**A Java Collection Framework Summary**

A Collection is a group of objects contained in a single object.

The Collection framework is a Set of Classes for storing collections and it’s made of four main interfaces:

**List:** an ordered collection of elements that allows duplicates, that can be accessed by an *int* index.

**Set:** a collection that does not allow duplicate entries.

**Queue:** a collection that orders its elements in a specific order for processing.

**Map:** a collection that maps keys to values, with no duplicates keys allowed.

The Collection framework is the root of all collections except maps. Map doesn’t implement the Collection interface but it’s considered part of the Collections Framework because it contains group of objects.



**The List Interface:**

Is an ordered collection that can contain duplicate entries. Items can be inserted and retrieved at specific positions in the list based on an *int* index

**List Implementations:**

**ArrayList:** an ArrayList is like a resizable array. When elements are added, the ArrayList automatically grows.

The main benefit of using this implementations is that an element can be looked up in constant time. Adding or removing an element is slower than accessing an element.

This makes an ArrayList is a good choice when more readings than writings are needed.

**LinkedList:** It is a special implementation because it implements both List and Queue interfaces.

The main benefit of using it is that elements can be accessed, added and removed from the beginning and the end of the list in constant time. The tradeoff is that dealing with an arbitrary index takes linear time.

This makes a LinkedList is a good choice when is used as a Queue.

**Vector:** Is an old implementation that was replaced by ArrayList. Vector does the same things as an ArrayList but slower, but as a benefit is thread – safe.

**Stack:** Is an old implementation of a data structure where elements can be added and removed from the top of the stack.

**The Set Interface:**

A collection that does not allow duplicate entries.

**Set Implementations:**

**HashSet:** A collection that stores its elements in a Hash Table. This means that it uses the *hashCode()* method of the objects to retrieve them more efficiently.

The main benefit is that adding elements and checking if an element is in the set both constant time.

The tradeoff is that the order in which elements are inserted is lost.

**TreeSet:** A collection that inserts its elements in a sorted tree structure.

The main benefit is that the structure is always ordered.

The tradeoff is that adding and checking if an element is present are both

**Queue Interface:**

A collection that is used when elements are added and removed in a specific order. Is assumed to be FIFO (First-In, First-Out) but it can change depending the implementation.

**Queue Implementations:**

**LinkedList:** A double-ended queue that allows inserting and removing elements from both front and back of the structure.

**ArrayDeque:** It stores its elements in a resizable array and its more efficient than a LinkedList.

The main benefit of this implementation is that is double-ended and it can be used as a LIFO or FIFO structure depending on what is needed.

**Map Interface:**

An interface that identify values by a key.

**Map Implementations:**

**HashMap:** Stores the keys in a Hash Table. This means that is used *hashCode()* method of the keys to retrieve their values more efficiently.

The main benefit is that adding and retrieving elements by key happens in constant time.

The tradeoff is that the order in which elements are inserted is lost.

**LinkedHashMap:** Is used when the order in which elements are inserted is important.

**TreeMap:** Stores the keys in a sorted tree structure. The main benefit is that the keys are always sorted.

The tradeoff is that adding and checking if a key is present happens in O (log n) time.

**HashTable:** Is like Vector for the List interface. Is very old and it’s also thread – safe.

The table summarizes the principle classes in Java Collections:

**D**: Duplicate elements is allowed?

**O**: Elements are ordered?

**S**: Elements are sorted?

**TS**: The collection is thread – safe?

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Collection Classes** | **D** | **O** | **S** | **TS** |
| **ArrayList** | Yes | Yes | No | No |
| **LinkedList** | Yes | Yes | No | No |
| **Vector** | Yes | Yes | No | Yes |
| **HashSet** | No | No | No | No |
| **LinkedHashSet** | No | Yes | No | No |
| **TreeSet** | No | Yes | Yes | No |
| **HashMap** | No | No | No | No |
| **LinkedHashMap** | No | Yes | No | No |
| **Hashtable** | No | No | No | Yes |
| **TreeMap** | No | Yes | Yes | No |