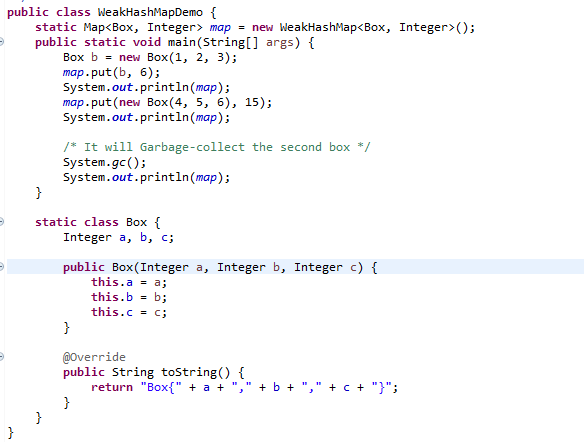
1. **WeakHashMap**

WeakHashMap allows its keys to be garbage-collected if there is no other valid reference remaining in the code. In the below code, we create a WeakHashMap and put two keys in it. 1st key is referenced by variable b throughout the life of the program and so it will not be garbage collected. 2nd key is not referenced by any variable and so its garbage collected as soon as GC runs.



**Unexpected behavior of a HashMap**

Because the garbage collector may discard keys at any time, a WeakHashMap behaves as if an unknown thread is silently removing entries from it. Even if the access to a WeakHashMap is synchronized, its size() and isEmpty() method can change values over time (as other threads die or lose reference to objects or the Garbage-Collector collects some keys in the background).

**WeakHashMap implementation**

WeakHashMap is very similar to a normal HashMap in terms of implementation except one important difference. It has a **ReferenceQueue** which is automatically updated by the GC for any objects which are about to be garbage collected. This queue is checked by the function **expungeStaleEntries()** . If any entry is found in this queue, then expungeStaleEntries() removes the same from the hash-map also.

This function is called before most map functions like size(), get() etc. thus making sure that whatever entry has been garbage-collected by GC, is also removed before any kind of access.

1. **TreeMap**

TreeMap is a map which keeps its keys in a sorted order. Internally TreeMap maintains a red-black binary tree of Map.Entry objects.

TreeMap does not use hashCode() or equals() at all! It solely relies on the compareTo() method and if your equals() does not comply with compareTo(), TreeMap does not care.

This could result in a serious application error where two objects considered equal by equals() return different values from TreeMap.

The TreeMap is so different from HashMap in implementation that its performance is also widely different.

For objects with good hashCode() methods, HashMap is blazing fast with O(1) time for inserts and lookups

While TreeMap cannot be faster than O(log N) even with the best compareTo() methods.

1. **LinkedHashMap**

LinkedHashMap is a map which maintains the order in which keys are inserted into it. LinkedHashMap remembers insertion order by maintaining a doubly-linked list of Map.Entry objects.

**Conceptual Implementation**

LinkedHashMap derives HashMap and overrides some of its methods to achieve the above functionality.

It also declares LinkedHashMap.Entry class as a derived class of HashMap.Entry to overrides some of its methods too.

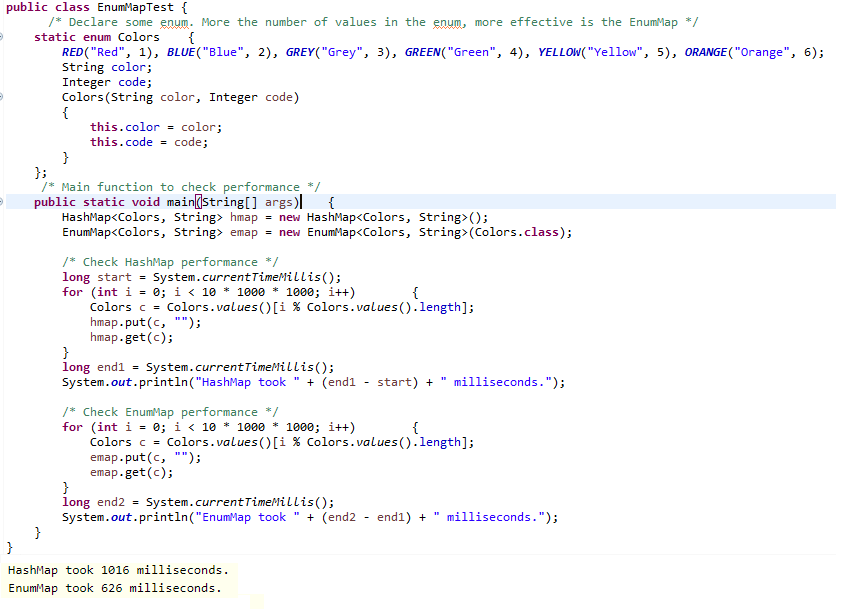
1. **IdentityHashMap**

IdentityHashMap compares keys by == instead of equals().

IdentityHashMap overrides the hash(key) method such that it calls System.identityHashCode(key) instead of calling the regular key.hashCode()

This overridden method is used for all the map functions like contains(key), put(key, value) etc.

1. **EnumMap**

EnumMap is a special Map whose keys can only be enum values. The enum values belong to a single enum only. Enum maps are represented internally as arrays which makes them extremely compact and efficient.

1. There are two common implementations of a queue, one uses a “circular array” and one uses a linked list. If you need a circular array queue, use the ArrayDeque class. For a linked list queue, simply use the LinkedList class—it implements the Queue interface.

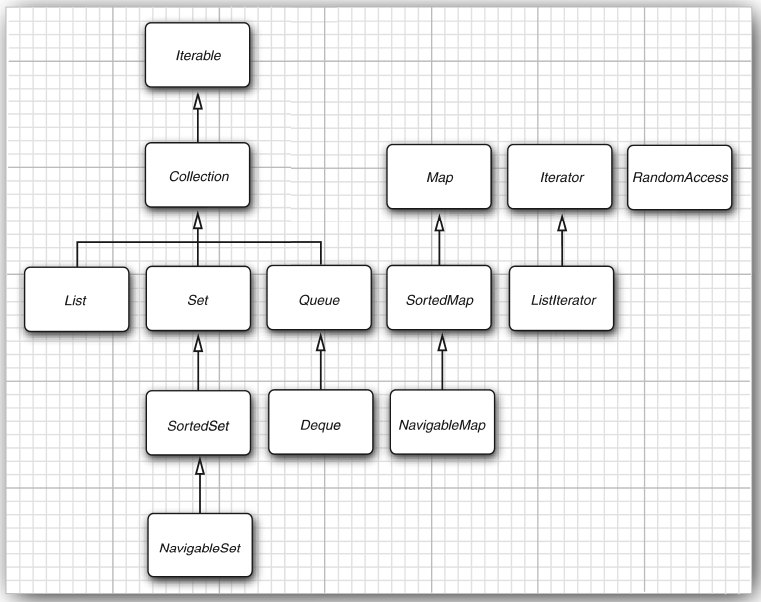
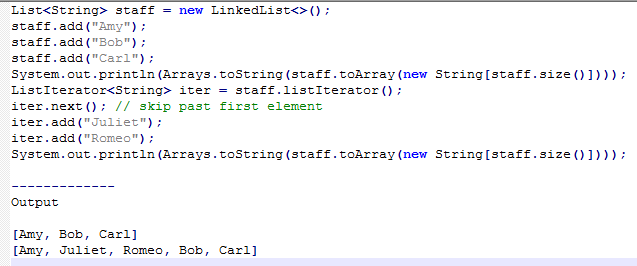


Figure 1: The interfaces of the collections framework

When we use the add operation with an iterator that was freshly returned from the listIterator method and that points to the beginning of the linked list, the newly added element becomes the new head of the list. When the iterator has passed the last element of the list (that is, when hasNext returns false), the added element becomes the new tail of the list. If the linked list has n elements, there are n + 1 spots for adding a new element.

These spots correspond to the n + 1 possible positions of the iterator. For example, if a linked list contains three elements, A, B, and C, there are four possible positions (marked as |) for inserting a new element:

|ABC

A|BC

AB|C

ABC|

