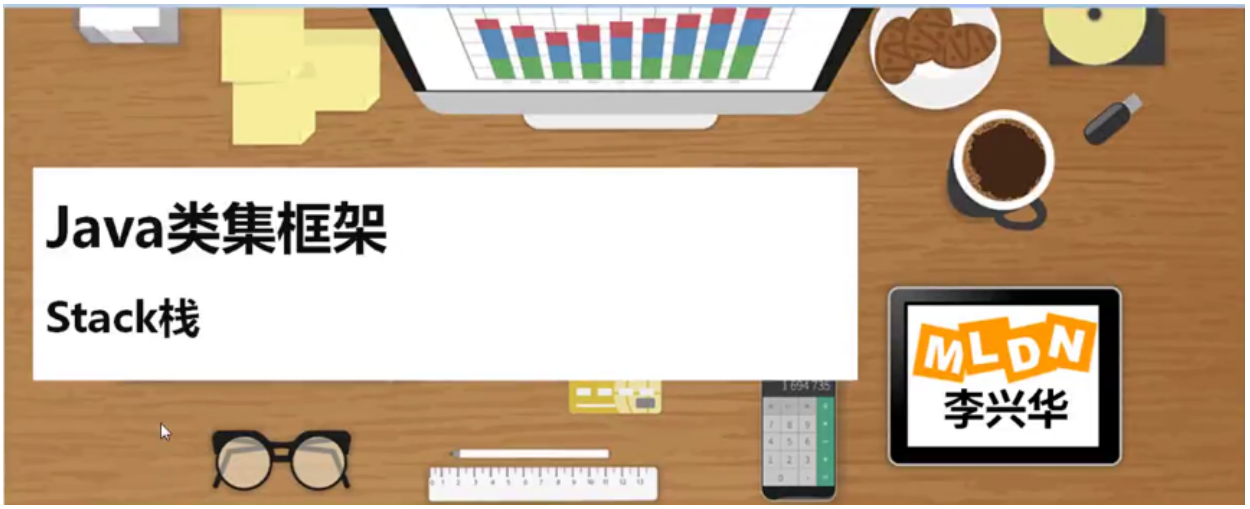
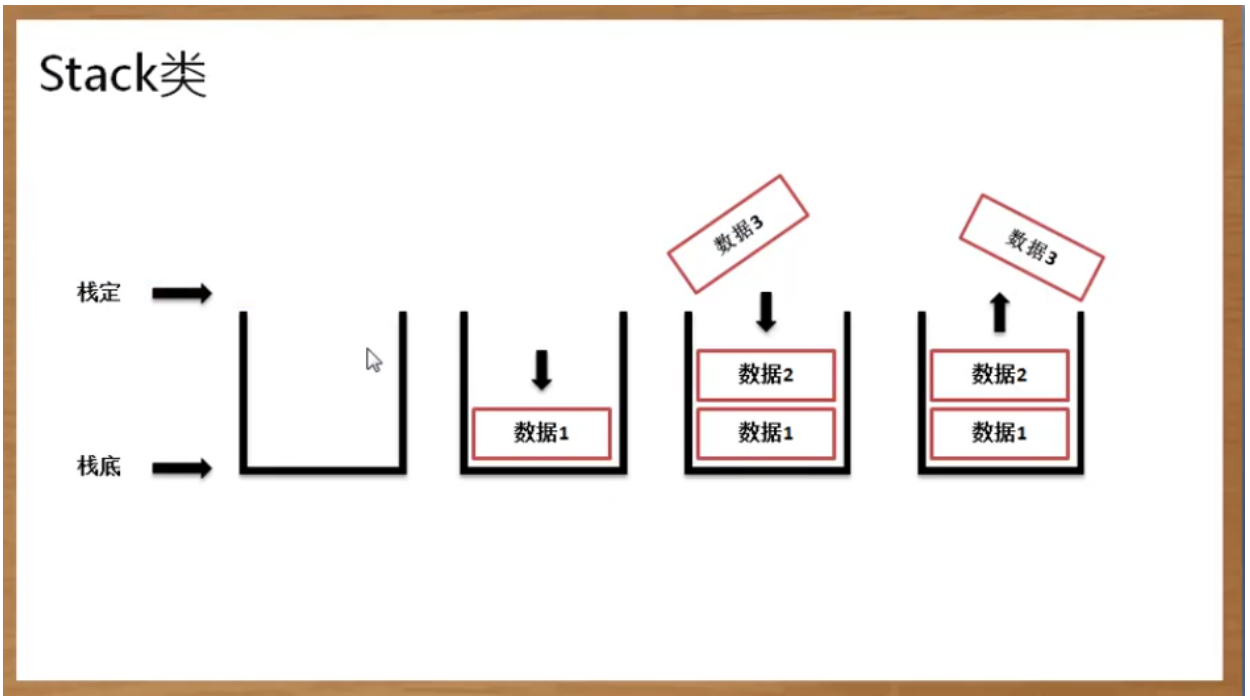


博客： https://www.cnblogs.com/HOsystem/p/14116443.html

2、具体内容

栈是一种先进后出的数据结构。例如：在文本编辑器上都有撤销功能，那么每次使用的时候你都会发现，最后一次的编辑操作永远是最先撤销，那么这个功能就是利用栈来实现的，栈的基本操作形式如下。



在Java程序里面使用stack来描述栈的操作，这个类定义如下：

```
public class Stack<E>  extends Vector<E>
```

可以发现stack是vector子类，但是它使用的并不是vector类之中提供的方法，而是采用如下的两个方法：

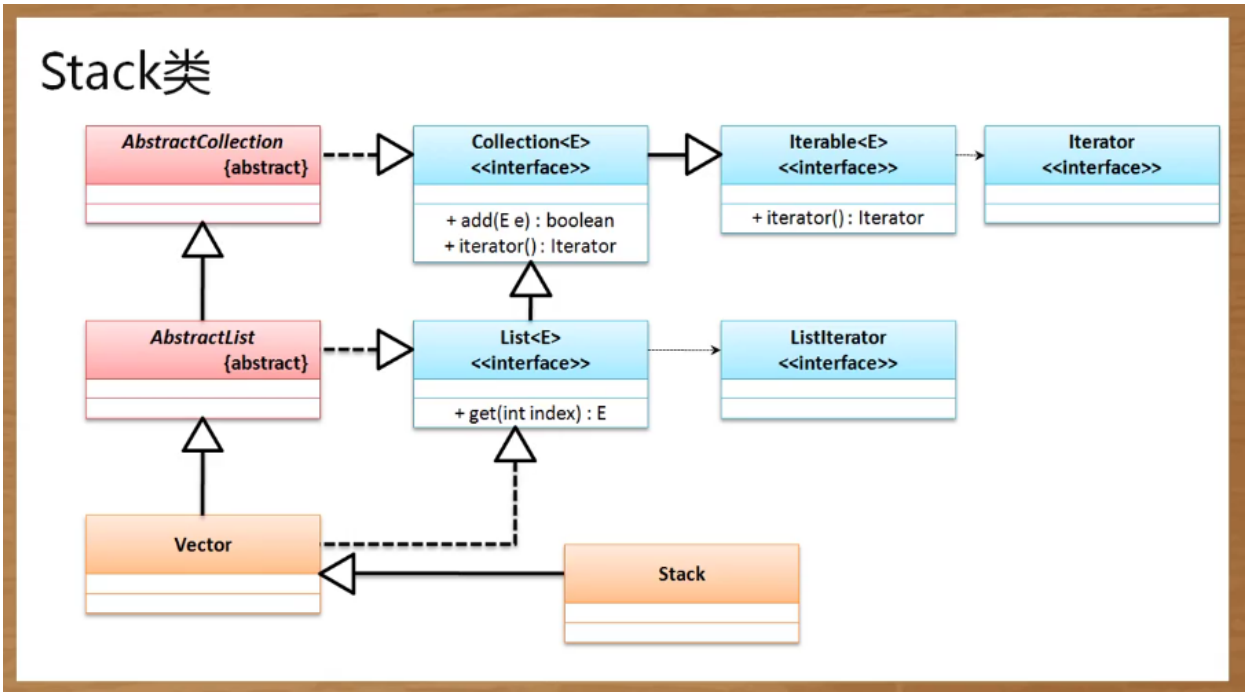
·入栈：public E push(E item);

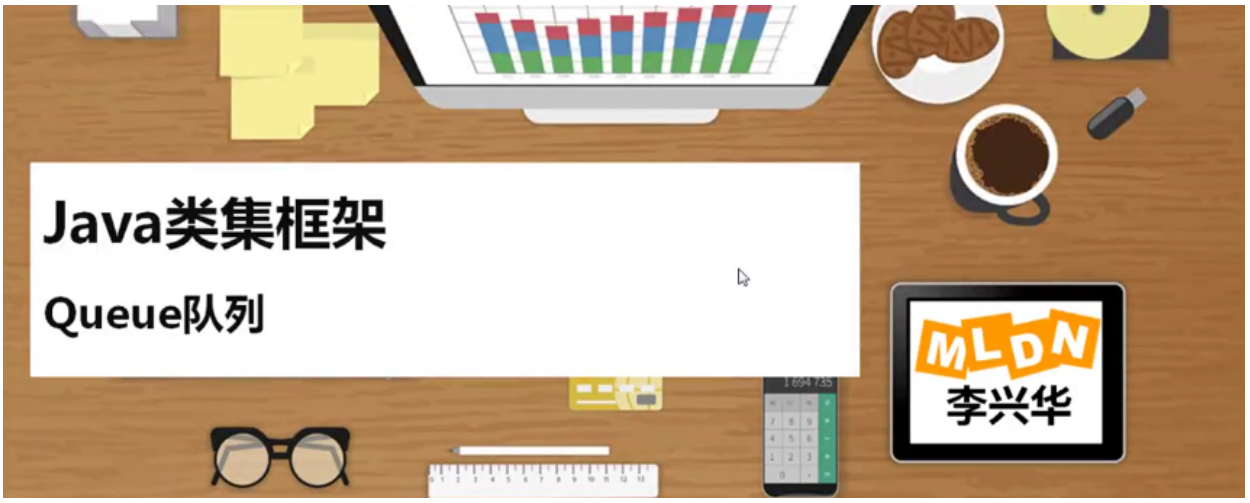
·出栈：public E pop();

范例： 实现栈的操作

```
package cn.mldn.demo;
import java.util.Stack;
public class JavaAPIDemo {
	public static void main(String[] args) throws Exception {
		Stack<String> all = new Stack<String>() ;
		all.push("A") ;
		all.push("B") ;
		all.push("C") ;
		System.out.println(all.pop());
		System.out.println(all.pop());
		System.out.println(all.pop());
		System.out.println(all.pop());  // 无数据、EmptyStackException
	}
}
```

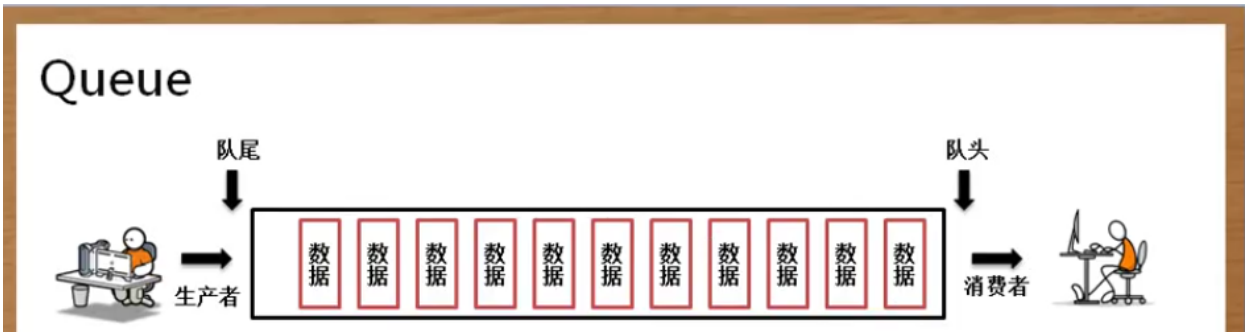
通过此时的操作可以发现，所有的保存之后将按照倒序的形式进行弹出，如果栈已经空了，则会抛出空栈异常。





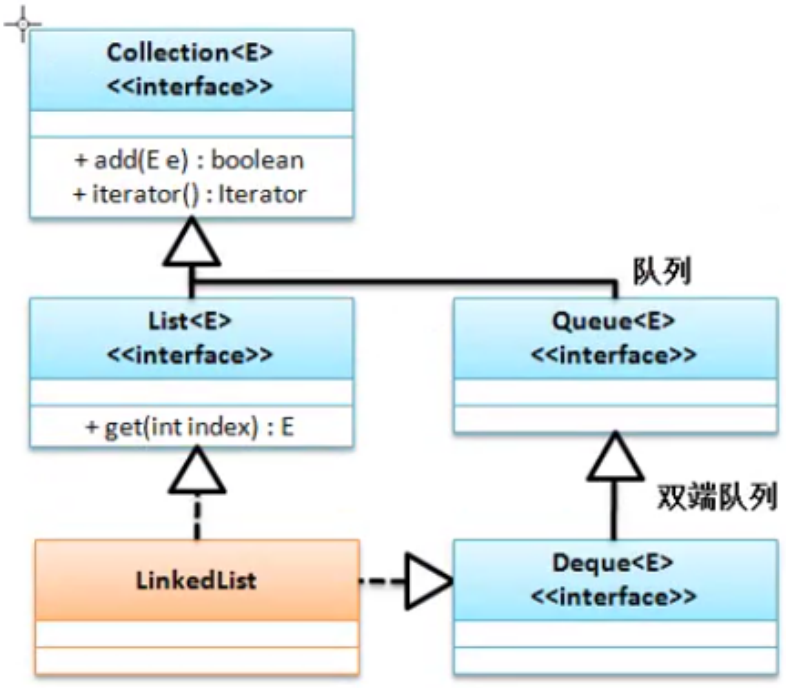
2、具体内容

queue描述的是一个队列，而队列的主要特点是实现先进先出的操作形式。其基本的操作形式如下：



如果将队列应用在多线程的”生产者-消费者“的模型处理上，那么对于生产过快的情况下就没有必要等待消费者获取数据了，可以将所有的内容直接保存在队列之中，队列的实现可以使用LinkedList子类来完成，观察这个类的定义：

Queue



队列的使用主要依靠Queue接口之中提供的方法来处理，提供有如下的方法：

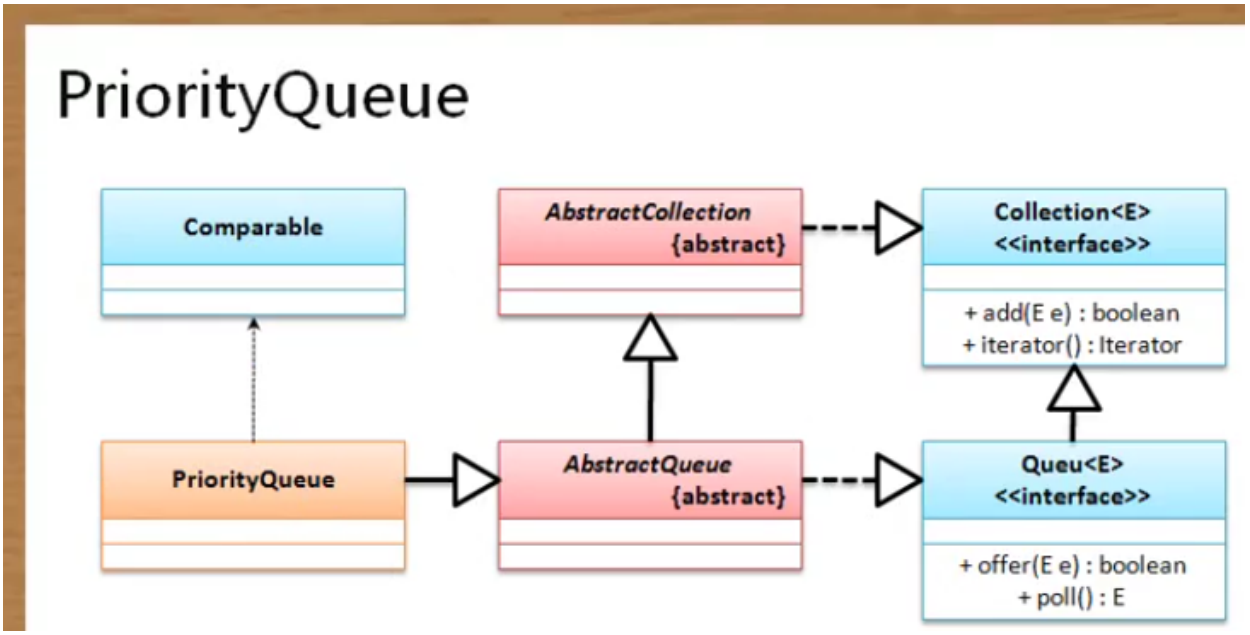
·向队列中追加数据：public boolean offer(E e)，可以直接使用add()方法；

·通过队列获取数据：public E poll()，弹出后删除数据；

范例：实现队列操作

```
package cn.mldn.demo;
import java.util.LinkedList;
import java.util.Queue;
public class JavaAPIDemo {
	public static void main(String[] args) throws Exception {
		Queue<String> queue = new LinkedList<String>() ;
		queue.offer("X") ;	// 追加队列数据，通过队尾追加
		queue.offer("A") ;	// 追加队列数据，通过队尾追加
		queue.offer("Z") ;	// 追加队列数据，通过队尾追加
		System.out.println(queue.poll()); // 弹出数据、X
		System.out.println(queue.poll()); // 弹出数据、A
		System.out.println(queue.poll()); // 弹出数据、Z
	}
}
```

除了LinkedList子类之外，还有一个优先级队列的概念，可以使用PriorityQueue实现优先级队列（比较功能）



public class PriorityQueue<E> extends AbstractQueue<E> implements Serializable

范例：使用优先级队列

```
package cn.mldn.demo;
import java.util.PriorityQueue;
import java.util.Queue;
public class JavaAPIDemo {
	public static void main(String[] args) throws Exception {
		Queue<String> queue = new PriorityQueue<String>() ;
		queue.offer("X") ;	// 追加队列数据，通过队尾追加
		queue.offer("A") ;	// 追加队列数据，通过队尾追加
		queue.offer("Z") ;	// 追加队列数据，通过队尾追加
```

```
            System.out.println(queue.poll()); // 弹出数据、X
            System.out.println(queue.poll()); // 弹出数据、A
            System.out.println(queue.poll()); // 弹出数据、Z
        }
}
```

对于队列的使用原则也是需要根据实际的项目环境来决定的。

2、具体内容

在之前讲解国际化程序的时候讲解过资源文件(*.properties),那么这类文件的存储结构是按照“key=value”的形式存储的，而这种结构的保存形式与Map集合很相似的，但是唯一的区别在于其所保存的内容只能够是字符串，所以为了可以方便的描述属性的定义，java.util包里面提供有Properties类型，此类是Hashtable的子类。

```
public class Properties extends Hashtable<Object,Object>
```

可以发现在继承Hashtable的时候为hashtable中定义的泛型为Object，properties是不需要操作泛型的，因为它可以操作的类型只能是String类型，在properties之中如果要想实现属性的操作可以采用如下的方法：

No	方法名称	类型	描述
01	public Object setProperty(String key,String value)	普通	设置属性
02	public String getProperty(String key)	普通	取得属性,key不存在返回null
03	public String getProperty(String key,String defaultValue)	普通	取得属性,不存在返回默认值
04	public void store(OutputStream out,String comments)throws IOException	普通	输出属性内容
05	public void load(InputStream inStream) throws IOException	普通	通过输入流读取属性内容

范例：观察属性的设置和取得

```
package cn.mldn.demo;
```

```
import java.util.Properties;
public class JavaAPIDemo {
	public static void main(String[] args) throws Exception {
		Properties prop = new Properties() ;
		// 设置的内容只允许是字符串
		prop.setProperty("mldn", "www.mldn.cn") ;
		prop.setProperty("mldnjava", "www.mldnjava.cn") ;
		System.out.println(prop.getProperty("mldn"));
		System.out.println(prop.getProperty("sina","NoFound"));
		System.out.println(prop.getProperty("sina"));
	}
}
```

通过代码可以发现Properties里面可以像Map集合那样进行内容的设置与获取，但是唯一的差别是它只能够操作String类型，另外需要注意的是，之所以会提供有properties来还有一个最重要的功能是它可以通过输出流输出属性，也可以使用输入流读取属性内容。

范例：将属性内容保存在文件之中

```
package cn.mldn.demo;
import java.util.Properties;
public class JavaAPIDemo {
	public static void main(String[] args) throws Exception {
		Properties prop = new Properties() ;
		// 设置的内容只允许是字符串
		prop.setProperty("mldn", "www.mldn.cn") ;
		prop.setProperty("mldnjava", "www.mldnjava.cn") ;
		prop.setProperty("beijing","北京");
		prop.store(new FileOutputStream(new File("D:"+ File.separator +
"info.properties")),"中文的注释是看不见的-english");
	}
}
```

通过程序的执行可以发现，的确可以实现资源文件的输入处理，但是如果输出的是中文则自动帮助用户进行处理。

范例：读取资源文件

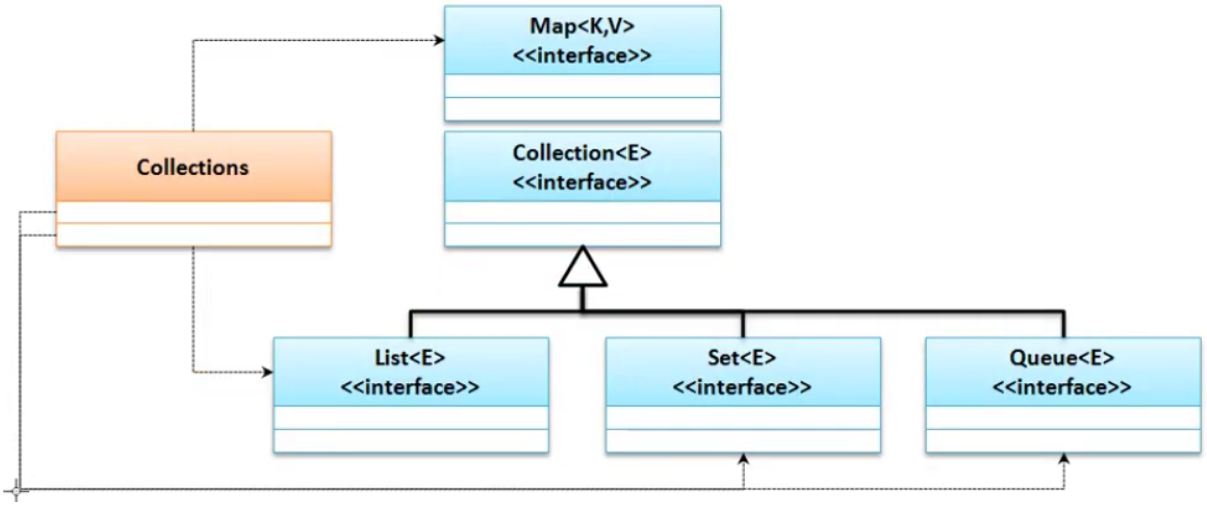
```
package cn.mldn.demo;
import java.io.File;
import java.io.FileInputStream;
import java.util.Properties;
public class JavaAPIDemo {
	public static void main(String[] args) throws Exception {
		Properties prop = new Properties() ;
		prop.load(new FileInputStream(new File("D:" + File.separator +
"info.properties")));
		System.out.println(prop.getProperty("mldn"));
	}
}
```

使用properties类型的最大特点是可以进行资源内容的输入与输出的处理操作，但是在实际的开发之中properties往往用于读取配置资源的信息，这一点主要是在标准设计之中做

程序初始化准备的时候使用。

2、具体内容

Collections是java提供的一组集合数据的操作工具类，也就是说利用它可以实现各个集合的操作。



范例：使用Collections操作List集合

```
package cn.mldn.demo;
import java.util.ArrayList;
import java.util.Collections;
import java.util.List;
public class JavaAPIDemo {
	public static void main(String[] args) throws Exception {
		List<String> all = new ArrayList<String>();
		Collections.addAll(all, "Hello", "World", "MLDN");
		System.out.println(all);
	}
}
```

范例：数据的反转

```
package cn.mldn.demo;
import java.util.ArrayList;
```

```
import java.util.Collections;
import java.util.List;
public class JavaAPIDemo {
	public static void main(String[] args) throws Exception {
		List<String> all = new ArrayList<String>();
		Collections.addAll(all, "Hello", "World", "MLDN");
		Collections.reverse(all);
		System.out.println(all);
	}
}
```

范例：使用二分查找

```
package cn.mldn.demo;
import java.util.ArrayList;
import java.util.Collections;
import java.util.List;
public class JavaAPIDemo {
	public static void main(String[] args) throws Exception {
		List<String> all = new ArrayList<String>();
		Collections.addAll(all, "Hello", "World", "MLDN");
		Collections.sort(all) ; // 先进行排序处理
		System.out.println(all);
		System.out.println(Collections.binarySearch(all, "MLDN"));
	}
}
```

大部分情况下对于集合的使用可能没有那么多复杂要求，更多的情况下就是利用集合保存数据要么进行输出要么进行查询。

面试题：请解释Collections与collection的区别?

·Collection是集合接口，允许保存单值对象；

·Collections是集合操作的工具类；