第十四章 集合类库（上）

14.1 集合的概述（重点）

14.1.1 集合的由来

- 当需要在Java程序中记录单个数据内容时，则声明一个变量。
- 当需要在Java程序中记录多个类型相同的数据内容时，声明一个一维数组。
- 当需要在Java程序中记录多个类型不同的数据内容时，则创建一个对象。
- 当需要在Java程序中记录多个类型相同的对象数据时，创建一个对象数组。
- 当需要在Java程序中记录多个类型不同的对象数据时，则准备一个集合。

14.1.2 集合的框架结构

- Java中集合框架顶层框架是：java.util.Collection集合 和 java.util.Map集合。
- 其中Collection集合中存取元素的基本单位是：单个元素。
- 其中Map集合中存取元素的基本单位是：单对元素。

14.2 Collection集合（重点）

14.2.1 基本概念

- java.util.Collection接口是List接口、Queue 接口以及Set接口的父接口，因此该接口里定义的方法既可用于操作List集合，也可用于操作Queue集合和Set集合。

14.2.1 常用的方法（练熟、记住）

方法声明	功能介绍
boolean add(E e);	向集合中添加对象
boolean addAll(Collection<? extends E> c)	用于将参数指定集合c中的所有元素添加到当前集合中
boolean contains(Object o);	判断是否包含指定对象
boolean containsAll(Collection<?> c)	判断是否包含参数指定的所有对象
boolean retainAll(Collection<?> c)	保留当前集合中存在且参数集合中存在的所有对象
boolean remove(Object o);	从集合中删除对象
boolean removeAll(Collection<?> c)	从集合中删除参数指定的所有对象
void clear();	清空集合
int size();	返回包含对象的个数
boolean isEmpty();	判断是否为空
boolean equals(Object o)	判断是否相等
int hashCode()	获取当前集合的哈希码值
Object[] toArray()	将集合转换为数组
Iterator iterator()	获取当前集合的迭代器

14.3 Iterator接口（重点）

14.3.1 基本概念

- java.util.Iterator接口主要用于描述迭代器对象，可以遍历Collection集合中的所有元素。
- java.util.Collection接口继承Iterator接口，因此所有实现Collection接口的实现类都可以使用该迭代器对象。

14.3.2 常用的方法

方法声明	功能介绍
boolean hasNext()	判断集合中是否有可以迭代/访问的元素
E next()	用于取出一个元素并指向下一个元素
void remove()	用于删除访问到的最后一个元素

- 案例题目：

  如何使用迭代器实现toString方法的打印效果？

14.4 for each循环（重点）

14.4.1 基本概念

- Java5推出了增强型for循环语句，可以应用数组和集合的遍历。
- 是经典迭代的“简化版”。

14.4.2 语法格式

- for(元素类型 变量名 : 数组/集合名称) {
  循环体;
  }

14.4.3 执行流程

- 不断地从数组/集合中取出一个元素赋值给变量名并执行循环体，直到取完所有元素为止。

14.5 List集合（重中之重）

14.5.1 基本概念

- java.util.List集合是Collection集合的子集合，该集合中允许有重复的元素并且有先后放入次序。
- 该集合的主要实现类有：ArrayList类、LinkedList类、Stack类、Vector类。
- 其中ArrayList类的底层是采用动态数组进行数据管理的，支持下标访问，增删元素不方便。
- 其中LinkedList类的底层是采用双向链表进行数据管理的，访问不方便，增删元素方便。
- 可以认为ArrayList和LinkedList的方法在逻辑上完全一样，只是在性能上有一定的差别，ArrayList更适合于随

  机访问而LinkedList更适合于插入和删除；在性能要求不是特别苛刻的情形下可以忽略这个差别。
- 其中Stack类的底层是采用动态数组进行数据管理的，该类主要用于描述一种具有后进先出特征的数据结构，叫做栈(last in first out LIFO)。
- 其中Vector类的底层是采用动态数组进行数据管理的，该类与ArrayList类相比属于线程安全的类，效率比较低，以后开发中基本不用。

14.5.2 常用的方法

方法声明	功能介绍
void add(int index, E element)	向集合中指定位置添加元素
boolean addAll(int index, Collection<? extends E> c)	向集合中添加所有元素
E get(int index)	从集合中获取指定位置元素
int indexOf(Object o)	查找参数指定的对象
int lastIndexOf(Object o)	反向查找参数指定的对象
E set(int index, E element)	修改指定位置的元素
E remove(int index)	删除指定位置的元素
List subList(int fromIndex, int toIndex)	用于获取子List

- 案例题目

  准备一个Stack集合，将数据11、22、33、44、55依次入栈并打印，然后查看栈顶元素并打印，然后将栈中所有数据依次出栈并打印。

  再准备一个Stack对象，将数据从第一个栈中取出来放入第二个栈中，然后再从第二个栈中取出并打印。

14.6 Queue集合（重点）

14.6.1 基本概念

- java.util.Queue集合是Collection集合的子集合，与List集合属于平级关系。
- 该集合的主要用于描述具有先进先出特征的数据结构，叫做队列(first in first out FIFO)。
- 该集合的主要实现类是LinkedList类，因为该类在增删方面比较有优势。

14.6.2 常用的方法

方法声明	功能介绍
boolean offer(E e)	将一个对象添加至队尾，若添加成功则返回true
E poll()	从队首删除并返回一个元素
E peek()	返回队首的元素（但并不删除）

- 案例题目

  准备一个Queue集合，将数据11、22、33、44、55依次入队并打印，然后查看队首元素并打印，然后将队列中所有数据依次出队并打印。