Module 5-I/O流

1. 需要掌握的类是File, FileReader, BufferedReader, FileWriter, BufferedWriter, PrintWriter
2. 对于File类：
   a) 它是文件和文件夹的一种抽象，并不代表文件系统中一个真正的文件。
   b) 它不用于读写文件，而是用来创建、删除搜索文件或者文件夹。
   c) 只有调用createNewFile()才会创建真实文件，用exists()来判定文件是否存在
   d) File类的构造函数用String作为参数，没有直接用一个File对象创建另一个file对象的构造函数
3. FileWriter/BufferedWriter
   a) FileWriter的构造函数由File或者String对象作为参数
   b) BufferedWriter由Writer对象做参数创建
   c) BufferedWriter比FileWriter方法更多更方便；常用BufferedWriter来封装FileWriter
4. 通常创建的顺序是

```
File f = new File(“someFile”);
FileReader fr = new FileReader(f);
//或者FileReader fr = new FileReader(“somefile”);
BufferedReader br = new BufferedReader(fr);
```

5. PrintWriter
   a) 从Java 5开始,可以由File或者String对象创建,它还有个println()方法
   b) 要注意PrintWriter自有的一些方法，如print(), format(), println()等等

java.io Class	Extends From	Key Constructor(s) Arguments	Key Methods
File	Object	File, String String String, String	createNewFile() delete() exists() isDirectory() isFile() list() mkdir() renameTo()
FileWriter	Writer	File String	close() flush() write()
BufferedWriter	Writer	Writer	close() flush() newLine() write()
PrintWriter	Writer	File (as of Java 5) String (as of Java 5) OutputStream Writer	close() flush() format()*, printf()* print(), println() write()
FileReader	Reader	File String	read()
BufferedReader	Reader	Reader	read() readLine()
			*Discussed later

6.DataInputSream 和 DataOutputStream 不考

7.序列化和反序列化

a）FileXXXStream 是低层次的 Stream，通常用 ObjectXXXStream 来封装它们，这个类似于 FileReader 和 BufferedReader 的关系。

b）声明为 transient 的变量和 static 变量不会被 serialize。

c）要 serialize 的类声明实现 Serializable 接口，这个是个空接口(market interface)。

d）在 serialize 的时候，虚拟机会自动做深度拷贝(注意拷贝的是对象内容而不是其引用)。如果类的定义中有引用类型的实例变量(instance variable)，那么这个类也应该实现 Serializable 接口，除非这个引用没有指向 new 出的对象，否则会抛出 NotSerializableException，注意不是编译错误。

例如：

```
class Leg {}
class Dog implements Serializable {
    Leg leg;
}
//没问题
class Leg {}
class Dog implements Serializable {
Leg leg = new Leg();
}
//异常！
```

e) 对于Has-a关系:

i.有时候我们需要自己实现writeObject()和readObject(),其方法签名如下：

```
private void writeObject(ObjectOutputStream os) throws IOException
private void readObject(ObjectInputStream is) throws IOException, ClassNotFoundException
```

ii.注意这两个方法一般情况下是要一起覆盖的，如果只覆盖其中的一个方法的时候，那么这个方法体里面必须要调用defaultXXXXObject()

iii.如果一起覆盖的话，可以在这两个方法体里面调用writeXXX(),或者readXXX()（请参见TestDefaultRead.java）

f) 对于Is-a关系:

i.如果父类实现了Serializable接口，子类自动实现。

ii.如果子类实现了Serializable, 父类可以不实现。如父类没有实现Serializable,那么当de-serialize的时候,父类的**默认构造函数**会被调用(子类的构造函数不被调用)。

iii.如果子类实现了Serializable而父类没有实现，父类必须要有**默认构造函数**，否则虽然不会产生编译错误，但是运行时会抛出异常。(请参见TestSerializable.java)

Module 6-字符串、格式化以及封装类

1. 对于 String
   a) 首先要记住的是 String 对象是不可改变的，但是指向 String 对象的引用可以改变。所以在使用 String 类中的方法时，要把返回值赋值给新的引用，否则原来的 String 对象将丢失。
   b) String 类是 final 的，不能够被继承。
   c) 得到String对象的长度使用的是length()方法,String类并没有 length 这样的属性。
2. 对于 StringBuilder 和 StringBuffer
   a) StringBuffer 是 synchronized，因此比 StringBuilder 慢。
   b) 这两个类的对象都是可以改变的，他们的方法会改变对象本身，无需再赋值。
3. 记住 StringBuffer.delete(start, end)删除的是从索引 start 到 end-1 的字符串
4. 两个实体类Date, Locale, 3个抽象类Calendar, DateFormat,NumberFormat

Class	Key Instance Creation Options
util.Date	new Date(); new Date(long millisecondsSince010170);
util.Calendar	Calendar.getInstance(); Calendar.getInstance(Locale);

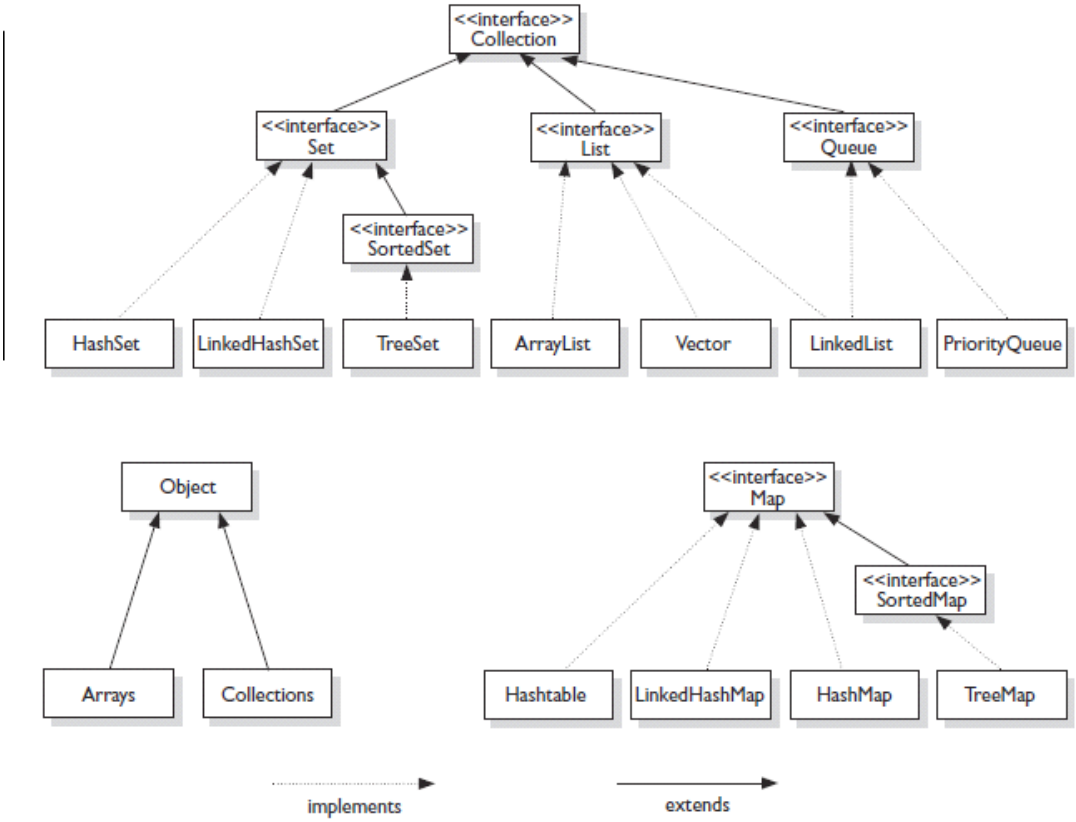
util.Locale	Locale.getDefault();
	new Locale(String language);
	new Locale(String language, String country);

text.DateFormat	DateFormat.getInstance(); DateFormat.getDateInstance(); DateFormat.getDateInstance(style); DateFormat.getDateInstance(style, Locale);
text.NumberFormat	NumberFormat.getInstance() NumberFormat.getInstance(Locale) NumberFormat.getNumberInstance() NumberFormat.getNumberInstance(Locale) NumberFormat.getCurrencyInstance() NumberFormat.getCurrencyInstance(Locale)

5. Date类的大部分方法都已经deprecated, 它由一个long来存储自1997年1月1日以来经过的秒数, Calendar有比它更丰富的计算日期的方法，但是其他类DateFormat, NumberFormat都要用Date来存储日期。
6. 关于DateFormat和NumberFormat:
   a) 它们是抽象类，所以只能由静态工厂方法获得(具体见构造函数表)
   b) 静态工厂方法可以有两个参数style，locale
   c) DateFormat和NumberFormat都有parse和format方法来进行类型转换

7.Locale构造函数有两个参数，一个是language, 一个是country
8.主要是考这几个类之间的关系，构造函数(是否工厂方法)和常用方法
9.关于正则表达式(regular expressions/ regex):
a) 正则表达式用于创建一个模式(pattern)或者元字符(metacharacters)，以此来匹配要搜索的字符串等内容
b) 正则表达式中:
i. \d 表示数字
ii. \s 表示空格符
iii. \w 表示一个字符(字母，下划线，数字)
iv. . 表示任何字符
c) 量化符(quantifiers)
i. ? 表示 0 次或 1 次
ii. * 表示 0 次或多次
iii. + 表示一次或多次
d) 使用 java.util.regex.Pattern 和 Matcher 来模式匹配
i.
```
Pattern p = Pattern.compile(args[0]);
Matcher m = p.matcher(args[1]);
while(boolean b = m.find()) {
    System.out.println(m.start() + " " + m.group())
}
```
ii. 注意方法compile(regex), matcher(source)。
iii. Matcher类的几个方法，find()从source开始找到下一个匹配是否存在，start()返回上一个匹配的索引，group()返回上一个匹配的字符串
10.对于 tokenizing
a) 用分隔符 delimiter 将字符串分开的过程叫做 tokenizing，分号的字符串叫做 tokens
b) 有两种方式来 tokenizing, 一种是 String.split, 它的方法签名是：
public String[] split(String regex)
c) 第二种方式是用 Scanner 类
i. 默认的 delimiter 是空格，可以用 useDelimiter()来设定 delimitor
ii. 它用循环来一个个得出下一个 token，所以可以随时退出。hasNextXxx()测试下一个 token 的值但并不返回下一个 token。
iii. nextXxx()返回得到下一个 token 的值,并且移动到下一个 token
iv. 注意如果没有匹配那么 next()返回的是整个 source
11.对于格式化输出：
有两个方法要考，format()和 printf()，它们两个功能相同，其中：
%b---------boolean(null 为负)
%c---------char
%d---------integer
%f---------float
%s---------string
如果参数类型不匹配，会抛出异常 IllegalFormatConversionException

Module 7-集合及泛型



1. == 和 equals(): ==判断的是两个引用是不是指向同一个对象，而 equals()判断两个对象是否真正意义上相等
2. equals()和 hashCode():
   a) 如果两个对象 equals()，那么他们的 hashCode()也相等
   b) 但是如果 hashCode()相等，它们不一定 equals()
3. HashMap, HashSet, Hashtable, LinkedHashMap 和 LinkedHashSet 都用 hash
4. 注意区别题目中 legal 和 appropiate，返回常数的 hashCode()是 legal 的，但是效率不高
5. 有三个 sorted 类：TreeSet, TreeMap 和 PriorityQueue
6. 各个容器类的特点：
   a) ArrayList 迭代速度快，随机访问速度快，允许 null
   b) LinkedList 添加删除速度快， 适合在尾部添加元素
   c) HashSet 快速访问，没有重复，但是无法迭代
   d) LinkedHashSet 按插入顺序迭代
   e) HashMap 允许一个 null key, 多个 null value
   f) Hashtable 是 HashMap 的对应类，不允许 null, 线程安全
   g) LinkedHashMap 快速迭代，也允许一个 null key 和多个 null value，按照插入顺序或者最后访问顺序迭代
   h) PriorityQueue 按照元素的优先级排序, sorted
7. 一些容器类的特有方法：
   a) Iterator: 很多 List 和 Set 都可以迭代，传统的方法是用 Iterator:

Iterator it = list.iterator(); hasNext()方法返回是否还有下一个，并不向后移动 iterator。 next()返回下一个元素并且向后移动 iterator。

b) Map 的 key 必须实现 equals()和 hashCode()， HashMap 往 Map 中 put()时候，会先判断 hasCode()是否相等，相等的话再用 equals()判断是否重复，如果重复就刷掉前面的值。所以如果类的实现没有 override hashCode()，那么即使出现重复也可以放进去。

c) Queue 接口下面有两个类，一个是 LinkedList，另外一个是 PriorityQueue，它们都有一些特有的方法，offer()，peek()，poll()。offer()向容器中添加元素，peek()返回队列头部，但是不从队列中删除；poll()返回并删除。

8. Arrays 和 Collections 类：
   a) 用它查找时 binarySearch()之前一定要先 sort()过，而且要用同一个 Comparator
   b) 用 Arrays.asList()从一个数组创建一个 List 对象，注意这个 List 和数组是连在一起的，共同变化
   c) Collections.reverse()把 List 中的元素反过来，reverseOrder()返回一个和原来 Comparator 相反的 Comparator
   d) List 接口和 Set 接口都有一个 toArray()方法转化成数组
      i. Object[] toArray()
      ii. <T> T[] toArray(T[] a)
9. 同步与不同步(synchronized)
   a) 同步的: Hashtable, Vector
   b) 不同步的:HashMap, ArrayList
10. 关于 Comparable 和 Comparator
   a) 如果要用 Comparable，该类就要 implements Comparable，实现 int compareTo(Object)方法，然后直接用 Collections.sort(list)就可以了，Arrays 相同
   b) 如果要用 Comparator，就要实现 compare(Object, Object)方法。然后用 Colletions.sort(list, comparator);

java.lang.Comparable	java.util.Comparator
`int objOne.compareTo(objTwo)`	`int compare(objOne, objTwo)`
Returns negative if objOne < objTwo zero if objOne == objTwo positive if objOne > objTwo	Same as Comparable
You must modify the class whose instances you want to sort.	You build a class separate from the class whose instances you want to sort.
Only **one** sort sequence can be created	**Many** sort sequences can be created
Implemented frequently in the API by: String, Wrapper classes, Date, Calendar...	Meant to be implemented to sort instances of third-party classes.

11. 以下所有容器类都要掌握

Class	Map	Set	List	Ordered	Sorted
HashMap	x			No	No
HashTable	x			No	No
TreeMap	x			Sorted	By *natural order* or custom comparison rules
LinkedHashMap	x			By insertion order or last access order	No
HashSet		x		No	No
TreeSet		x		Sorted	By *natural order* or custom comparison rules
LinkedHashSet		x		By insertion order	No
ArrayList			x	By index	No
Vector			x	By index	No
LinkedList			x	By index	No
PriorityQueue				Sorted	By to-do order

13. Generics 是编译时刻的类型安全，运行时对于虚拟机来说没有变化
14. 当把一个使用泛型的 collection 放进一个非泛型的方法中时，编译器就无法阻止该方法往里面放不兼容类型的对象。虽然编译可以通过，但会产生一个编译器警告，告诉你这样做可能有危险。
15. 要注意题目中 compile without error 不等于 compile without warning。
16. 多态赋值不适用于泛型类参数，如：
    a) List<Animal> aList = new ArrayList<Animal>(); 可以
    b) List<Animal> aList = new ArrayList<Dog>(); 不可以
17. 可以在参数中使用?, 如：
    a) void addD(List<? extends Dog>){} 可以传入 Dog 及其子类对象
    b) 注意 extends 关键字这里既适用于类也适用于接口
    c) 使用了 extends 的容器只能被访问不能被修改。如果要修改的话，需要改成 void addD(List<? super Dog>){}
18. 可以用来声明自己的泛型类和方法，两个例子如下:
    a) class Sample<T>{}
    b) public <T> void samleMethod(T t){} 注意这里 T 不是返回值
19. 注意? 是用在参数中的，而 T 是声明泛型方法用的，不要混淆：
    a) public void addD(List<? extends Dog>){}
    b) public <T extends Animal> void addD(T t){}

Module 8-多线程

1. 要注意题目是否正确实现了 run()方法,经常会缺少 public 关键字作为陷阱。正确的 run()方法名是：public void run()
2. 注意构造新的 thread 对象时，总是要用 new Thread。如果是 myRunnable 实现了 Runnable，那么用 new Thread(myRunnable)
3. Thread 类对象的 start()方法只能调用一次，否则会抛出 java.lang.IllegalThreadException，值得注意的是一个线程抛出了异常，其他线程仍然会执行。
4. sleep 方法：
   public **static** void sleep(long millis) throws InterruptedException
要注意以下几点：
   a) 它是**静态(static)**方法,会使当前正在执行的线程休眠**最少** millis 毫秒
   b) 要用 try/catch 或者声明 throws InterruptedException
   c) 它不会丢失对象锁标记
5. yield 方法：
   public static void yield() throws InterruptedException
   a) 它是**静态(static)**方法，将当前正在运行的线程变为 runnable 状态
   b) 没有保证说它不会从线程池中被再次选中变为 running 状态
   c) 要用 try/catch 或者声明 throws InterruptedException
6. join 方法：
   public final void join () throws InterruptedException
   a) 调用 a.join()会使当前正在运行的线程等待，先执行完其它线程的方法，然后再往下继续执行
   b) 要用 try/catch 或者声明 throws InterruptedException
   c) 它有两个重载：join(long millis)和 join(long millis, int nanos)
7. 关于 synchronized 修饰符
   a) synchronized 可以用来修饰方法和语句块，用于语句块的时候要锁某个对象
   b) 不能用 synchronized 来锁一个 primitive 类型变量(int/long/boolean 等)
   c) 每个 Object 都有一个锁标记，非静态方法可以锁对象
   d) 每个类都有一个 java.land.Class(比如 MyClass.class),静态方法可以用它来 lock
8. 关于 wait()/notify()/notifyAll()
   a) 它们都是 Object 类的方法，不属于 Thread 类
   b) 调用 wait()将立即释放对象的锁标记,而 notify()并不会立即释放锁标记，要等 synchronized 语句块结束才会释放
   c) 三个方法都只能在 synchronized 的上下文中使用，而且要有当前的 object 才行
   d) notify()无法指定唤醒某个线程，这个是由虚拟机决定的