Java Collection Framework

Java Collections

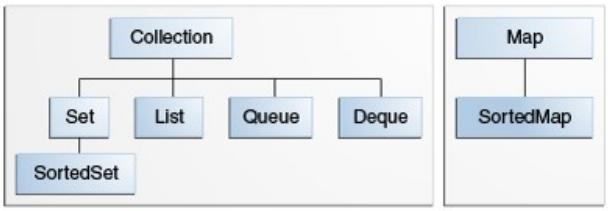
- A collection is an object that groups multiple elements into a single unit.
- Collections are used to store, retrieve, manipulate, and communicate aggregate data.

Collection Framework

- A collections framework is a unified architecture for representing and manipulating collections.
- **Interfaces:** These are abstract data types that represent collections. Interfaces allow collections to be manipulated independently of the details of their representation.
- **Implementations:** These are the concrete implementations of the collection interfaces. In essence, they are reusable data structures.
- **Algorithms:** These are the methods that perform useful computations, such as searching and sorting, on objects that implement collection interfaces.

3

Interfaces



4

Core Collection Interfaces

- Collection — the root of the collection hierarchy.
- List — an ordered collection (sometimes called a sequence). Lists can contain duplicate elements.
- Set — a collection that cannot contain duplicate elements.
- List — an ordered collection (sometimes called a sequence). Lists can contain duplicate elements.
- Queue — a collection used to hold multiple elements prior to processing. Besides basic Collection operations, a Queue provides additional insertion, extraction, and inspection operations.
- Map — an object that maps keys to values. A Map cannot contain duplicate keys; each key can map to at most one value.

Sorted versions of Set and Map

- SortedSet — a Set that maintains its elements in ascending order.
- SortedMap — a Map that maintains its mappings in ascending key order.

List Interface

- A List is an ordered Collection (sometimes called a sequence). Lists may contain duplicate elements. In addition to the operations inherited from Collection, the List interface includes operations for the following:
- Positional access — manipulates elements based on their numerical position in the list. This includes methods such as get, set, add, addAll, and remove.
- Search — searches for a specified object in the list and returns its numerical position. Search methods include indexOf and lastIndexOf.
- Iteration — extends Iterator semantics to take advantage of the list's sequential nature. The listIterator methods provide this behavior.
- Range-view — The sublist method performs arbitrary range operations on the list.

7

The Set Interface

- The Java platform contains three general-purpose Set implementations: HashSet, TreeSet, and LinkedHashSet.
- HashSet, which stores its elements in a hash table, is the best-performing implementation; however it makes no guarantees concerning the order of iteration.
- TreeSet, which stores its elements in a red-black tree, orders its elements based on their values; it is substantially slower than HashSet.
- LinkedHashSet, which is implemented as a hash table with a linked list running through it, orders its elements based on the order in which they were inserted into the set.

8

The Map Interface

- A Map is an object that maps keys to values.
- A map cannot contain duplicate keys: Each key can map to at most one value.
- The Map interface includes methods for basic operations (such as put, get, remove, containsKey, containsValue, size, and empty), bulk operations (such as putAll and clear), and collection views (such as keySet, entrySet, and values).
- The Java platform contains three general-purpose Map implementations: HashMap, TreeMap, and LinkedHashMap.
- Their behavior and performance are precisely analogous to HashSet, TreeSet, and LinkedHashSet, as described in The Set Interface section.

The Queue Interface

- A Queue is a collection for holding elements prior to processing. Besides basic Collection operations, queues provide additional insertion, removal, and inspection operations. The Queue interface follows.

```
public interface Queue<E> extends Collection<E> {
    E element();
    boolean offer(E e);
    E peek();
    E poll();
    E remove();
}
```

The Deque Interface

- Usually pronounced as deck, a deque is a double-ended-queue. A double-ended-queue is a linear collection of elements that supports the insertion and removal of elements at both end points.
- The Deque interface is a richer abstract data type than both Stack and Queue because it implements both stacks and queues at the same time.
- The Deque interface, defines methods to access the elements at both ends of the Deque instance.
- Methods are provided to insert, remove, and examine the elements. Predefined classes like ArrayDeque and LinkedList implement the Deque interface.