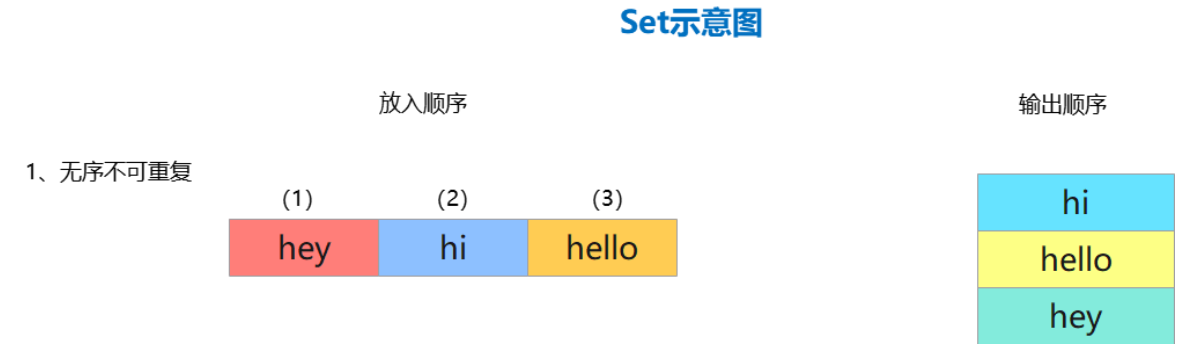
【第06话：经常被问到的Java集合中这些长得很像的兄弟】

List 和 Set 的区别?



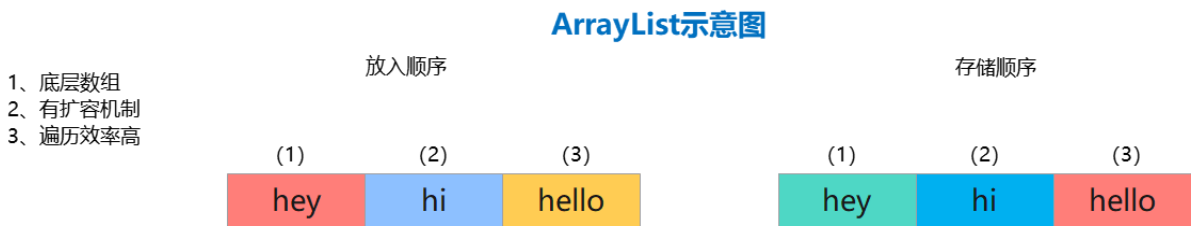
List和Set的区别?

这个问题比较简单，List和Set都是Collection接口的子接口。分别表示有序可重复和无序不可重复。具体点说：

List 接口：只要实现List接口都表示类中数据是有顺序的，存储时的顺序和添加顺序有关系。因为是有顺序的，所以里面的数据也是可以有重复数据的。

Set 接口：只要实现Set接口，里面的数据都是不允许重复的。且存储顺序和添加顺序是不一致的，所以认为是无序的。但是Set接口很多实现类都是基于Map实现的，所有存储的数据会按照特定的结构和顺序进行存储。我们每次从Set里面获取到的数据都是“固定顺序的”。

ArrayList和LinkedList的区别?



ArrayList和LinkedList区别?

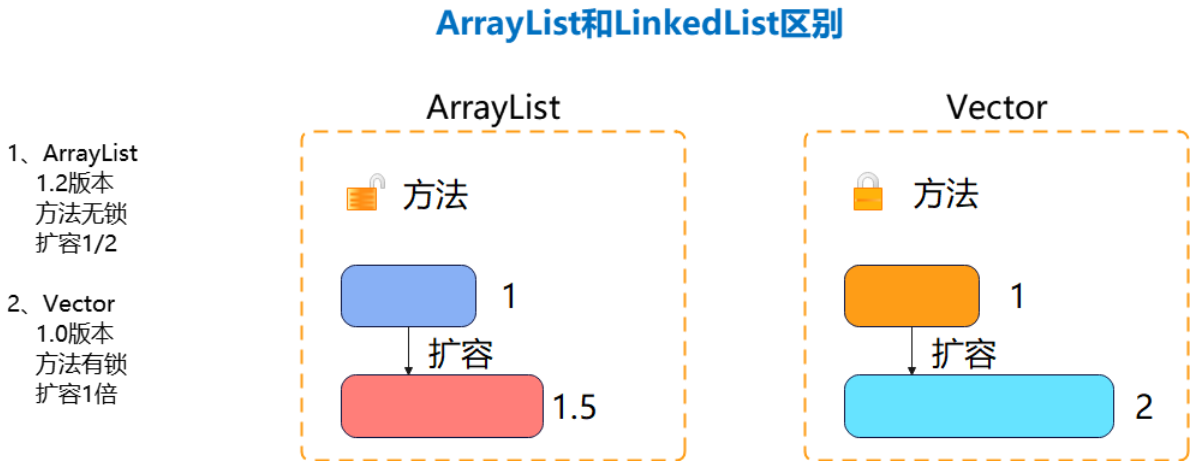
ArrayList和LinkedList都是List接口的实现类。这两个类都是存储有顺序、可重复数据的容器。他们的区别在于底层数据结构，ArrayList底层是数组，LinkedList底层是链表。

ArrayList：底层是数组，在添加数据时如果数组存放不下进行扩容。由于底层是数据，所以遍历效率较高。

LinkedList：底层是双向非循环链表。添加数据时会在链表末尾添加一个节点。由于底层是链表，新增和删除效率较高。并且LinkedList因为是链表结构所以在实现时相比ArrayList还多了addFirst()、addLast()、getFirst()、getLast()、removeFist()、removeLast()等头尾删除的方法。

同时因为LinkedList从1.6版本开始实现了Deque接口，LinkedList也表示基于链表实现的双向队列。还提供了push()、peek()和poll()等队列操作方法。

ArrayList和Vector的区别?



ArrayList：新增方法没有使用synchronized关键字修饰，get()、remove()方法也使用了synchronized修饰

```
public boolean add(E e) {
    ensureCapacityInternal( minCapacity: size + 1);  // Increments modCount!!
    elementData[size++] = e;
    return true;
}
```

ArrayList：扩容方法每次扩容1/2

```
  private void grow(int minCapacity) {
      // overflow-conscious code
      int oldCapacity = elementData.length;
      int newCapacity = oldCapacity + (oldCapacity >> 1);
      if (newCapacity - minCapacity < 0)
          newCapacity = minCapacity;
      if (newCapacity - MAX_ARRAY_SIZE > 0)
          newCapacity = hugeCapacity(minCapacity);
      // minCapacity is usually close to size, so this is a win:
      elementData = Arrays.copyOf(elementData, newCapacity);
  }
```

Vector：新增方法使用synchronized关键字修饰

```
  public synchronized boolean add(E e) {
      modCount++;
      ensureCapacityHelper( minCapacity: elementCount + 1);
      elementData[elementCount++] = e;
      return true;
  }
```

Vector：扩容方法每次扩容1倍

```
private void grow(int minCapacity) {
    // overflow-conscious code
    int oldCapacity = elementData.length;
    int newCapacity = oldCapacity + ((capacityIncrement > 0) ?
                                     capacityIncrement : oldCapacity);
    if (newCapacity - minCapacity < 0)
        newCapacity = minCapacity;
    if (newCapacity - MAX_ARRAY_SIZE > 0)
        newCapacity = hugeCapacity(minCapacity);
    elementData = Arrays.copyOf(elementData, newCapacity);
}
```

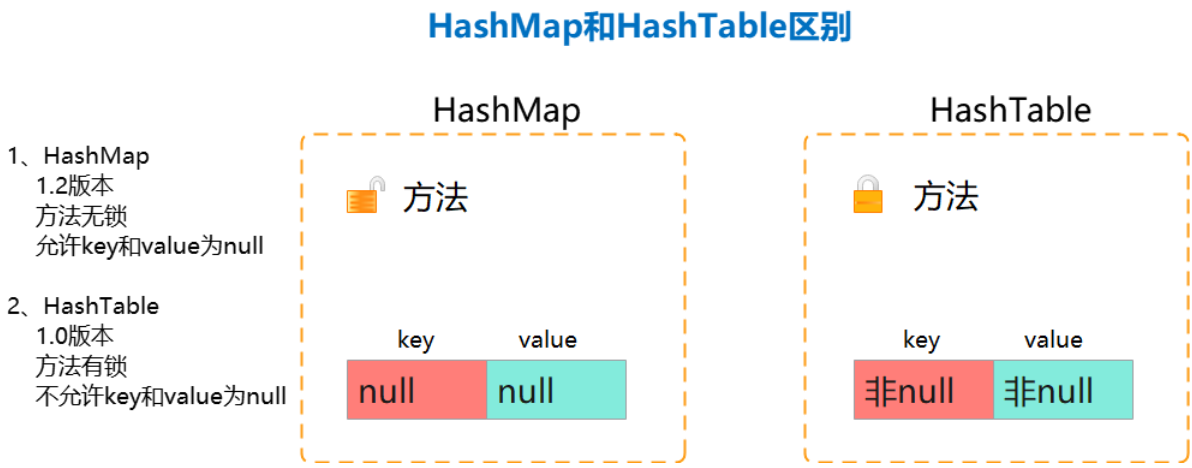
ArrayList和Vector的区别?

ArrayList和Vector都是List接口的实现类，存储的都是有序可重复数据，底层也都是数组。主要区别是ArrayList是1.2版本出现的，Vector是1.0就出现的。ArrayList每次扩容1/2，Vector每次扩容1倍。ArrayList是非线程安全的，Vector是线程安全的。

ArrayList：底层每次扩容1/2，新数组长度是原数组长度的3/2。且由于不是synchronized的方法，所以在多线程下效率更高，但不能保证线程安全性。

Vector：底层每次扩容1倍，新数组长度是原数组的2倍。方法使用synchronized关键字修饰，是线程安全的。多线程下其他线程必须等待当前线程执行完成才能执行。

HashMap和HashTable的区别?



HashMap的put()

```
public V put(K key, V value) {
    return putVal(hash(key), key, value, onlyIfAbsent: false, evict: true);
}
```

HashTable的put()


```
public synchronized V put(K key, V value) {
    // Make sure the value is not null
    if (value == null) {
        throw new NullPointerException();
    }

    // Makes sure the key is not already in the hashtable.
    Entry<?,?> tab[] = table;
    int hash = key.hashCode();
    int index = (hash & 0x7FFFFFFF) % tab.length;
    /unchecked/
    Entry<K,V> entry = (Entry<K,V>)tab[index];
    for(; entry != null ; entry = entry.next) {
        if ((entry.hash == hash) && entry.key.equals(key)) {
            V old = entry.value;
            entry.value = value;
            return old;
        }
    }

    addEntry(hash, key, value, index);
    return null;
}
```

HashMap和HashTable区别?

HashMap和HashTable都是Map接口的实现类，底层都是散列表。主要的区别是HashMap是非线程安全的，HashTable是线程安全的。HashMap允许Key为null，而HashTable不允许Key和value为null

HashMap：方法没有使用synchronized修饰。Key和Value都允许为null。但是由于Map的key是不允许重复的，第二次key=null时会覆盖上次的value。

HashTable：方法使用synchronized修饰。Key和Value都不允许为null。目前HashTable已经很少被使用了。如果希望在多线程下线程安全，推荐使用ConcurrentHashMap。