第九章 集合

面试题

1.Java集合框架是什么?说出一些集合框架的优点?

每种编程语言中都有集合,最初的Java版本包含几种集合类: Vector、Stack、HashTable和Array。随着集合的广泛使用,Java1.2提出了囊括所有集合接口、实现和算法的集合框架。在保证线程安全的情况下使用泛型和并发集合类,Java已经经历了很久。它还包括在Java并发包中,阻塞接口以及它们的实现。集合框架的部分优点如下:

- (1) 使用核心集合类降低开发成本,而非实现我们自己的集合类。
- (2) 随着使用经过严格测试的集合框架类,代码质量会得到提高。
- (3) 通过使用JDK附带的集合类,可以降低代码维护成本。
- (4) 复用性和可操作性。

2.集合框架中的泛型有什么优点?

Java1.5引入了泛型,所有的集合接口和实现都大量地使用它。泛型允许我们为集合提供一个可以容纳的对象类型,因此,如果你添加其它类型的任何元素,它会在编译时报错。这避免了在运行时出现ClassCastException,因为你将会在编译时得到报错信息。泛型也使得代码整洁,我们不需要使用显式转换和instanceOf操作符。它也给运行时带来好处,因为不会产生类型检查的字节码指令。

3.Java集合框架的基础接口有哪些?

Collection为集合层级的根接口。一个集合代表一组对象,这些对象即为它的元素。Java平台不提供这个接口任何直接的实现。

Set是一个不能包含重复元素的集合。这个接口对数学集合抽象进行建模,被用来代表集合,就如一副牌。

List是一个有序集合,可以包含重复元素。你可以通过它的索引来访问任何元素。**List**更像长度动态变换的数组。

Map是一个将**key**映射到**value**的对象.一个Map不能包含重复的key:每个key最多只能映射一个value。

一些其它的接口有Queue、Dequeue、SortedSet、SortedMap和ListIterator

4.为何Collection不从Cloneable和Serializable接口继承?

Collection接口指定一组对象,对象即为它的元素。如何维护这些元素由 Collection的具体实现决定。例如,一些如List的Collection实现允许重复的元素,而其它的如Set就不允许。很多Collection实现有一个公有的clone方法。然而,把它放到集合的所有实现中也是没有意义的。这是因为Collection是一个抽象表现。重要的是实现。

当与具体实现打交道的时候,克隆或序列化的语义和含义才发挥作用。所以,具体实现应该决定如何对它进行克隆或序列化,或它是否可以被克隆或序列化。

在所有的实现中授权克隆和序列化,最终导致更少的灵活性和更多的限制。特 定的实现应该决定它是否可以被克隆和序列化。

5.为何Map接口不继承Collection接口?

尽管Map接口和它的实现也是集合框架的一部分,但Map不是集合,集合也不是Map。因此,Map继承Collection毫无意义,反之亦然。

如果Map继承Collection接口,那么元素去哪儿? Map包含key-value对,它提供抽取key或value列表集合的方法,但是它不适合"一组对象"规范。

6.Iterator是什么?

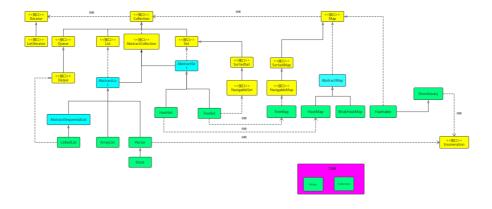
Iterator接口提供遍历任何**Collection**的接口。我们可以从一个**Collection**中使用迭代器方法来获取迭代器实例。迭代器取代了**Java**集合框架中的**Enumeration**。迭代器允许调用者在迭代过程中移除元素。

7.Enumeration和Iterator接口的区别?

Enumeration的速度是Iterator的两倍,也使用更少的内存。Enumeration是非常基础的,也满足了基础的需要。但是,与Enumeration相比,Iterator更加安全,因为当一个集合正在被遍历的时候,它会阻止其它线程去修改集合。

迭代器取代了Java集合框架中的Enumeration。迭代器允许调用者从集合中移除元素,而Enumeration不能做到。为了使它的功能更加清晰,迭代器方法名已经经过改善。

Java的集合类主要由三个接口派生而出,即Collection和Map接口和Iterator接口



LinkedList

```
public boolean add(E e) {
    linkLast(e);
    return true;
}

void linkLast(E e) {
    final Node<E> l = last;
    final Node<E> newNode = new Node<>(l, e, null);
    last = newNode;
    if (l == null)
        first = newNode;
    else
        l.next = newNode;
    size++;
    modCount++;
}
```