new TypesOfMap<Key,Value > ();

eg :

Map<Integer,String> m = new HashMap<Integer, String > ();

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

HashTable:

* contains unique elements.
* doesn't allow null key or value

**HashMap**

* It contains values based on the key.
* It contains only **unique keys**.
* It may have **one null key** and **multiple null values**.
* It maintains Random Order.

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

Map Methods :

put() size() keyset() values() containsKey() containsValue() entrySet()

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

Parent Class of all class in JAVA🡪 Object

Parent Class of all Interface in JAVA🡪 Iterable

Parent Class of all Interface in Selenium🡪 Search Context

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*