Java Collection Framework

Java collection framework represents a hierarchy of set of interfaces and classes that are used to manipulate group of objects.

Collections framework was added to Java 1.2 version. Prior to Java 2, Java provided adhoc classes such as Dictionary, Vector, Stack and Properties to store and manipulate groups of objects.

Java collections framework is contained in java.util package. It provides many important classes and interfaces to collect and organize group of objects.

What is collection

Collection in java can be referred to an object that collects multiple elements into a single unit. It is used to store, fetch and manipulate data. For example, list is used to collect elements and referred by a list object.

Components of Collection Framework

The collections framework consists of:

- Collection interfaces such as sets, lists, and maps. These are used to collect different types of objects.
- Collection classes such as ArrayList, HashSet etc that are implementations of collection interfaces.
- Concurrent implementation classes that are designed for highly concurrent use.
- Algorithms that provides static methods to perform useful functions on collections, such as sorting a list.

Advantage of Collection Framework

- It reduces programming effort by providing built-in set of data structures and algorithms.
- Increases performance by providing high-performance implementations of data structures and algorithms.
- Provides interoperability between unrelated APIs by establishing a common language to pass collections back and forth.
- Increase Productivity
- Reduce operational time
- versatility to work with current collection as well

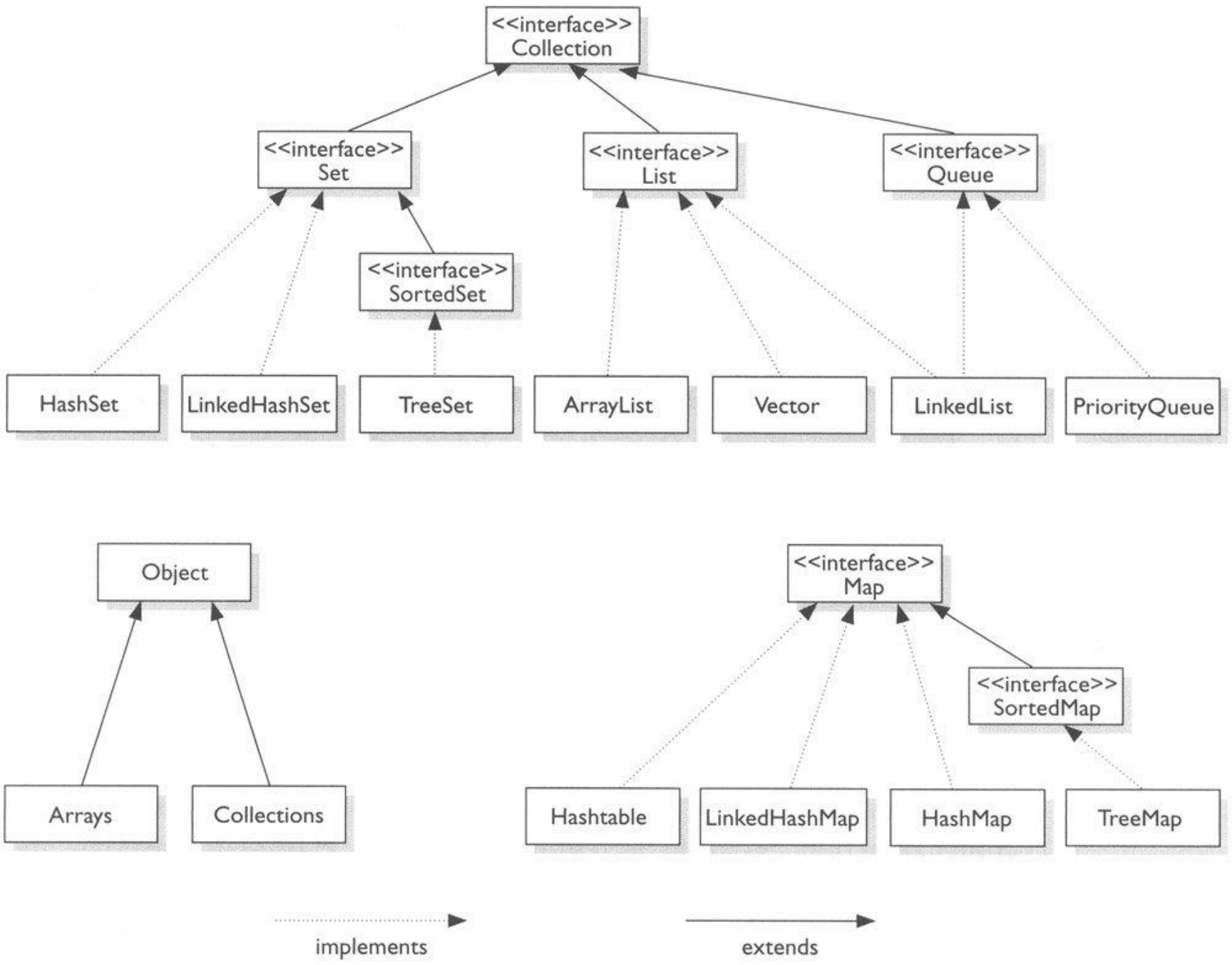
Important Interfaces of Collection API

Interface	Description
Collection	Enables you to work with groups of object; it is at the top of Collection hierarchy
Deque	Extends Queue to handle double ended queue.
List	Extends Collection to handle sequences list of object.

Queue	Extends Collection to handle special kind of list in which element are removed only from the head.
Set	Extends Collection to handle sets, which must contain unique element.
SortedSet	Extends Set to handle sorted set.

Java Collection Hierarchy

Collection Framework hierarchy is represented by using a diagram. You can get idea how interfaces and classes are linked with each other and in what hierarchy they are situated. Top of the framework, Collection framework is available and rest of interfaces are sub interface of it.



Collection Heirarchy

All these Interfaces give several methods which are defined by collections classes which implement these interfaces.

Commonly Used Methods of Collection Framework

Method	Description

public boolean add(E e)	It inserts an element in this collection.
public boolean addAll(Collection<? extends E> c)	It inserts the specified collection elements in the invoking collection.
public boolean remove(Object element)	It deletes an element from the collection.
public boolean removeAll(Collection<?> c)	It deletes all the elements of the specified collection from the invoking collection.
default boolean removeIf(Predicate<? super E> filter)	It deletes all the elements of the collection that satisfy the specified predicate.
public boolean retainAll(Collection<?> c)	It deletes all the elements of invoking collection except the specified collection.
public int size()	It returns the total number of elements in the collection.
public void clear()	It removes the total number of elements from the collection.
public boolean contains(Object element)	It searches for an element.
public boolean containsAll(Collection<?> c)	It is used to search the specified collection in the collection.

Why Collections were made Generic?

Generics added **type safety** to Collection framework. Earlier collections stored **Object class** references which meant any collection could store any type of object. Hence there were chances of storing incompatible types in a collection, which could result in run time mismatch. Hence Generics was introduced through which you can explicitly state the type of object being stored.

Collections and Autoboxing

We have studied that Autoboxing converts primitive types into Wrapper class Objects. As collections doesn't store primitive data types(stores only refrences), hence Autoboxing facilitates the storing of primitive data types in collection by boxing it into its wrapper type.

Most Commonly thrown Exceptions in Collections Framework

There may be chance of getting exceptions while working with collections. We have listed some most common exceptions that may occur during program execution.

Exception Name	Description
UnSupportedOperationException	occurs if a Collection cannot be modified
ClassCastException	occurs when one object is incompatible with another
NullPointerException	occurs when you try to store null object in Collection
IllegalArgumentException	thrown if an invalid argument is used
IllegalStateException	thrown if you try to add an element to an already full Collection