HEXAWARE

Java - Collection

Session Objective

- Collection Framework
- Collection Hierarchy
- Raw Collection
- Scanning of objects in Collection Framework
- Basics of Generics
- Advance concept in Generics

Collection Framework

- Prior to Java 2, Java provided ad hoc classes such as **Dictionary, Vector, Stack**, and **Properties** to store and manipulate groups of objects. Although these classes were quite useful for implementing data structure, they lacked a central, unifying theme. But the way that you used Vector was different from the way that you used Properties.

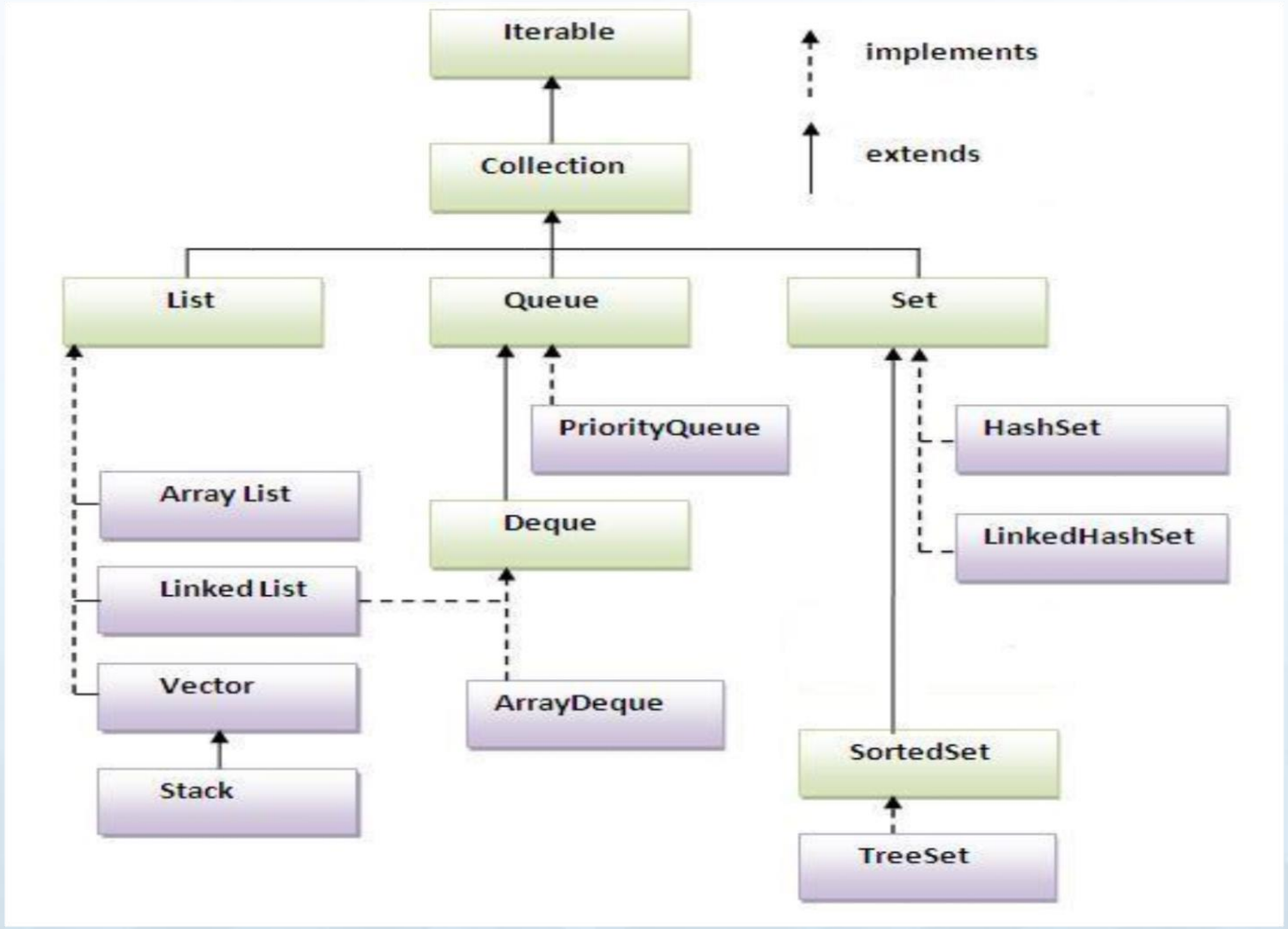
Collection Framework cont.…

- The collections framework was designed to meet several goals.
- The framework had to be high-performance.
- The framework allow different types of collections to work in a similar manner and with a high degree of interoperability.
- Extending and/or adapting a collection had to be easy.

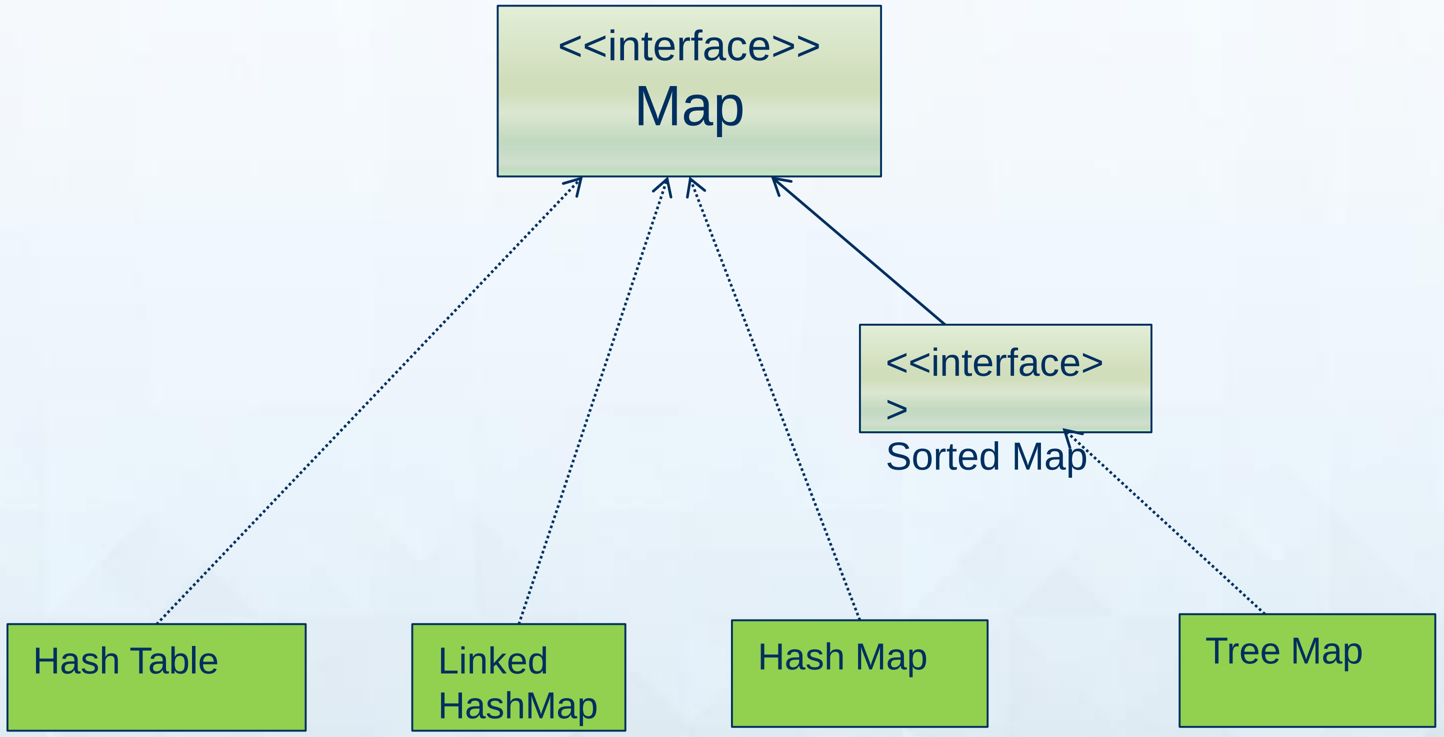
Collections in java

- Collection represents a single unit of objects i.e. a group.
- provides readymade architecture.
- represents set of classes and interface.
- Collection framework represents a unified architecture for storing and manipulating group of object. It has:
- Interfaces and its implementations i.e. classes and Algorithm
- The **java.util** package contains all the classes and interfaces for Collection framework.

Collection Hierarchy



Collection Hierarchy cont...



List

Property of the List Interface:

The List interface extends Collection to define an ordered collection with duplicates allowed. The List interface adds position-oriented operations, as well as a new list iterator that enables the user to traverse the list bi-directionally.

Under List we have classes like

- ArrayList
- LinkedList
- Vector
- Stack

Set

Property of the Set Interface:

The Set interface extends the Collection interface. It will make sure that an instance of Set contains no duplicate elements. The concrete class implements hashcode and equals methods to make sure uniqueness of objects.

Under Set we have concrete classes like

- HashSet
- LinkedHashSet
- TreeSet

Queues:

Property of the Queue Interface:

A queue is a first-in, first-out data structure. Elements are appended to the end of the queue and are removed from the beginning of the queue. In a priority queue, elements are assigned priorities. When accessing elements, the element with the highest priority is removed first

Under Queue we have concrete classes like

- Priority Queue
- Array Dequeue

Priority Queue

S.N.	Method & Description
1	boolean add(E e) This method inserts the specified element into this priority queue.
2	void clear() This method removes all of the elements from this priority queue.
3	Comparator<? super E> comparator() This method returns the comparator used to order the elements in this queue, or null if this queue is sorted according to the natural ordering of its elements.
4	boolean contains(Object o) This method returns true if this queue contains the specified element.
5	Iterator<E> iterator() This method returns an iterator over the elements in this queue.
6	boolean offer(E e) This method inserts the specified element into this priority queue.
7	E peek() This method retrieves, but does not remove, the head of this queue, or returns null if this queue is empty.
8	E poll() This method retrieves and removes the head of this queue, or returns null if this queue is empty.
9	boolean remove(Object o) This method removes a single instance of the specified element from this queue, if it is present.
10	int size() This method returns the number of elements in this collection.
11	Object[] toArray() This method returns an array containing all of the elements in this queue.
12	<T> T[] toArray(T[] a) This method returns an array containing all of the elements in this queue; the runtime type of the returned array is that of the specified array.

Dequeue (Array Dequeue)

- A collection that can be used as a stack or a queue Means “double-ended queue” (and is pronounced “deck”)
- A queue provides FIFO operations-add and remove()methods
- A stack provides LIFO (last in, first out) operations-push and pop() methods Deque is a child interface of Collection
- A queue is often used to track asynchronous message requests so they can be processed in order.
- A stack can be very useful for traversing a directory tree or similar structures.

Methods of Queue

When a deque is used as a queue, FIFO (First-In-First-Out) behavior results. Elements are added at the end of the deque and removed from the beginning.

Queue Method	**Equivalent Deque Method**
add(e)	addLast(e)
offer(e)	offerLast(e)
remove()	removeFirst()
poll()	pollFirst()
element()	getFirst()
peek()	peekFirst()

Methods of stack

Deques can also be used as LIFO (Last-In-First-Out) stacks. This interface should be used in preference to the legacy Stack class. When a deque is used as a stack, elements are pushed and popped from the beginning of the deque.

Stack Method	Equivalent Deque Method
push(e)	addFirst(e)
pop()	removeFirst()
peek()	peekFirst()

Map

HEXAWARE

Property of the Map Interface:

A map is a container that stores the elements along with the keys. The keys are like indexes. In List, the indexes are integers. In Map, the keys can be any objects. A map cannot contain duplicate keys. Each key maps to one value. A key and its corresponding value from an entry, which is actually stored in a map.

Under Queue we have concrete classes like

- HashMap,
- HashTable,
- TreeMap
- LinkedHashMap

Traverse of objects in Collection Framework

Traverse of collection object is possible through the interfaces like :

Iterator

- Iterator enables you to cycle through a collection, obtaining or removing elements
- An Iterator is an object that's associated with a specific collection. It let's you loop through the collection step by step. There are two important Iterator methods.
- Iteration processes will be happen through hasNext(),next() and remove()

List Iterator

- ListIterator extends Iterator to allow bidirectional traversal of a list, and the modification of elements.

Enumeration

- Enumeration interface used to Traverse through the object of legacy collection using its methods hasMoreElements() and nextElement()

Generics

Generics are one of the core feature of Java programming and it was introduced in Java 5.

Generics allows a type or method to operate on objects of various types while providing compile-time type safety that allows programmers to catch invalid types at compile time, making Java a fully statically typed language.

Syntax

ArrayList <String> list=new ArrayList<String>();

List <Sample> list=new ArrayList<String>();

Set <Integer> set=new HashSet<String>();

Collection before Java2
Collection in java2
Collection in java 1.5

Diamond Operator

Java7 replace the type set of type parameters (<>) This pair of angle brackets is informally called the *diamond*.

The Diamond Operator reduces some of Java's verbosity surrounding generics by having the compiler infer parameter types for constructors of generic classes.

Before java7

```
List<String> list = new ArrayList<String>();
```

In Java SE 7

```
List<String> list = new ArrayList<>();
```

LET ME RECOLLECT

Recollect

1) Which of the following are true about ListIterator and Iterator ?

a) ListIterator can traverse lists in both direction
b) ListIterator extends the Iterator interface
c) Iterator can traverse in forward direction only
d) Iterator can remove the elements

2) Which of the following Collection are synchronized by nature ?

a) Vector
b) SortedSet
c) Hashtable
d) HashMap

3) which of the following are false about Collections and Collection ?

a) Collections is a utility class
b) Collection is an interface to Set and List
c) Collections is a special type of collection which holds Set of collections
d) Both Collections and Collection entity belongs to java.util package.

Collection Assignment

Collection Assignment

HEXAWARE

Innovative Services

Passionate Employees

Delighted Customers

Thank you

www.hexaware.com