java集合

一、集合的概念

1、数学概念

由一个或多个确定的元素所构成的整体。

通俗的说：集合是一个数据容器，可以装多个数据。每一个数据称之为集合的一个元素。

2、java中集合的概念

Java集合类存放于 java.util 包中，是一个用来存放对象的容器。

以前接触过得容器：数组、StringBuffer（String的容器）

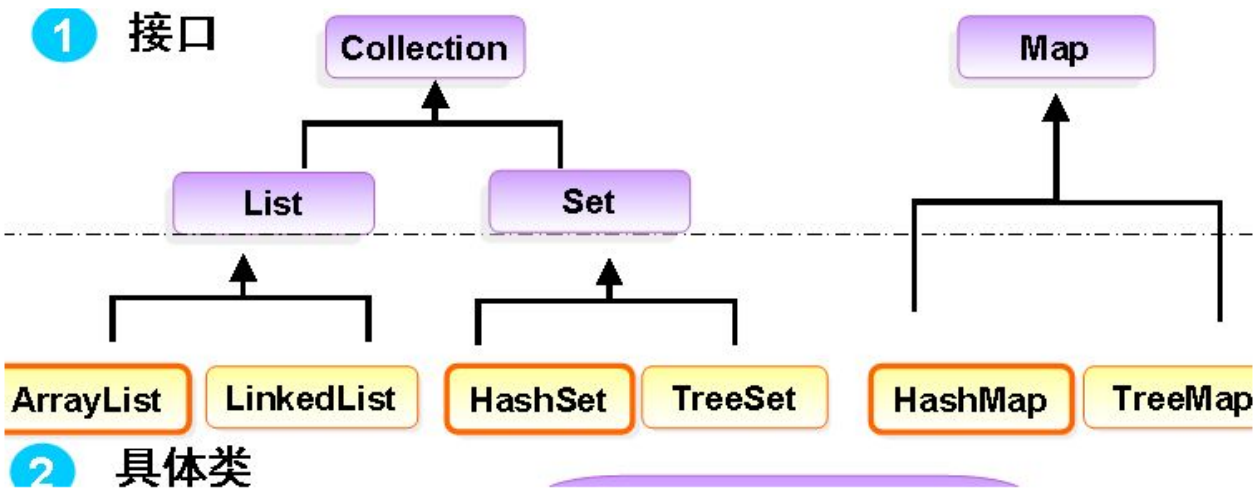
2.1 集合与数组

数组：（可以存储基本数据类型）是用来存现对象的一种容器，但是数组的长度固定，

不适合在对象数量未知的情况下使用。

集合：（只能存储对象，对象类型可以不一样）的长度可变，可在多数情况下使用。

二、java中的集合结构



在上图中：紫色部分为集合的主要接口。

在java中，集合分为两大类：

Collection： 线性结构的集合（数组就是一个线性结构）

List： 有序、可重复

Set： 无序、不可重复

Map： 键值（key-value）对的集合

三、List

List里存放的对象是有序的，同时也是可以重复的，List关注的是下标（List可以将元素非常精确的插入到集中的具体位置），拥有一系列和下标相关的方法，查询速度快。因为往list集合里插入或删除数据时，会伴随着后面数据的移动，所以插入删除数据速度慢。

在API中，List中的方法大概分为如下几类：

新增数据：add

删除数据：remove，removeAll

更新数据：repalce、set

查询数据：get

返回集合的大小：size

（一）ArrayList

ArrayList是一个基于数组结构而实现的一个List集合。

与数组不同的是，ArrayList可以调整大小。

ArrayList是List接口的一个实现类，List接口定义的方法，ArrayList都有实现。如上所述。

1、ArrayList的实例

```
public class ArrayListDemo {

    public static void main(String[] args) {
        List<String> list = new ArrayList<String>();

        //新增元素
        list.add("abcd");
        list.add("xyz");
        list.add("123");

        printList(list);

        //修改元素
        list.set(1, "789");
        System.out.println("++++++++++++");

        printList(list);
        System.out.println("++++++++++++");
```

```
        //删除元素
        list.remove(1);
        printList(list);

    }

    private static void printList(List<String> list) {
        //迭代集合（遍历集合），size()获取集合的长度
        for (int i = 0; i < list.size(); i++) {
            //获取集合中的元素
            System.out.println(list.get(i));
        }
    }
}
```

ArrayList中，维护一个数组，我们调用list.add方法的时候，其本质将数据放入到ArrayList中的数组，

Object[] elementData;

当第一次往ArrayList中新增元素的时候，会初始化这个数组，长度默认为10，但是由于数组的长度不可变，

当元素的数量达到10个的时候，再次调用add方法的时候，就需要对数组扩容。

第一次扩： int newCapacity = oldCapacity + (oldCapacity >> 1);

oldCapacity = 10，oldCapacity >> 1的结果是：5

2、集合的迭代

三种方式用来遍历集合：

1、for循环

2、foreach

3、iterator迭代器

```
private static void printList(List<String> list) {
        System.out.println("======for循环=======");
        for (int i = 0; i < list.size(); i++) {
            System.out.println(list.get(i));
        }

        System.out.println("======foreach=======");
        for (String str : list) {
            System.out.println(str);
        }

        System.out.println("======iterator迭代器=======");
```

```
        Iterator<String> iterator = list.iterator();
        while (iterator.hasNext()) {
            String next = iterator.next();
            System.out.println(next);
        }
    }
```

四、Collection接口的API

Empty ：容器是存在的，但是容器没有元素在里面。

瓶子是空的，盒子是空的，教室是空的。

Null： 空值。容器根本就不存在。

```
//方法
isEmpty() 判断集合中是否有意元素。返回boolean

package com.psfd.util;

import java.util.ArrayList;
import java.util.List;

public class Demo {
    public static void main(String[] args) {
        List<String> addrList1 = new ArrayList<>();
        addrList1.add("坪山");
        addrList1.add("福田");
        addrList1.add("南山");

        List<String> addrList2 = new ArrayList<>();
        addrList2.add("坑梓");
        addrList2.add("福田");
        addrList2.add("罗湖");

        List<String> addrList3 = new ArrayList<>();
        addrList3.add("坪山");
        addrList3.add("福田1");

        //contains  判断集合中是否包含某个元素
        System.out.println(addrList1.contains("南山"));

        //containsAll 判断集合addrList1中是否完全包含集合addrList3
        System.out.println(addrList1.containsAll(addrList3));

        //retainAll 取交集。仅仅保留集合addrList1中与集合addrList2中相同的元素。
        //不存在的元素，会被删除。如果没有交集，元素会被全部删除。
//      System.out.println(addrList1.retainAll(addrList2));
//      System.out.println("addrList1 = " + addrList1);
//      System.out.println("addrList2 = " + addrList2);
```

```
        //addAll 取并集，将另一个集合的元素全部添加到当前集合中
        addrList1.addAll(addrList2);
        System.out.println(addrList1);

        //indexOf  获取指定元素在集合中第一次出现的下标
        System.out.println(addrList1.indexOf("南山"));

        //lastIndexOf  获取指定元素在集合中最后一次出现的下标
        System.out.println(addrList1.lastIndexOf("福田"));

        //removeAll 删除当前集合中存在，并且也在指定集合中存在的元素。
//      addrList1.removeAll(addrList2);
//      System.out.println(addrList1);

        //subList  从当前集合中，截取一个子集合。>=1，<4
        //该方法返回一个新的集合，原来的集合不会改变。
        List<String> subList = addrList1.subList(1, 4);
        System.out.println(subList);
        System.out.println(addrList1);
    }

}
```

班级是一个集合。

班级新进来一个同学，用add

新进来一批同学，addAll

走了一个同学 ，remove

走了一部分同学，removeAll

班级是否有学生，isEmpty

班级是否有其他班级的学生，retainAll

班级有多少学生，size

五、LinkedList

LinkedList是一个基于双向链表结构的List集合。

ArrayList是一个基于数组结构的List集合。

5.1 数组和链表的区别

数组就像是我们出操的时候排成一列。

双向链表等同于围成一圈，手拉手。

很明显：排成一列的时候，查找某个元素，效率会比较快。但是如果增加，或者删除队列中的某个人，

会导致其他人的位置移动，因此，效率低。

围成一圈的时候，查找某个元素，会比较慢。但是，增加和删除，会比较快。

因此：数组的查找速度会快，但是增删的效率会慢。

链表查找速度慢，但是增删的效率快。

5.2 LinkedList方法测试

```
package com.psfd.util.list;

import java.util.LinkedList;

public class LinkedListDemo {

    public static void main(String[] args) {
        LinkedList<String> linkedList = new LinkedList<>();
        linkedList.add("坪山");
        linkedList.add("福田");
        linkedList.add("罗湖");
        linkedList.add("南山");

        System.out.println(linkedList);
        //addFirst 和 addLast    将元素添加到集合的头和尾
        linkedList.addFirst("宝安");
        System.out.println(linkedList);

        System.out.println("====================");

        //element  返回集合的第一个元素，但是不删除该元素,等价于：get(0)
        String element = linkedList.element();
        System.out.println(element);
        System.out.println(linkedList);

        //peek()、peekFirst、 peekLast  返回集合的头或者尾元素，都不删除。
        //等同于：get(0)，get(list.size() - 1)
        String peek = linkedList.peek();
        System.out.println(peek);

        System.out.println(linkedList.peekFirst());
        System.out.println(linkedList.peekLast());
        System.out.println(linkedList);

        //offer、offerFirst、offerLast  新增元素到集合的头或者尾
        //等同于：add(0,""),add("")
        System.out.println(linkedList.offer("龙华"));
        System.out.println(linkedList.offerFirst("龙岗"));
        System.out.println(linkedList);

        System.out.println("====================");
```

```
        //poll、pollFirst、pollLast  返回集合的头或者尾元素，会删除。
        //remove(0),remove(list.size() - 1)
        System.out.println(linkedList.poll());
        System.out.println(linkedList);

        //pop  弹出栈顶元素 。返回第一个元素，并且删除
        //remove(0)
        System.out.println(linkedList.pop());
        System.out.println(linkedList);

        //push 将元素压入栈。放在第一个位置
        //add(0,""),addFrist()
        linkedList.push("光明");
        System.out.println(linkedList);

        //removeFist 、 removeLast
        //删除第一个，删除最后一一个

    }
}
```

六、 Vector

Vector与ArrayList的底层完全一致，是基于数组而来的一个List集合。

但是： Vector是一个很古老的集合类，Vector对比ArrayList，区别在于：线程安全。

Vector是线程安全的集合，因此他的性能比ArrayList低。

ArrayList是线程不安全的集合，性能比Vector高。

七、Stack

Stack是Vector的子类。

LIFO : Last In First Out

后进先出（栈）

FIFO : First In First Out

先进先出（队列）

七、四种List集合的性能测试

```
package com.psfd.util.list;

import java.util.ArrayList;
import java.util.LinkedList;
```

```
import java.util.List;
import java.util.Stack;
import java.util.Vector;

public class ListTest {

    private static int COUNT = 100000;

    public static void main(String[] args) {
        ArrayList<String> arrayList = new ArrayList<>();
        LinkedList<String> linkedList = new LinkedList<>();
        Vector<String> vector = new Vector<>();
        Stack<String> stack = new Stack<>();

        System.out.println("新增性能测试");
        insetTest(arrayList);
        insetTest(linkedList);
        insetTest(vector);
        insetTest(stack);

        System.out.println("查询性能测试");
        queryTest(arrayList);
        queryTest(linkedList);
        queryTest(vector);
        queryTest(stack);

        System.out.println("删除性能测试");
        deleteTest(arrayList);
        deleteTest(linkedList);
        deleteTest(vector);
        deleteTest(stack);
    }

    public static void insetTest(List<String> list) {
        long beginTm = System.currentTimeMillis();

        for (int i = 0; i < COUNT; i++) {
            list.add(0, String.valueOf(i));
        }
        long endTm = System.currentTimeMillis();
        System.out.println(String.format("新增%d个元素，%s集合耗时为%d毫秒", COUNT,
getListName(list), (endTm - beginTm)));
    }

    public static void queryTest(List<String> list) {
        long beginTm = System.currentTimeMillis();

        for (int i = 0; i < COUNT; i++) {
            list.get(i);
        }
        long endTm = System.currentTimeMillis();
        System.out.println(String.format("查询%d个元素，%s集合耗时为%d毫秒", COUNT,
getListName(list), (endTm - beginTm)));
```

```
    }

    public static void deleteTest(List<String> list) {
        long beginTm = System.currentTimeMillis();

        for (int i = 0; i < COUNT; i++) {
            list.remove(0);
        }
        long endTm = System.currentTimeMillis();
        System.out.println(String.format("删除%d个元素，%s集合耗时为%d毫秒", COUNT,
getListName(list), (endTm - beginTm)));
    }

    public static String getListName(List<String> list) {
        if (list instanceof ArrayList) {
            return "ArrayList";
        } else if (list instanceof LinkedList) {
            return "LinkedList";
        } else if (list instanceof Vector) {
            return "Vector";
        }
        return "Stack";
    }
}
```

测试结果

```
新增性能测试
新增100000个元素，ArrayList集合耗时为1034毫秒
新增100000个元素，LinkedList集合耗时为18毫秒
新增100000个元素，Vector集合耗时为1002毫秒
新增100000个元素，Vector集合耗时为983毫秒
查询性能测试
查询100000个元素，ArrayList集合耗时为5毫秒
查询100000个元素，LinkedList集合耗时为12473毫秒
查询100000个元素，Vector集合耗时为5毫秒
查询100000个元素，Vector集合耗时为6毫秒
删除性能测试
删除100000个元素，ArrayList集合耗时为983毫秒
删除100000个元素，LinkedList集合耗时为4毫秒
删除100000个元素，Vector集合耗时为1014毫秒
删除100000个元素，Vector集合耗时为983毫秒
```

总结：

1、Vector和ArrayList，如果要确保线程安全，使用Vector或者Stack，否则使用ArrayList

2、如果要求随机访问数据的性能，使用ArrayList

3、如果要求随机删除和增加数据的性能，使用LinkedList

八、作业

分别使用ArrayList和LinkedList实现一个队列（FIFO）和栈(LIFO)。