# 第十章 面向对象思考

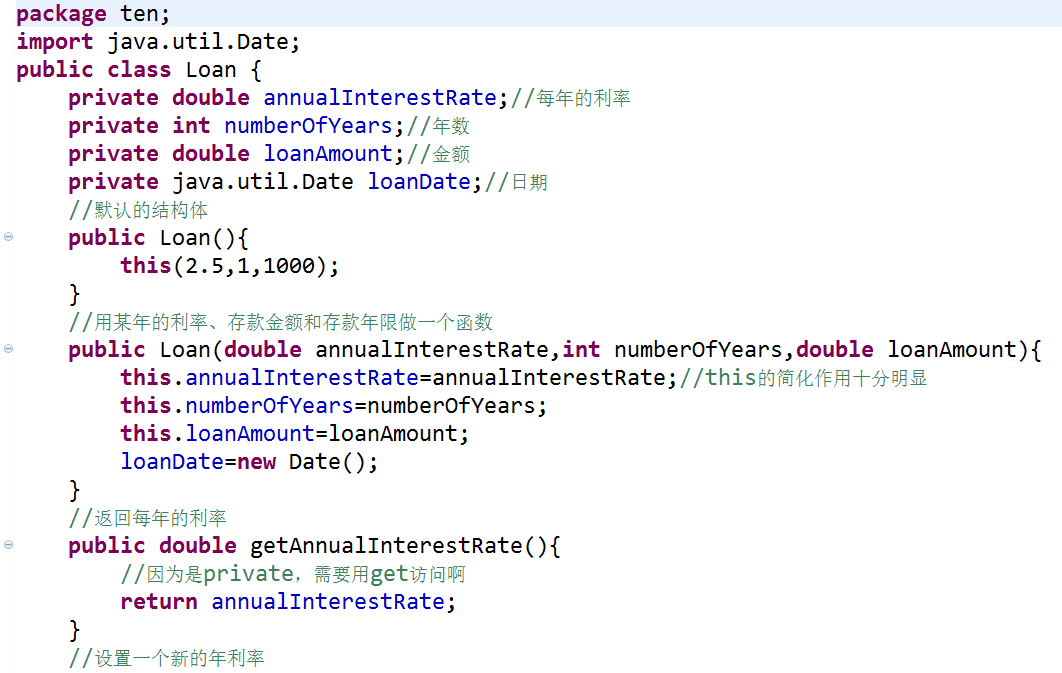
## 10.1引言

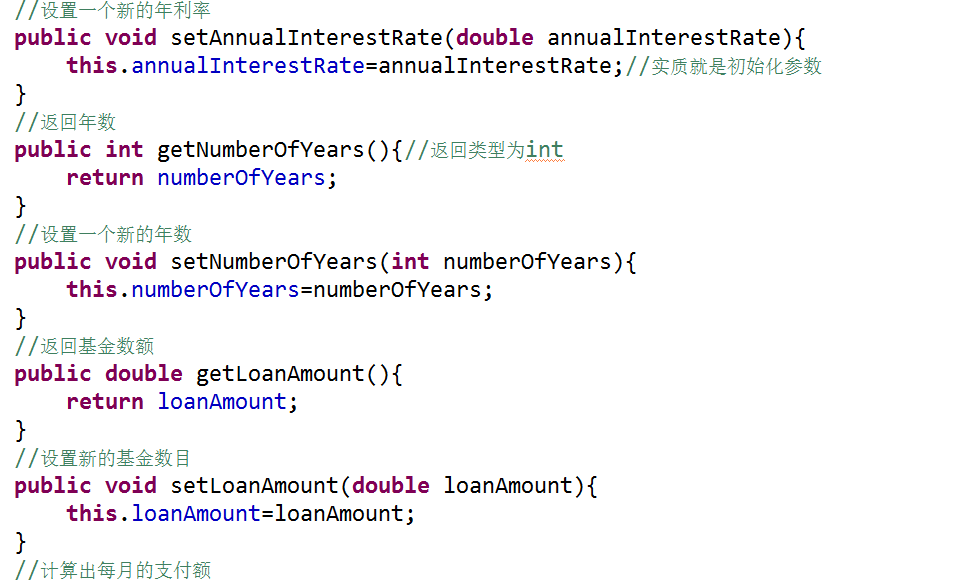
1.本章重点：“类设计”和探索“面向过程编程和面向对象编程”的不同。

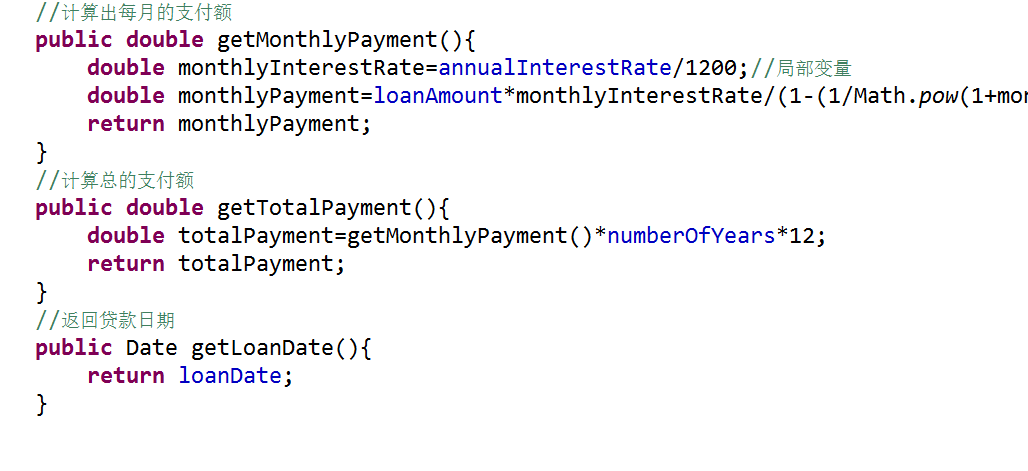
## 10.2类的抽象和封装

1.要点提示：类的抽象是指将 类的实现 和 类的使用 分离开，实现细节被封装并且对用户隐藏，这被称为类的封装。例如：可以创建一个Circle对象，并且可以在不知道面积如何计算出来的情况下，求出这个圆的面积。由于这个原因，类类也成为“抽象数据类型”。以下是一个贷款的示例程序，主类实现了对其他类的调用：

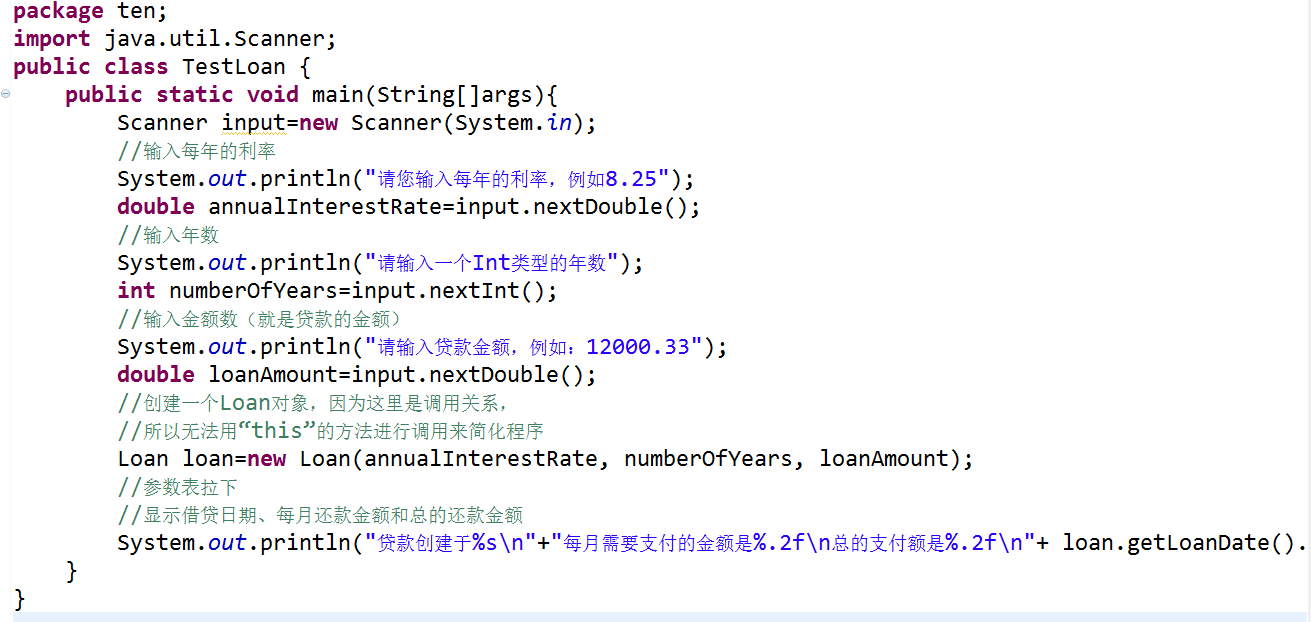
辅助类







主函数



## 10.3面向对象的思考

1.要点提示：面向过程的范式重点在于设计方法。面向对象的范式将数据和方法耦合在一起构成对象。使用面向对象范式的软件设计重点在对象以及对对象的操作上。

2.面向过程和面向对象方法的实例比较：

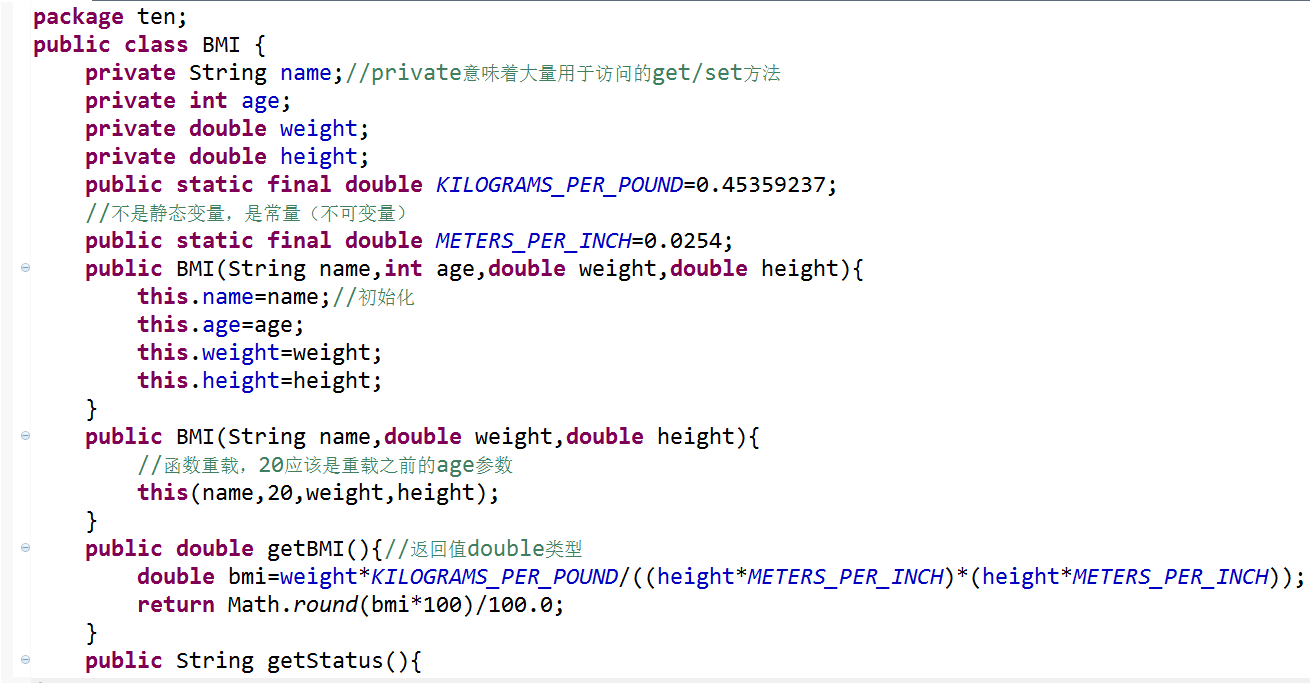
定义一个静态方法计算身体质量指数，如下

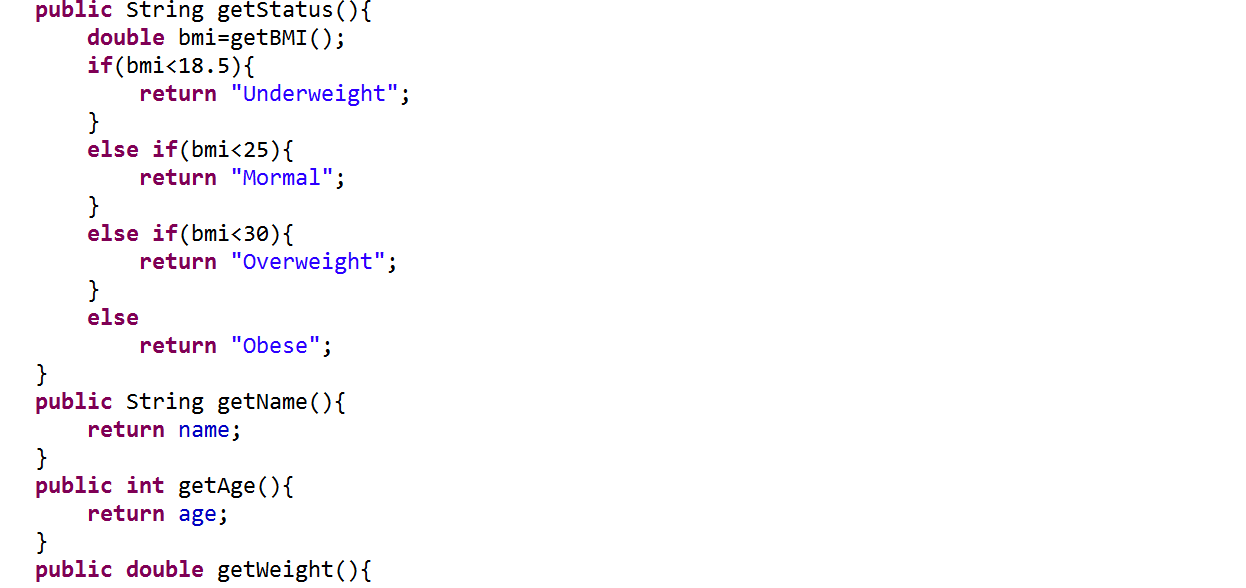
Public static double getBMI(double weight , double height)

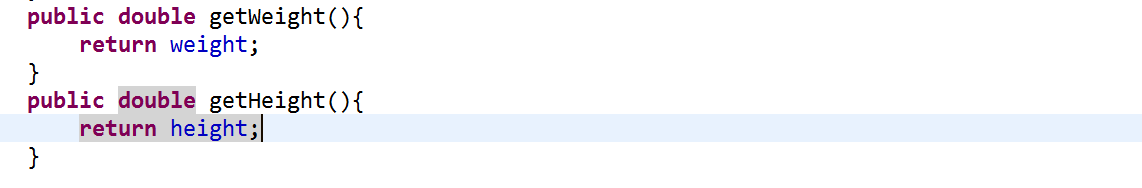
这个方法对于计算体重和身高的身体质量指数是很有用的。但是，它是有局限性的。假设需要将体重和身高同一个人的名字与出生日期相关联，虽然可以分别声明几个变量来存储这些值，但是这些值不是紧密耦合在一起的。将它们耦合在一起的理想方法就是创建一个包含它们的对象。因为这些值都被绑定到单独的对象上，所以它们应该存储在实例数据域中。

3.上面描述的方法具体实现如下：

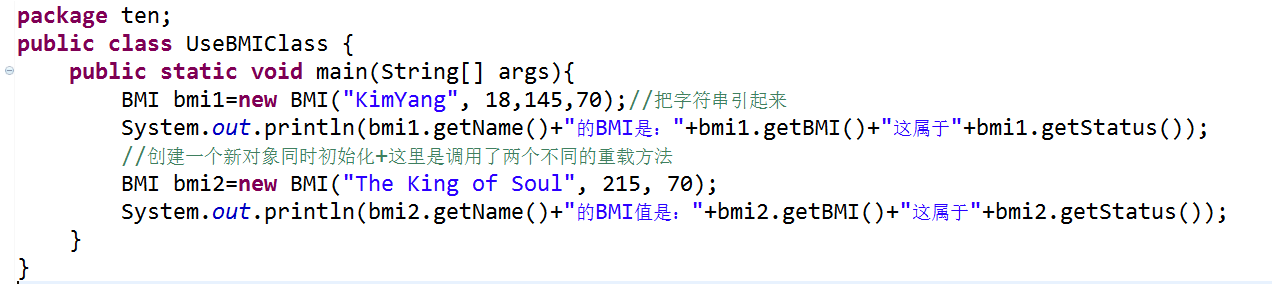
辅助类：







主函数：



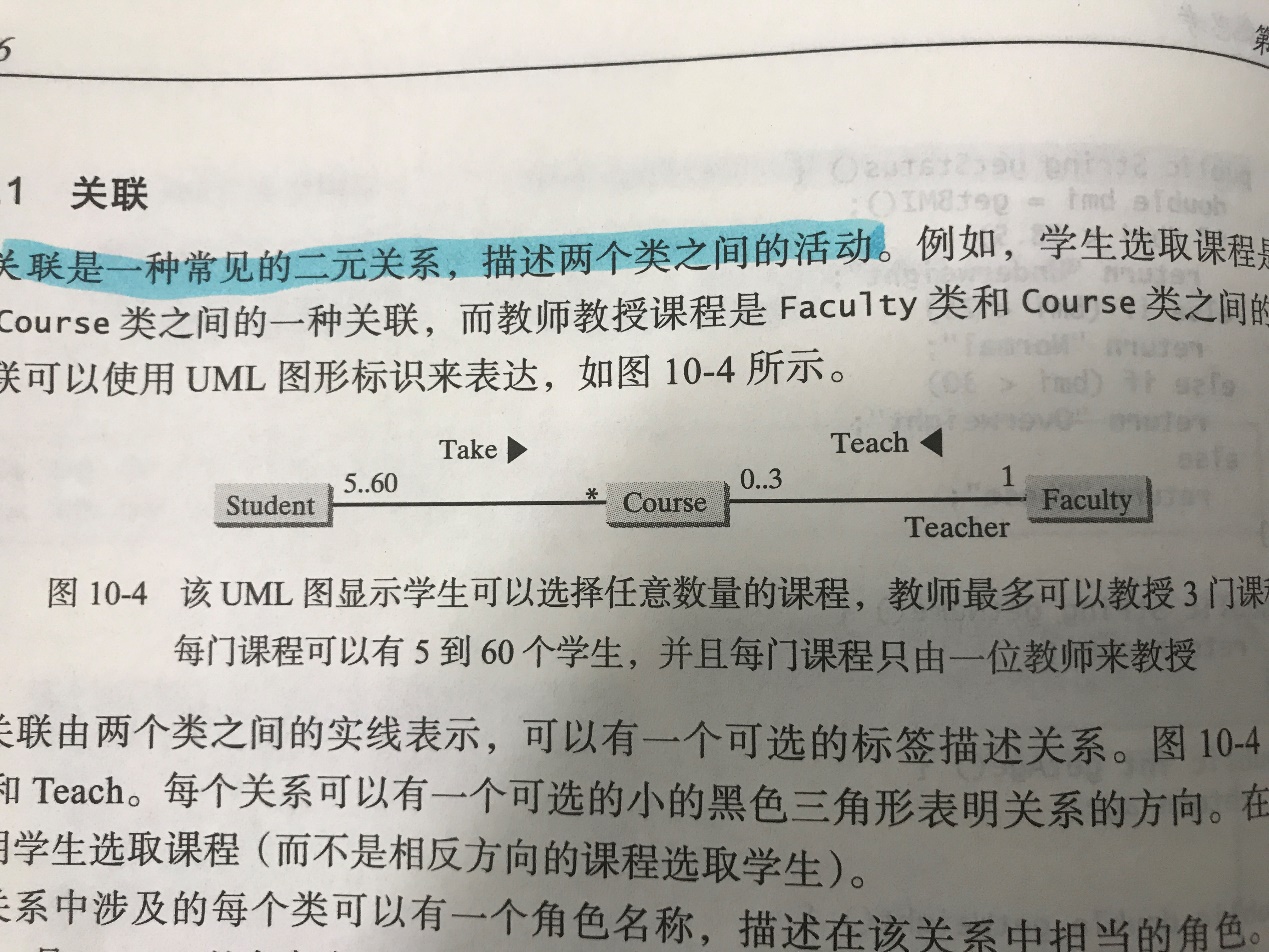
4.Java程序设计涉及对对象的思考，一个Java程序可以看作是一个相互操作的对象的集合。

## 10.4类的关系

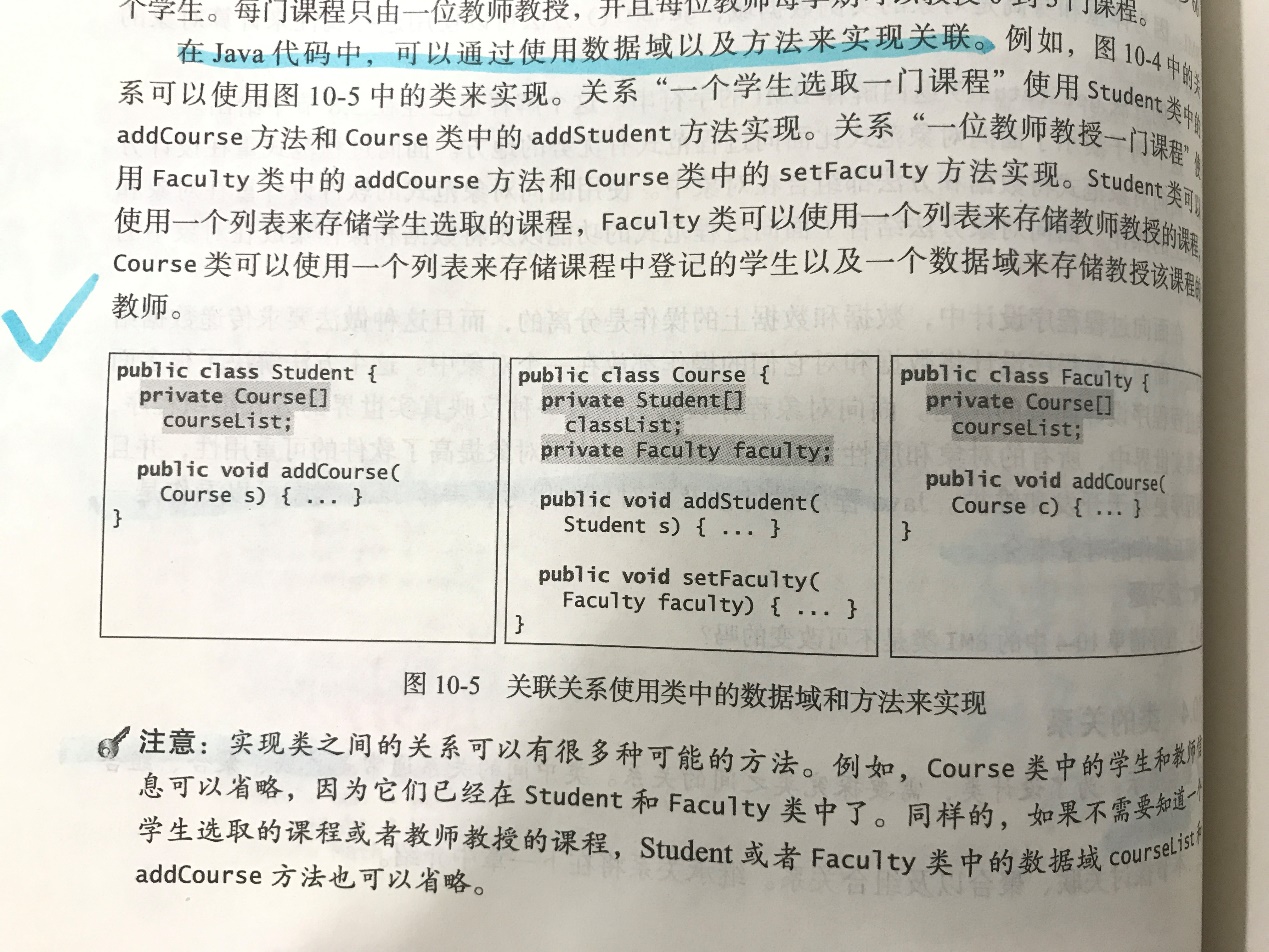
1.要点提示：类之间的关系通常是：关联、聚合、组合以及继承。

### 10.4.1关联

1.关联是一种常见的二元关系，描述两个类之间的活动。



2.在Java代码中，可以通过使用数据域以及方法来实现关联。



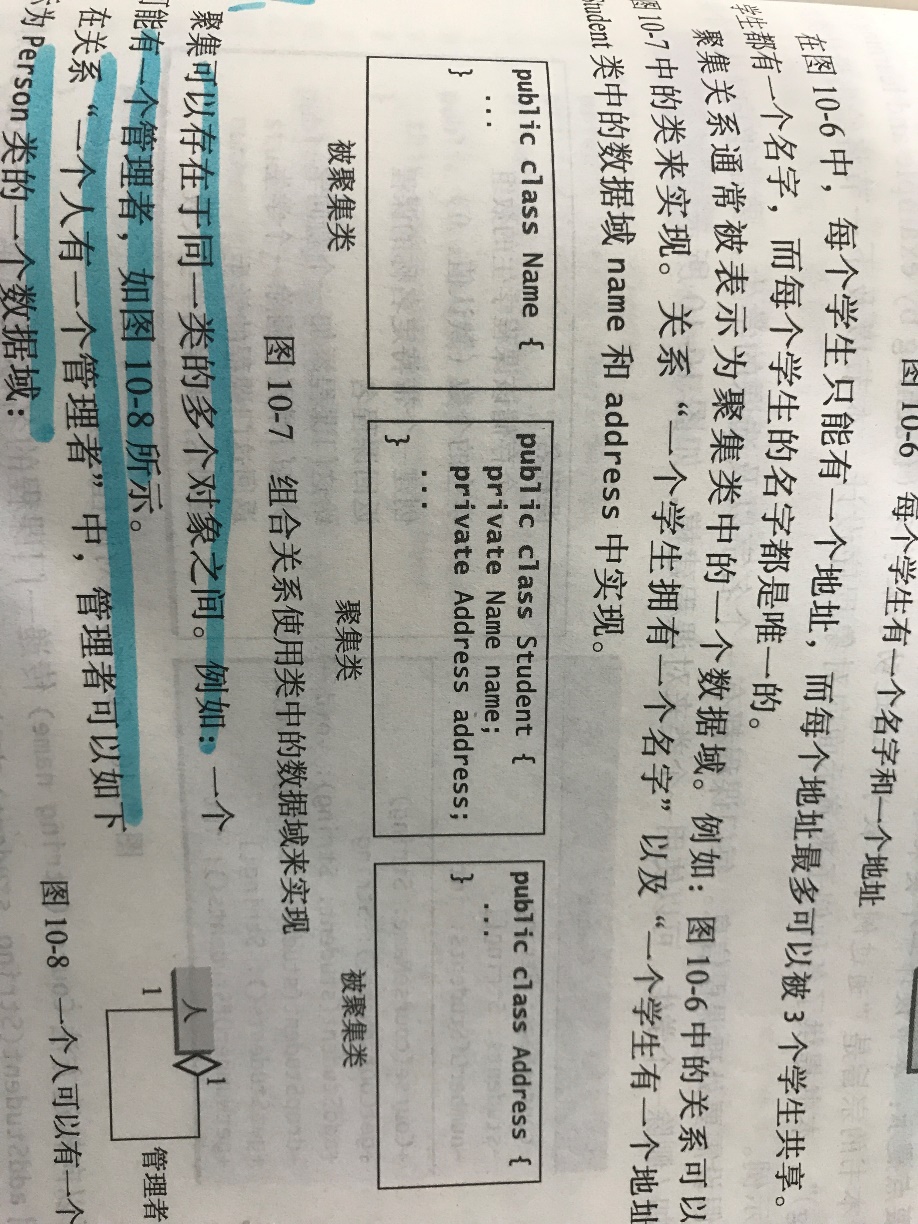
### 10.4.2聚集和组合

1.聚集是关联的一种特殊形式，代表了两个对象之间的归属关系。所有者对象称为狙击对象，它的类称为聚集类。而从属对象称为被聚集对象，它的类称为被聚集类。

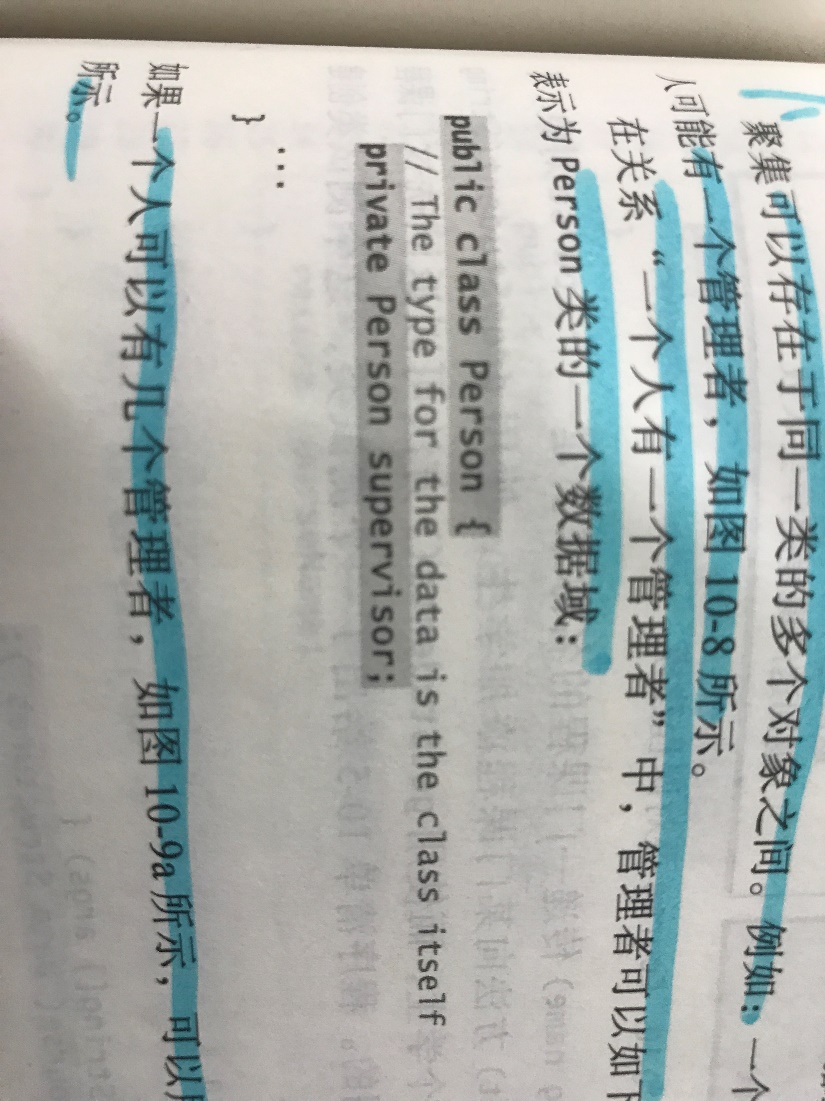
2.如果一个对象只属于一个聚集对象，那么它和聚集对象之间的关系就称为“组合”。例如：“一个学生有一个名字”就是学生类student和姓名类name之间的一个组合关系；

而“一个学生有一个地址”则是学生类student和地址类address之间的一个聚合关系，因为一个地址可以被几个学生所共享。

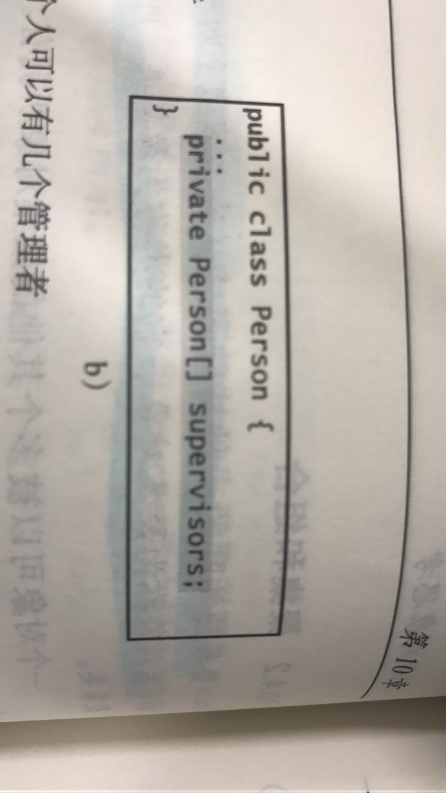
3.组合关系使用类中的数据域来实现：



4.聚集可以存在于同一类的多个对象之间。例如：一个人可能有一个管理者，数据域可以这么表示



如果一个人可以有多个管理者，可以用一个数组存储管理者，如下图所示：

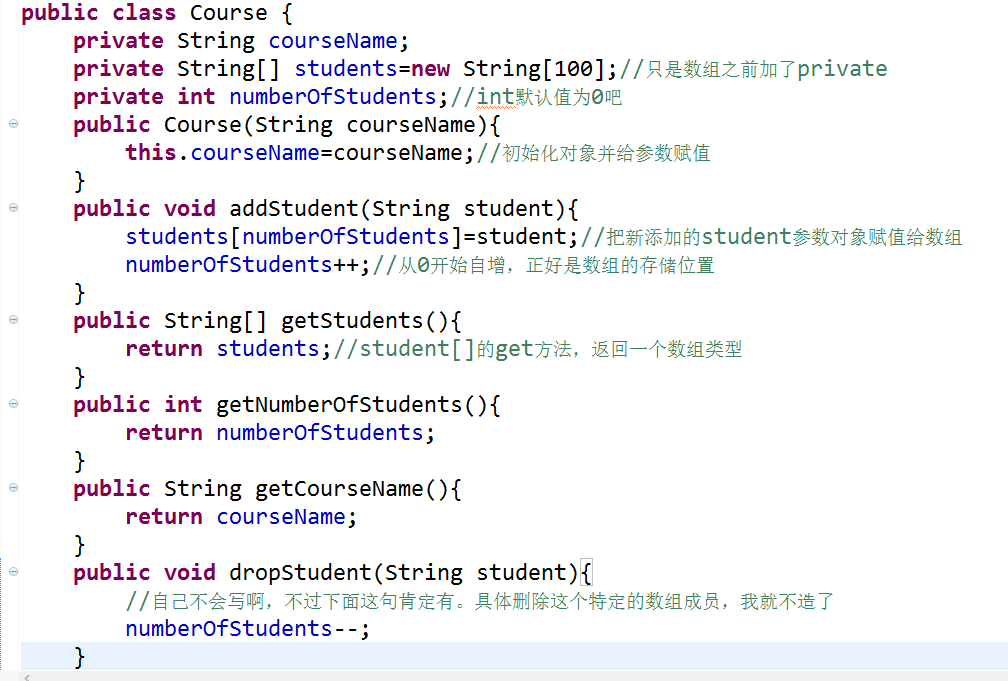


5.注意：由于聚集和组合关系都以同样的方式用类表示，我们不区分它们，将两者都称为组合。

## 10.5示例学习：设计Course类

1.要点提示：通过设计一个类来对课程建模。程序如下：

辅助类



主函数

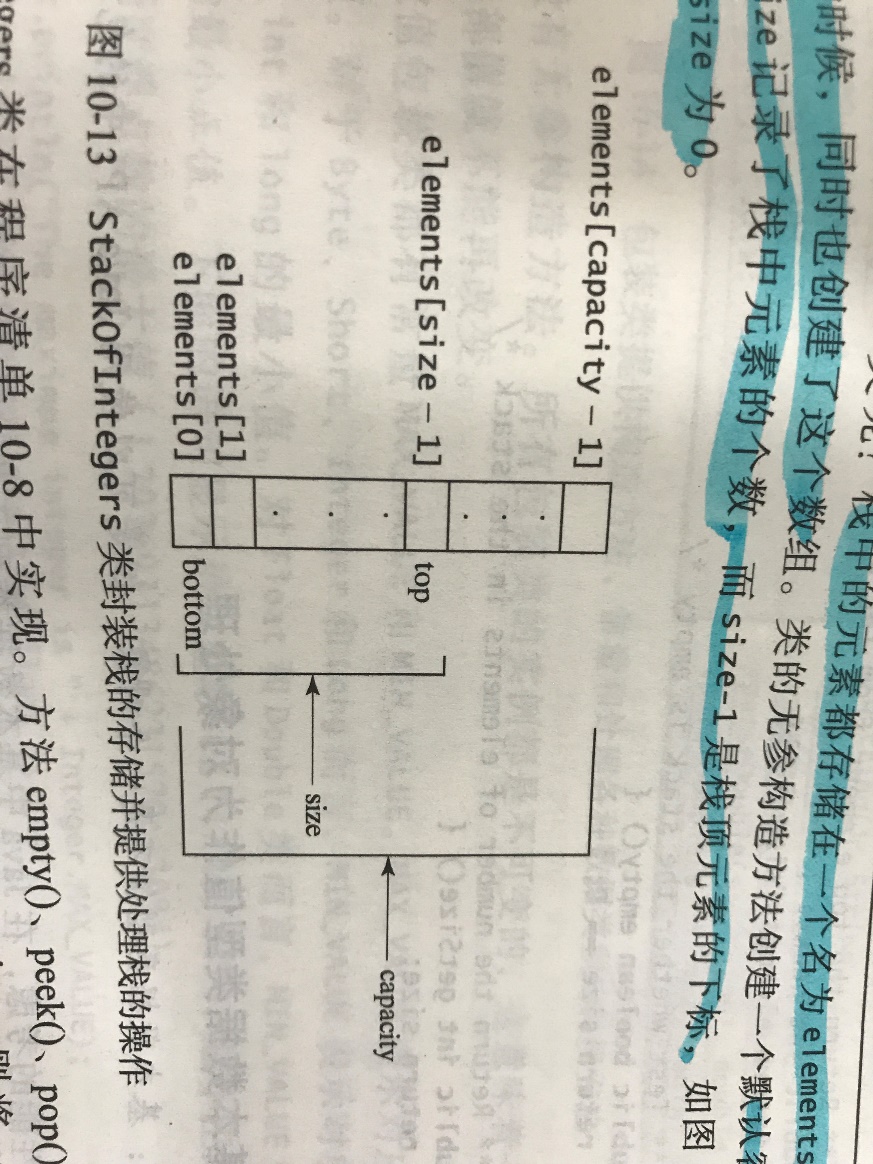


## 10.6示例学习：设计栈类

1.栈（stack）是一种“后进先出/先进后出”的数据结构，数据存放形式（自己想想就行了，这么简单）。

2．栈有很多应用。例如：编译器使用栈来处理方法调用。当调用某个方法时，方法的参数和局部变量都被压入栈中。当一个方法调用另一个方法时，新方法的参数和局部变量被压入栈中。当方法完成它的工作，返回他的调用者时，从栈中释放与它相关的空间。（疑惑？）

3.说明一：栈中元素都存储在一个名为elements的数组中。创建一个栈的时候，也创建了这个数组。我们构建一个默认容量为16的数组。变量size记录了栈中元素的个数，size-1是栈顶元素的下标。如图所示，对于空栈来说，size为0。



说明二：

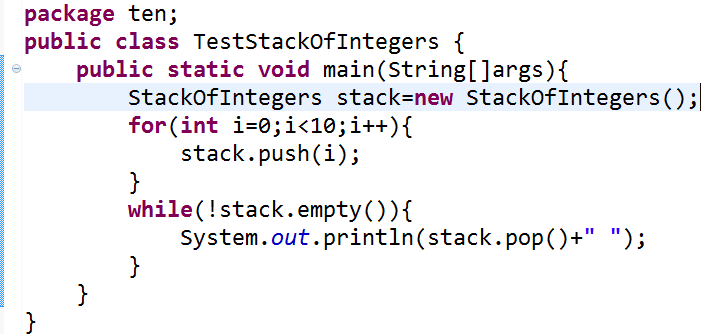
1. 如果size<capacity,则把value赋值给elements[size]。24行
2. 如果size>capacity(栈已满)，则创建一个容量为当前容量两倍的数组。 19行

将当前数组内容复制到新数组中。 20行

将新数组的引用赋值给栈中当前数组。 21行

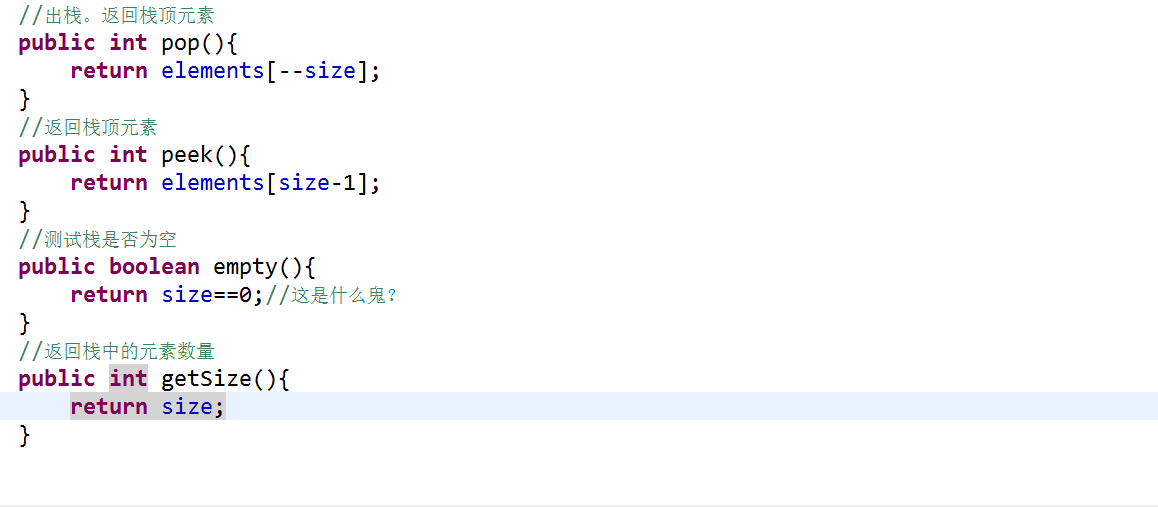
给新数组添加新值。 24行

主程序



实现程序





## 10.7将基本数据类型值作为对象处理（理论理解为主）

1.要点提示：基本数据类型值不是一个对象，但是可以使用Java API中的包装类包装成一个对象。

2.相关背景：出于性能的考虑，在Java中基本数据类型不作为对象使用。因为处理对象需要额外的系统开销，所以，如果将基本数据类型当作对象，就会给语言性能带来负面影响。然而，Java中的许多方法需要将对象作为参数。Java提供了一个简便的方法，即将基本数据类型并入对象或包装成对象（例如，将int包装成Integer类，将double包装成Double类，将char包装成character类）。通过使用包装类，可以将基本数据类型值作为对象处理。Java为基本数据类型提供了Boolean、Character、Double、Float、Byte、Short、Integer和Long等包装类。这些包装类都打包在Java.lang包里。Boolean类包装了布尔值true或者false。

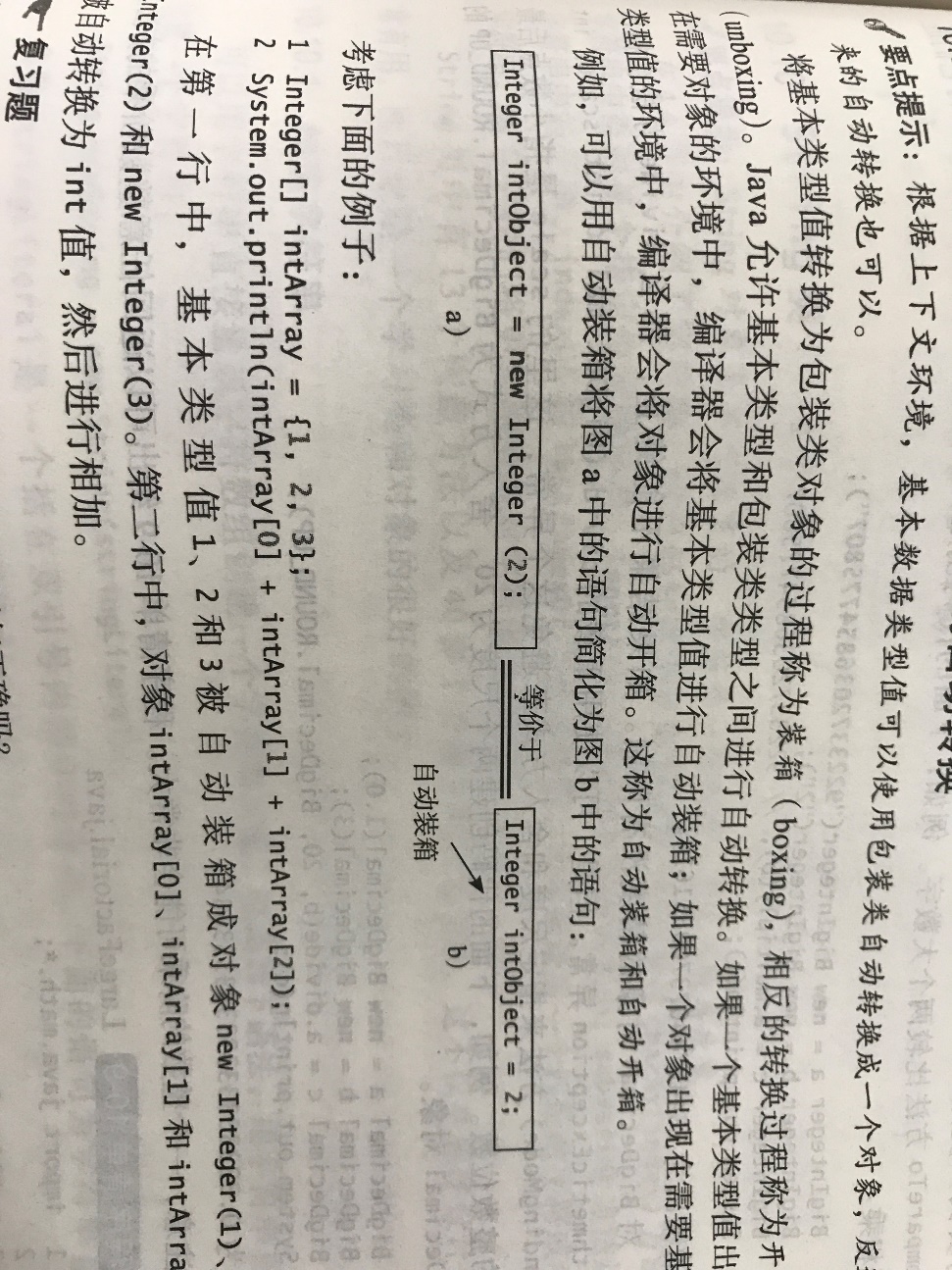
3.注意：大多数基本类型的包装类的名称与对应的基本数据类型名称一样，第一个字母要大写。Integer和Character例外。

## 10.8基本数据类型和包装类数据类型之间的自动转换

1.要点提示：根据上下文环境，基本数据类型值可以使用包装类自动转换成一个对象，反过来的转换也可以。

2.装箱和开箱：将基本类型值转化为包装类的对象称为装箱，相反的过程称为“开箱”。

3.自动装箱和自动开箱：Java允许基本类型和包装类类型之间进行转换。如果一个基本类型值出现在需要对象的环境中，编译器会将基本类型值进行自动装箱；如果一个对象出现在需要基本类型的环境中，编译器会将对象自动开箱。这称为“自动装箱”和“自动开箱”。



## 10.9 BigInteger和BigDecimal类

1.要点提示：BigInteger类和BigDecimal类可以用于表示任意大小和精度的整数或者十进制数。如果要进行非常大的数的计算或者高精度浮点值的计算，可以使用java.math包中的BigInteger和BigDecimal类。

2.示例程序：一个整数的阶乘可能会非常大。以下程序给出了可以返回任意整数阶乘的方法。



## 10.10 String类

1.要点提示：String对象是不可改变的。字符串一旦建立，内容不能再改变。

2.首先说，字符串是对象。可以通过charAt（index）方法从字符串中得到某个特定位置的字符。Length方法返回字符串的大小。Substring方法返回字符串中的子字符串。IndexOf和lastIndexOf方法返回第一个或者最后一个匹配字符串或者子字符串。

3.String类中有13个构造方法以及40多个处理字符串的方法。这不仅在程序设计中非常有用，而且也是一个学习类和对象的很好例子。

### 10.10.1构造字符串

1.总结：可以用字符串直接量或字符数组创建一个字符串对象

①用字符串直接量创建一个字符串：

String message=new String（“welcome to Java”）；

②还可以用字符数组创建一个字符串：

Char[ ] charArray={‘G’,’O’,’D’};

String message=new String(charArray);

3.注意：String变量存储的是对String对象的引用，String对象里存储的才是字符串的值。

严格的讲，术语String变量、String对象和字符串值是不同的。但在大多数情况下，它们之间的区别是可以忽略的。为简单起见，术语字符串将经常被用于指String变量，String对象和字符串的值。

### 10.10.2不可变字符串与限定字符串

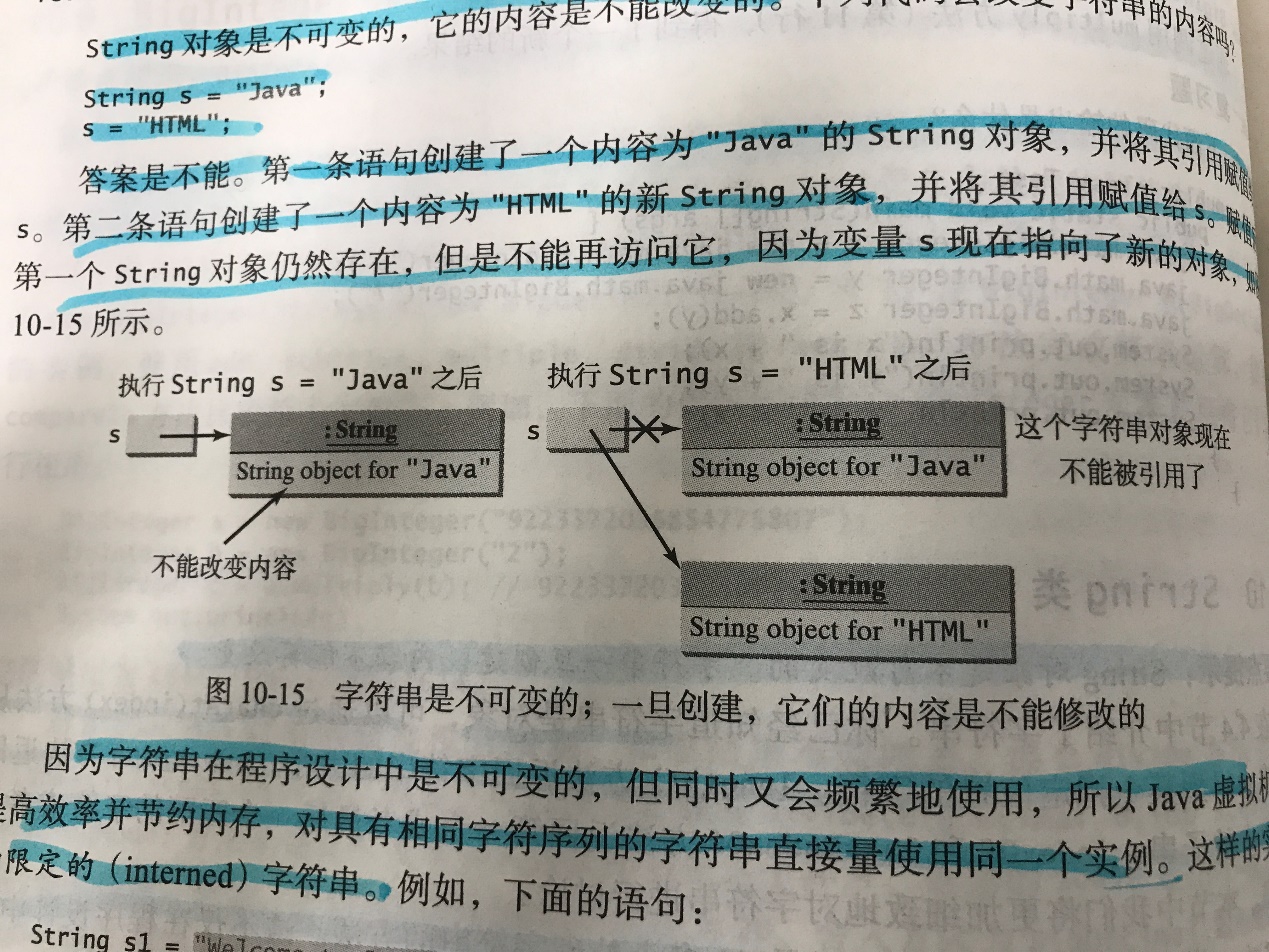
1.String对象是不可变的，它的内容是不能改变的。

2.下列代码会改变字符串的内容吗？

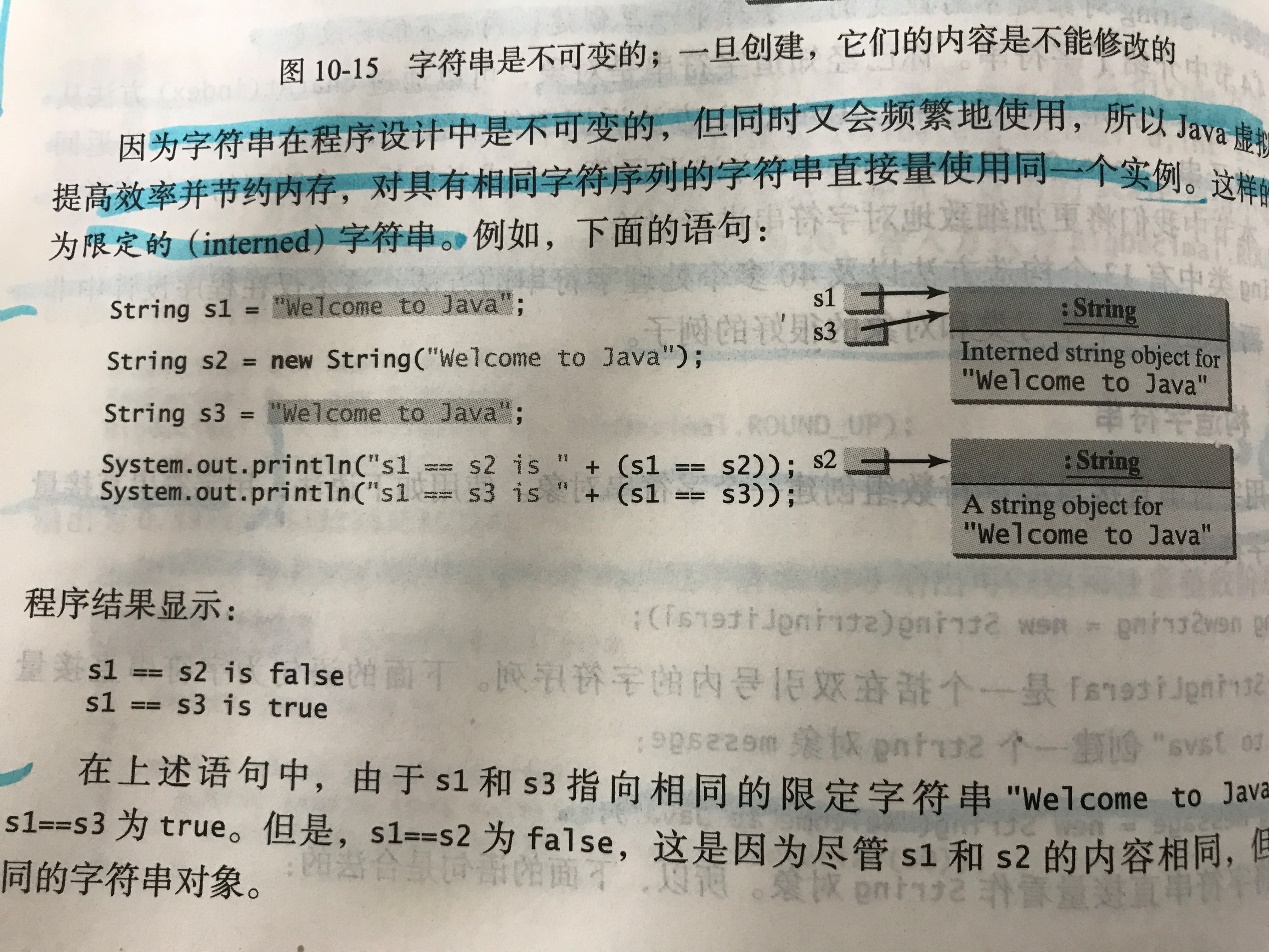
String s=”Java”;

S=”html”;

答案是不能。第一条语句创建了内容为“Java”的一个s对象，并且将其引用赋值给s。第二条语句创建了内容为“html”的对象，并且将其引用赋值给s。在这个过程中，赋值后的第一个String对象仍然存在，但是不能再访问它，因为变量s指向了新的对象，如图所示：

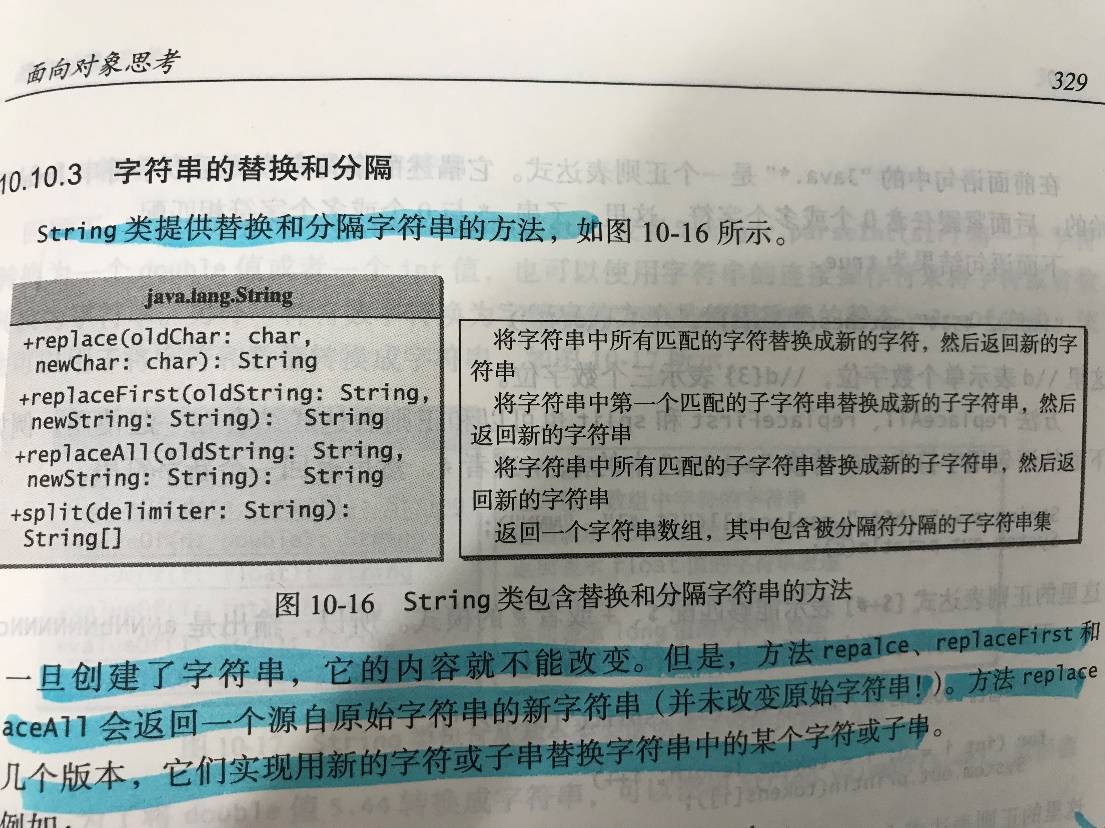


3.因为字符串在程序设计中是不可变的，但是同时又会频繁的使用，所以Java虚拟机为了提高效率并节约内存，对具有相同字符序列的字符串直接量使用同一个实例。这样的实例称为“限定的字符串”。如图所示：

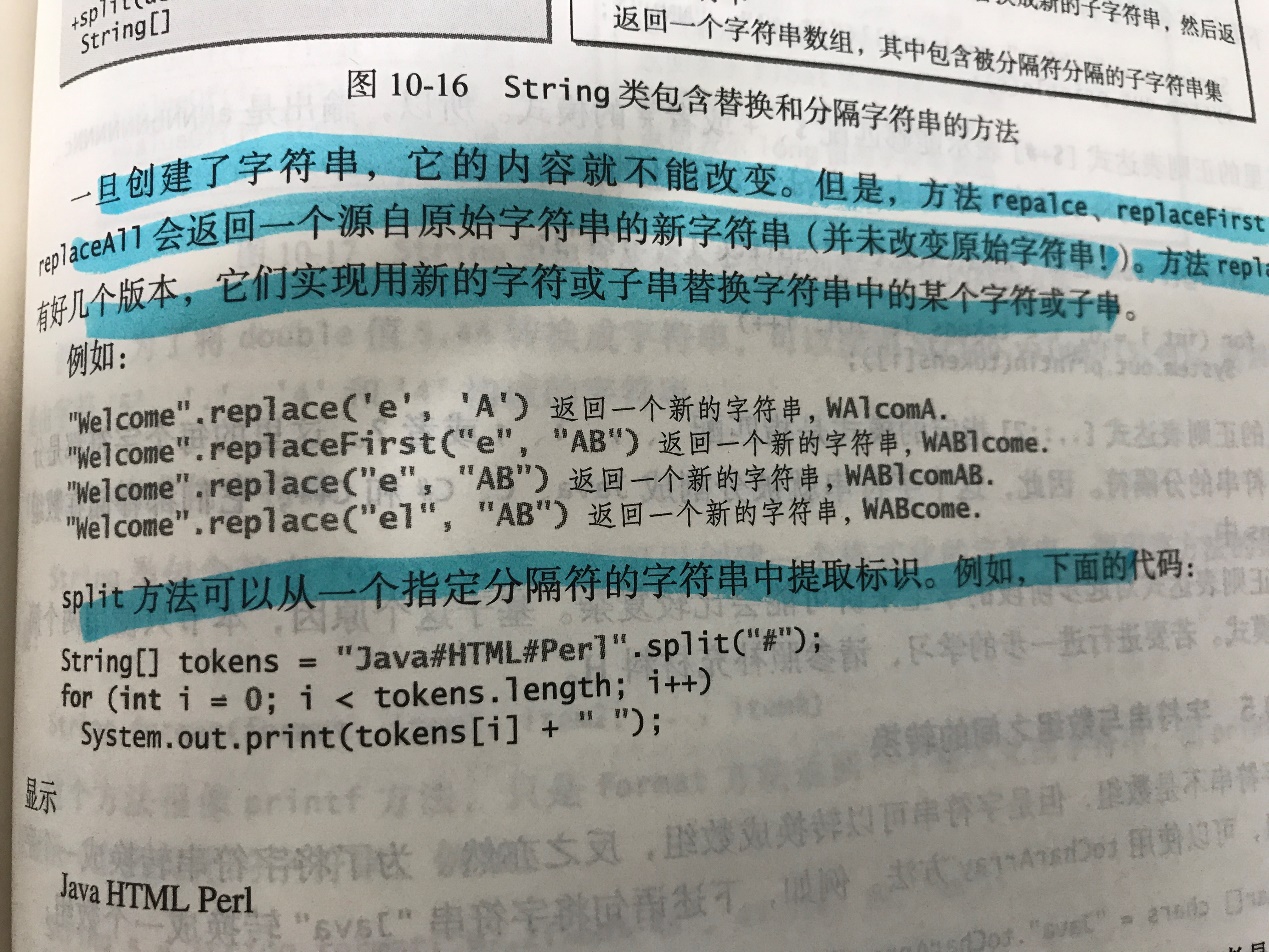


### 10.10.3字符串的替换和分隔

1.String类提供替换和分隔字符串的方法，如图所示：



2.一旦创建了字符串，它的内容就不能改变。但是，方法repalce、replaceFirst和replaceAll会返回一个源自原始字符串的新字符串（并未改变原始字符串！）。方法replace有好几个版本，它们实现用新的字符串或子串替换字符串中的某个字符或子串。



### 10.10.4依照模式匹配、替换和分隔

1.正则表达式是一个字符串，用于描述匹配一个字符串集的模式。可以通过制定某个模式来匹配、替换或分隔一个字符串。这是一种非常有用且功能强大的特性。

2.从String类中的matches方法开始。表面上看，matches方法和equals方法非常相似。例如，下面两条语句的值均为true：

“Java”.matches(“Java”);

“Java”.equals(“Java”);

但是，matches方法的功能更加强大。它不仅可以匹配定长的字符串，还能匹配一套遵从某种模式的字符串。例如，下面语句的结果均为true：

“Java is fun”matches（Java.\*）

### 10.10.5字符串与数组之间的转换

1.总结：字符串不是数组，但是字符串可以转换成数组，反之亦然。

2.字符串转换为数组：

为了将字符串转换成一个字符数组，可以使用toCharArry方法。例如，下述语句将字符串“Java”转换成一个数组：

Char[ ] chars=”Java”.toCharArray( );

因此，chars[0]是‘J’，chars[1]是‘a’，chars[2]是‘v’，chars[4]是‘a’。

3.将数组转换为字符串：

为了将一个字符数组转换成一个字符串，应该使用构造方法String（char[ ]）或者方法valueOf(char[ ])。

例如，下面的语句使用String构造方法由一个数组构成一个字符串：

String str=new String(new char[ ]{‘J’,’A’,’V’,’A’});

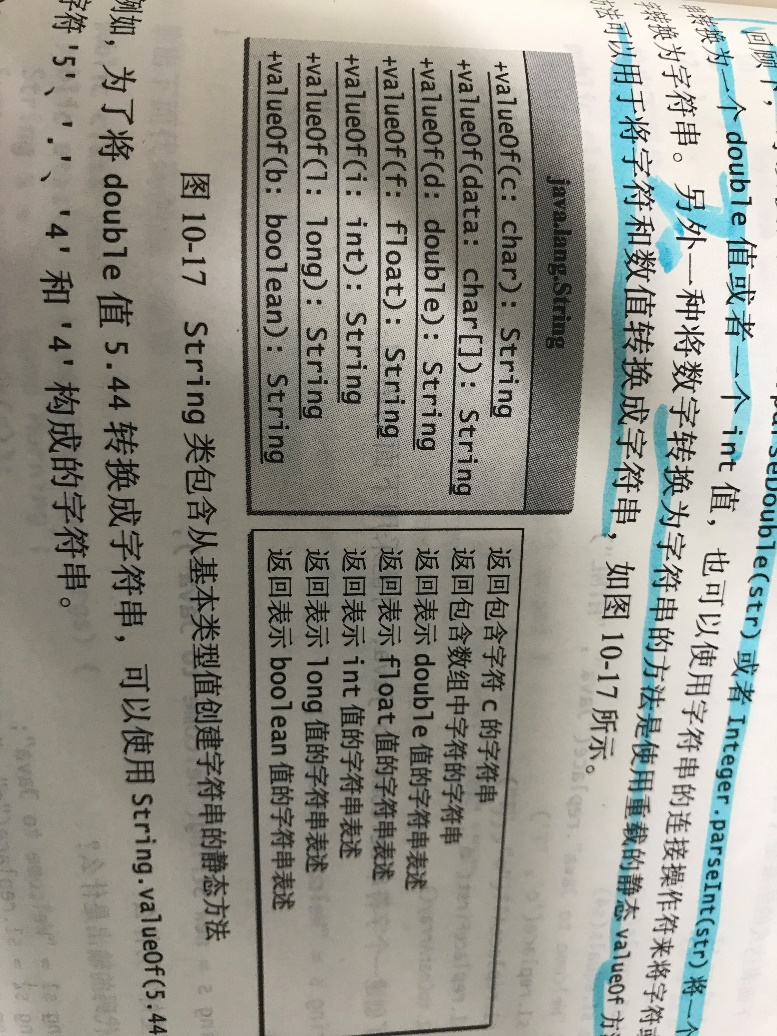
例如，下面的语句使用valueOf方法由一个数组构造一个字符串：

String str=String.valueOf(new char[ ]{‘J’,’a’,’v’,’a’});

### 10.10.6将字符和数值转换成字符串

1.可以使用Double。ParseDouble(str)或者Integer.parseInt(str)将一个字符串转换成一个double值或者一个int值，也可以使用字符串的连接操作符来将字符或者数字转换成字符串。

2.另一种方法是使用重载的静态valueOf方法。该方法可以用于将字符和数值转换成字符串。如图所示：



### 10.10.7格式化字符串

String类包含静态format方法，它可以创建一个格式化的字符串。调用该方法的语法是：

String.format(format,item1,item2,…,itemK)

## 10.11StringBuilder 和 StringBuffer类

1.要点提示：StringBuilder和StringBuffer类似于String类，区别在于String类是不可改变的。一般来说，只要使用字符串的地方，都可以使用StringBuilder/StringBuffer类，它们比String类更加灵活。可以给一个StringBuilder或StringBuffer中添加、插入或追加新的内容，但是String对象一旦创建，它的值就确定了。（对String进行“改变”的两种机制：一是copy一份改；二是这两种新类）。如果是多任务并发访问的，就是用StringBuffer；如果是单任务访问，使用StringBuilder会更有效。

### 10.11.1修改StringBuilder中的字符串

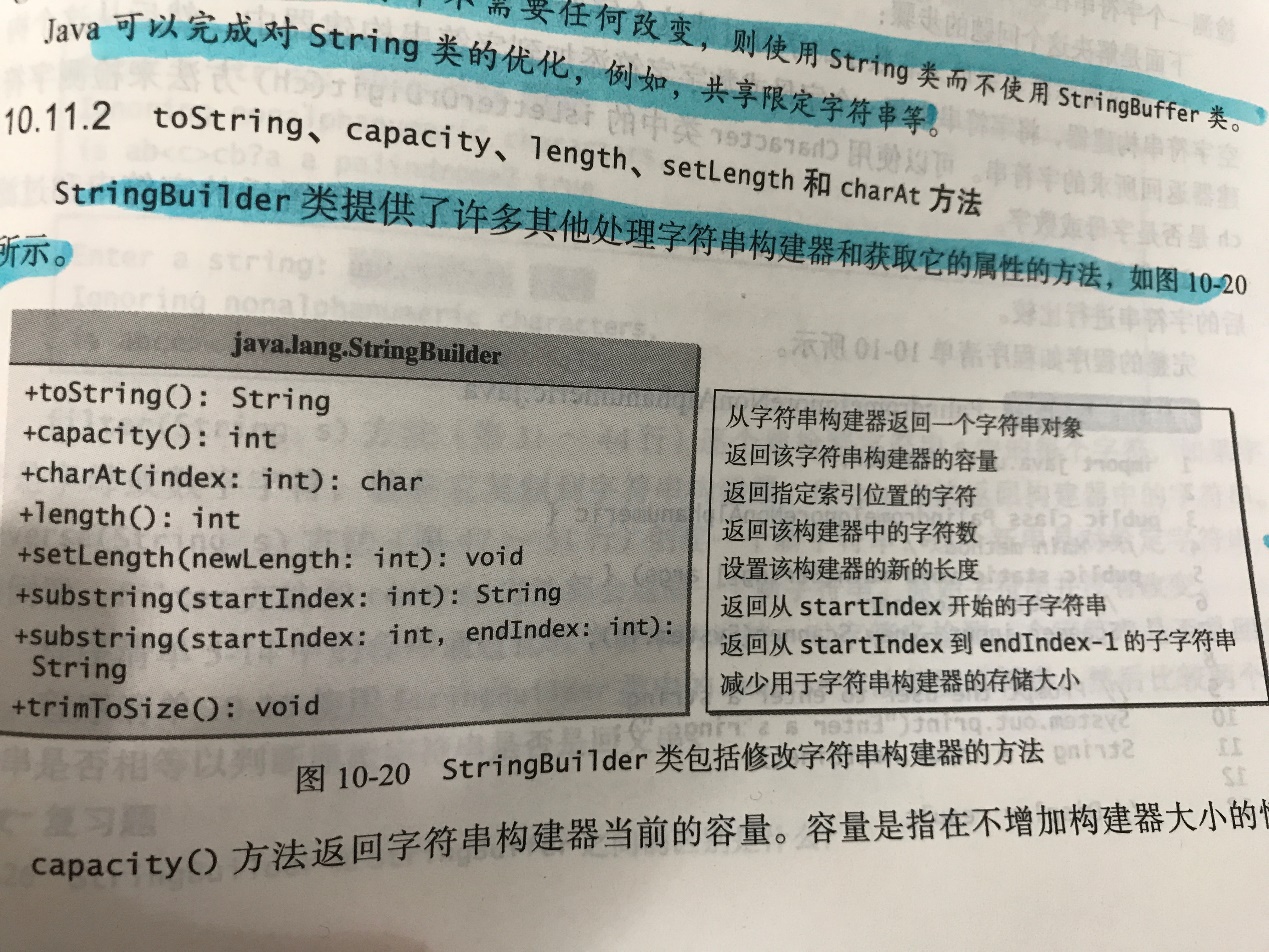
1.可以使用下表中的方法，在字符串构建器的末尾追加新的内容，在字符串构建器的特定位置添加新的内容，还可以删除或替换字符串构建器中的字符。



2.如果一个字符串不需要任何改变，则使用String类而不使用StringBuffer类。Java可以完成对String类的优化，例如，共享限定字符串等。

### 10.11.2 toString、capacity、length、setLength和charAt方法

1.StringBuilder类提供了许多其他处理字符串构建器和获取它的属性的方法，如图所示：



2.注意：字符串的长度总是小于或等于构建器的容量。长度是存储在构建器中字符串的实际大小，而容量是构建器的当前大小。如果有更多的字符添加到字符串构建器，超出它的容量，则构建器的容量就自动增加。在计算机内部，字符串构建器是一个字符数组，因此构建器的容量就是数组的大小。如果超出构建器的容量，就是用新的数组替换现有的数组。新数组的大小为2X（之前数组的长度+1）。

3.提示：可以使用new StringBuilder（initialCapacity）创建指定初始容量的StringBuilder。通过仔细选择初始容量能够使程序更有效。如果容量总是超过构建器的实际使用长度，JVM将永远不需要为构建器重新分配内存。另一方面，如果容量过大将会浪费内存空间。可以使用trimToSize（）方法将容量降到实际大小。

4. 编写一个程序，检测到一个字符串在忽略既非字母又非数字的字符串时是否是一个回文串。过程是“先剔除非法字符，再倒置，最后判断。”

