类集：

类集其实是一个动态的数组，长度不固定。

类集可以保存不同类型的对象，但是呢我们一般建议大家不要这么做。

Connection接口：

经常使用的是其子接口List

ArrayList是List接口的实现类

90%的情况下，单元素类集我们会使用List,或者说用ArraryList

List是类集的重点：

添加元素的方法：boolean **add**([E](mk:@MSITStore:C:\Documents%20and%20Settings\Administrator\桌面\JDK_API_1_6_zh_CN.CHM::/java/util/List.html) e)；

指定的位置添加元素：void **add**(int index,[E](mk:@MSITStore:C:\Documents%20and%20Settings\Administrator\桌面\JDK_API_1_6_zh_CN.CHM::/java/util/List.html) element)

清除列表中所有元素：void **clear**()

移除指定位置元素：[E](mk:@MSITStore:C:\Documents%20and%20Settings\Administrator\桌面\JDK_API_1_6_zh_CN.CHM::/java/util/List.html) **remove**(int index)

移除列表第一次出现的元素：boolean **remove**([Object](mk:@MSITStore:C:\Documents%20and%20Settings\Administrator\桌面\JDK_API_1_6_zh_CN.CHM::/java/lang/Object.html) o)

查找（列表中是否有此元素）：boolean **contains**([Object](mk:@MSITStore:C:\Documents%20and%20Settings\Administrator\桌面\JDK_API_1_6_zh_CN.CHM::/java/lang/Object.html) o)；

查找指定索引的元素: [E](mk:@MSITStore:C:\Documents%20and%20Settings\Administrator\桌面\JDK_API_1_6_zh_CN.CHM::/java/util/List.html) **get**(int index);

给指定位置设置新的元素[E](mk:@MSITStore:C:\Documents%20and%20Settings\Administrator\桌面\JDK_API_1_6_zh_CN.CHM::/java/util/List.html) **set**(int index,[E](mk:@MSITStore:C:\Documents%20and%20Settings\Administrator\桌面\JDK_API_1_6_zh_CN.CHM::/java/util/List.html) element)

|  |
| --- |
| **package** com.wanczy.ListDemo;  **import** java.util.ArrayList;  **import** java.util.List;  **public** **class** ListDemo01 {  **public** **static** **void** main(String[] args) {  List<String> list=**new** ArrayList<String>();  list.add("第一个元素");  list.add(0,"我才是第一个元素");  list.add("123");  **for**(String p:list){  System.*out*.println(p);//增强的for循环  }  System.*out*.println(list.contains("123"));  System.*out*.println(list.get(2));  }  } |

上面的代码，我们使用到了增强型的for循环，他是通常用于类集的遍历

那么List也有一个查找第一次出现此元素的方法:int indexOf(Object o);

判断这个集合是否为空: boolean **isEmpty**(),如果希望返回list的长度来判断是否为空的话，那么我们使用int size();

类集的输出：

最正式的输出方式：

使用Iterator迭代器进行输出：

|  |
| --- |
| **package** com.wanczy.ListDemo;  **import** java.util.ArrayList;  **import** java.util.Iterator;  **import** java.util.List;  **public** **class** ListDeom02 {  **public** **static** **void** main(String[] args) {  List<String> list=**new** ArrayList<String>();  list.add("第一个元素");  list.add(0,"我才是第一个元素");  list.add("123");  Iterator<String> it= list.iterator();  **while**(it.hasNext()){  String str=it.next();//迭代输出  System.*out*.println(str);  }  }  } |

LinkedList实现类：

这个实现类的用法跟ArrayList基本是一致的：

前者基于链表的数据结构，后者是动态数组的数据结构；

如果对于任意方位的get和set，前者由于需要移动指针，所以，性能不如后者

如果对于add和remove操作而言，前者就比较占优势，因为后者需要对数据进行移动

Set类集：

他也是Connection下面的一个接口。

虽然说set和list的使用方式基本一致，但是有一个很大区别就是，Set存放的元素是不允许重复的

|  |
| --- |
| **package** com.wanczy.ListDemo;  **import** java.util.HashSet;  **import** java.util.Iterator;  **import** java.util.Set;  **public** **class** ListDeom02 {  **public** **static** **void** main(String[] args) {  Set<String> set=**new** HashSet<String>();  set.add("第一个元素");    set.add("123");  set.add("123");  Iterator<String> it= set.iterator();  **while**(it.hasNext()){  String str=it.next();//迭代输出  System.*out*.println(str);  }  }  } |

从上面的代码我们可以看到，Set的实现类是HashSet，也是一个经常使用到的实现类。

HashSet类集之中存放的元素是无序

|  |
| --- |
| **package** com.wanczy.ListDemo;  **import** java.util.HashSet;  **import** java.util.Iterator;  **import** java.util.Set;  **public** **class** ListDeom02 {  **public** **static** **void** main(String[] args) {  Set<String> set=**new** HashSet<String>();  set.add("第一个元素");  set.add("321");  set.add("123");  set.add("333");  Iterator<String> it= set.iterator();  **while**(it.hasNext()){  String str=it.next();//迭代输出  System.*out*.println(str);  }  }  } |

如果我们希望set集合是有序的：

TreeSet可以满足我们的需求。这也算他跟HashSet的唯一区别。

Map：

该类集里存放的元素，是一个键值对。Key🡪Value

我们经常使用Map来进行的是数据的查找，而不是输出

Map的键值也是一样的，不能重复(因为Map的键值只能转化为Set集合)

使用时[Set](mk:@MSITStore:C:\Documents%20and%20Settings\Administrator\桌面\JDK_API_1_6_zh_CN.CHM::/java/util/Set.html)<[K](mk:@MSITStore:C:\Documents%20and%20Settings\Administrator\桌面\JDK_API_1_6_zh_CN.CHM::/java/util/Map.html)> **keySet**()

Map添加元素使用的是put(key,value);

Map里面的get(key)方法其实是通过key值查找value的方法

Map的输出：

先将key转化成set集合，然后通过get（key）方法来获得value

|  |
| --- |
| **import** java.util.HashMap;  **import** java.util.Iterator;  **import** java.util.Map;  **import** java.util.Set;  **public** **class** MapDemo01 {  **public** **static** **void** main(String[] args) {  Map<String,String> m= **new** HashMap<String,String>();  m.put("张军师", "DH0201");  m.put("方倩", "DH0202");  // System.out.println(m.get("张军师"));  Set<String> set=m.keySet();  Iterator<String> it=set.iterator();  **while**(it.hasNext()){  String key= it.next();  System.*out*.println("名字："+key+"编号："+m.get(key));  }    }  } |

Map最标准的输出方式，可以实现键与值的分离——使用Map.entry对象

|  |
| --- |
| **package** com.wanczy.ListDemo;  **import** java.util.HashMap;  **import** java.util.Iterator;  **import** java.util.Map;  **import** java.util.Set;  **public** **class** MapDemo01 {  **public** **static** **void** main(String[] args) {  Map<String,String> m= **new** HashMap<String,String>();  m.put("张军师", "DH0201");  m.put("方倩", "DH0202");  Set<Map.Entry<String, String>> set=m.entrySet();  Iterator<Map.Entry<String, String>> it=set.iterator();  **while**(it.hasNext()){  Map.Entry<String, String> me=it.next();  System.*out*.println("名字："+me.getKey()+"\t"+"编号:"+me.getValue());  }    }  } |

HashMap是Map的实现类：

允许一个键值为null；

TreeMap方法：是有序的存放，还是根据key升序的排列，但是他就不允许有键值为null

Vector: 类集

这个类集跟List基本是一致的用法，他就想到与List在java1.2版本之前的作用

它使用的输出时枚举（Enumeration）输出

区别

|  |  |
| --- | --- |
| List | Vector |
| Iterator | Enumeration |
| java1.2之后出现的 | java1.1之前版本 |
| 使用较多 | 使用较少 |
| 异步 | 同步 |

练习：使用集合保存对象，进行增删改查