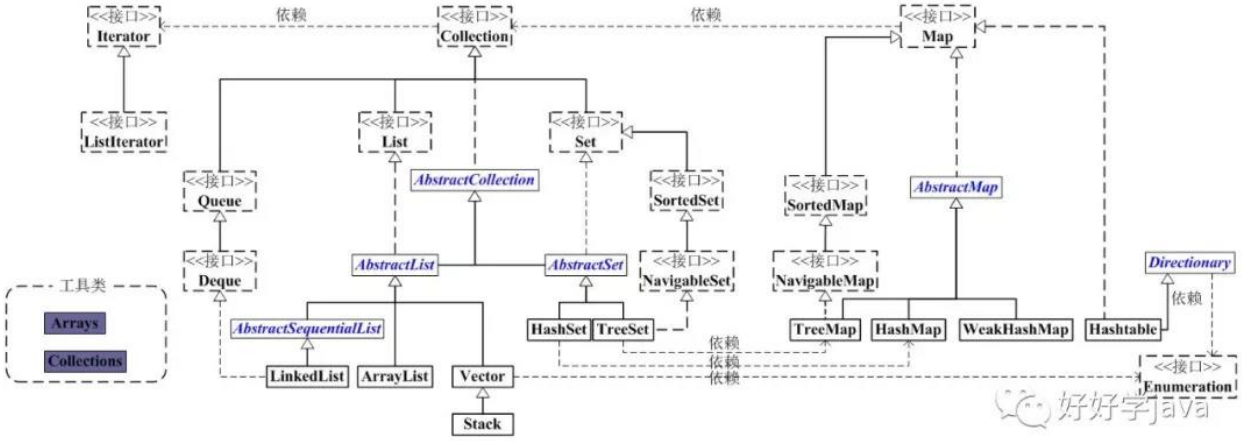
Java 集合系列（1）： Collection 架构

概要

首先，我们对 Collection 进行说明。下面先看看 Collection 的一些框架类的关系



Collection 是一个接口，它主要的两个分支是： List 和 Set。

List 和 Set 都是接口，它们继承于 Collection。**List 是有序的队列，List 中可以有重复的元素；而 Set 是数学概念中的集合，Set 中没有重复元素！**

List 和 Set 都有它们各自的实现类。

为了方便，我们抽象出了 AbstractCollection 抽象类，它实现了 Collection 中的绝大部分函数；这样，在 Collection 的实现类中，我们就可以通过继承 AbstractCollection 省去重复编码。AbstractList 和 AbstractSet 都继承于 AbstractCollection，具体的 List 实现类继承于 AbstractList，而 Set 的实现类则继承于 AbstractSet。

另外，Collection 中有一个 iterator()函数，它的作用是返回一个 Iterator 接口。通常，我们通过 Iterator 迭代器来遍历集合。ListIterator 是 List 接口所特有的，在 List 接口中，通过 ListIterator()返回一个 ListIterator 对象。

接下来，我们看看各个接口和抽象类的介绍；然后，再对实现类进行详细的了解。

1. Collection 简介

Collection 的定义如下：

```
public interface Collection<E> extends Iterable<E> {}
```

它是一个接口，是高度抽象出来的集合，它包含了集合的基本操作：添加、删除、清空、遍历(读取)、是否为空、获取大小、是否保护某元素等等。

Collection 接口的所有子类(直接子类和间接子类)都必须实现 2 种构造函数：不带参数

的构造函数 和 参数为 Collection 的构造函数。带参数的构造函数 ,可以用来转换 Collection 的类型。

```
// Collection 的 API
abstract boolean        add(E object)
abstract boolean        addAll(Collection<? extends E> collection)
abstract void           clear()
abstract boolean        contains(Object object)
abstract boolean        containsAll(Collection<?> collection)
abstract boolean        equals(Object object)
abstract int            hashCode()
abstract boolean        isEmpty()
abstract Iterator<E>    iterator()
abstract boolean        remove(Object object)
abstract boolean        removeAll(Collection<?> collection)
abstract boolean        retainAll(Collection<?> collection)
abstract int            size()
abstract <T> T[]        toArray(T[] array)
abstract Object[]       toArray()
```

2. List 简介

List 的定义如下：

```
public interface List<E> extends Collection<E> {}
```

List 是一个继承于 Collection 的接口，即 List 是集合中的一种。List 是有序的队列，List 中的每一个元素都有一个索引 ,第一个元素的索引值是 0 ,往后的元素的索引值依次+1。和 Set 不同，List 中允许有重复的元素。

关于 API 方面。既然 List 是继承于 Collection 接口，它自然就包含了 Collection 中的全部函数接口 ;由于 List 是有序队列 ,它也额外的有自己的 API 接口。主要有“添加、删除、获取、修改指定位置的元素”、“获取 List 中的子队列”等。

```
// Collection 的 API
abstract boolean        add(E object)
abstract boolean        addAll(Collection<? extends E> collection)
abstract void           clear()
abstract boolean        contains(Object object)
abstract boolean        containsAll(Collection<?> collection)
abstract boolean        equals(Object object)
abstract int            hashCode()
abstract boolean        isEmpty()
```

```
abstract Iterator<E>     iterator()
abstract boolean         remove(Object object)
abstract boolean         removeAll(Collection<?> collection)
abstract boolean         retainAll(Collection<?> collection)
abstract int             size()
abstract <T> T[]         toArray(T[] array)
abstract Object[]        toArray()
// 相比与 Collection，List 新增的 API：
abstract void               add(int location, E object)
abstract boolean            addAll(int location, Collection<? extends E>
collection)
abstract E                  get(int location)
abstract int                indexOf(Object object)
abstract int                lastIndexOf(Object object)
abstract ListIterator<E>     listIterator(int location)
abstract ListIterator<E>     listIterator()
abstract E                  remove(int location)
abstract E                  set(int location, E object)
abstract List<E>            subList(int start, int end)
```

3. Set 简介

Set 的定义如下：

```
public interface Set<E> extends Collection<E> {}
```

Set 是一个继承于 Collection 的接口，即 Set 也是集合中的一种。Set 是没有重复元素的集合。

关于 API 方面。Set 的 API 和 Collection 完全一样。

```
// Set 的 API
abstract boolean         add(E object)
abstract boolean         addAll(Collection<? extends E> collection)
abstract void            clear()
abstract boolean         contains(Object object)
abstract boolean         containsAll(Collection<?> collection)
abstract boolean         equals(Object object)
abstract int             hashCode()
abstract boolean         isEmpty()
abstract Iterator<E>     iterator()
abstract boolean         remove(Object object)
abstract boolean         removeAll(Collection<?> collection)
```

```
abstract boolean         retainAll(Collection<?> collection)
abstract int            size()
abstract <T> T[]         toArray(T[] array)
abstract Object[]         toArray()
```

4. AbstractCollection

AbstractCollection 的定义如下：

```
public abstract class AbstractCollection<E> implements Collection<E> {}
```

AbstractCollection 是一个抽象类，它实现了 Collection 中除 iterator()和 size()之外的函数。

AbstractCollection 的主要作用：它实现了 Collection 接口中的大部分函数。从而方便其它类实现 Collection，比如 ArrayList、LinkedList 等，它们这些类想要实现 Collection 接口，通过继承 AbstractCollection 就已经实现了大部分的接口了。

5. AbstractList

AbstractList 的定义如下：

```
public abstract class AbstractList<E> extends AbstractCollection<E> implements
List<E> {}
```

AbstractList 是一个继承于 AbstractCollection，并且实现 List 接口的抽象类。它实现了 List 中除 size()、get(int location)之外的函数。

AbstractList 的主要作用：它实现了 List 接口中的大部分函数。从而方便其它类继承 List。

另外，和 AbstractCollection 相比，AbstractList 抽象类中，实现了 iterator()接口。

6. AbstractSet

AbstractSet 的定义如下：

```
public abstract class AbstractSet<E> extends AbstractCollection<E> implements
Set<E> {}
```

AbstractSet 是一个继承于 AbstractCollection，并且实现 Set 接口的抽象类。由于 Set 接口和 Collection 接口中的 API 完全一样，Set 也就没有自己单独的 API。和 AbstractCollection 一样，它实现了 List 中除 iterator()和 size()之外的函数。

AbstractSet 的主要作用：它实现了 Set 接口中的大部分函数。从而方便其它类实现 Set 接口。

7. Iterator

Iterator 的定义如下：

```
public interface Iterator<E> {}
```

Iterator 是一个接口，它是集合的迭代器。集合可以通过 Iterator 去遍历集合中的元素。Iterator 提供的 API 接口，包括：是否存在下一个元素、获取下一个元素、删除当前元素。

注意：Iterator 遍历 Collection 时，是 fail-fast 机制的。即，当某一个线程 A 通过 iterator 去遍历某集合的过程中，若该集合的内容被其他线程所改变了；那么线程 A 访问集合时，就会抛出 ConcurrentModificationException 异常，产生 fail-fast 事件。关于 fail-fast 的详细内容，我们会在后面专门进行说明。

```
// Iterator 的 API
abstract boolean hasNext()
abstract E next()
abstract void remove()
```

8. ListIterator

ListIterator 的定义如下：

```
public interface ListIterator<E> extends Iterator<E> {}
```

ListIterator 是一个继承于 Iterator 的接口，它是队列迭代器。专门用于便利 List，能提供向前/向后遍历。相比于 Iterator，它新增了添加、是否存在上一个元素、获取上一个元素等等 API 接口。

```
// ListIterator 的 API
// 继承于 Iterator 的接口
abstract boolean hasNext()
abstract E next()
abstract void remove()
// 新增 API 接口
abstract void add(E object)
abstract boolean hasPrevious()
abstract int nextIndex()
abstract E previous()
abstract int previousIndex()
abstract void set(E object)
```