

- 10.27 如何为一个字符串创建字符串构建器？如何从一个字符串构建器获取字符串？
- 10.28 使用 `StringBuilder` 类中的 `reverse` 方法编写三条语句，倒置字符串 `s`。
- 10.29 编写三条语句，从包含 20 个字符的字符串 `s` 中删除下标从 4 到 10 的子串。使用 `StringBuilder` 类中的 `delete` 方法。
- 10.30 字符串和字符串构建器内部用什么存储字符？
- 10.31 假设给出如下所示的 `s1` 和 `s2`：

```
StringBuilder s1 = new StringBuilder("Java");
StringBuilder s2 = new StringBuilder("HTML");
```

显示执行下列每条语句之后 `s1` 的结果。假定这些表达式都是相互独立的。

- a. `s1.append(" is fun");`
- b. `s1.append(s2);`
- c. `s1.insert(2, "is fun");`
- d. `s1.insert(1, s2);`
- e. `s1.charAt(2);`
- f. `s1.length();`
- g. `s1.deleteCharAt(3);`
- h. `s1.delete(1, 3);`
- i. `s1.reverse();`
- j. `s1.replace(1, 3, "Computer");`
- k. `s1.substring(1, 3);`
- l. `s1.substring(2);`

- 10.32 给出下面程序的输出结果：

```
public class Test {
    public static void main(String[] args) {
        String s = "Java";
        StringBuilder builder = new StringBuilder(s);
        change(s, builder);

        System.out.println(s);
        System.out.println(builder);
    }

    private static void change(String s, StringBuilder builder) {
        s = s + " and HTML";
        builder.append(" and HTML");
    }
}
```

## 关键技术语

Abstract data type (ADT) 抽象数据类型 (ADT)

Aggregation (聚集)

Boxing (装箱)

class abstraction (类抽象)

class encapsulation (类封装)

class contract (类的合约)

composition (组合)

has-a relationship (拥有关系)

multiplicity (多重性)

stack (栈)

unboxing (开箱)

## 本章小结

1. 面向过程范式重在设计方法。面向对象范式将数据和方法耦合在对象中。使用面向对象范式的软件设计重在对象和对象上的操作。面向对象方法结合了面向过程范式的功能以及将数据和操作集成在对象中的特点。
2. 许多 Java 方法要求使用对象作为参数。Java 提供了一个便捷的办法，将基本数据类型合并或包装到一个对象中（例如，包装 `int` 值到 `Integer` 类中，包装 `double` 值到 `Double` 类中）。
3. Java 可以根据上下文自动地将基本类型值转换为对应的包装对象，反之亦然。
4. `BigInteger` 类在计算和处理任意大小的整数方面是很有用的。`BigDecimal` 类可以用作计算和处理带任意精度的浮点数。
5. `String` 对象是不可变的，它的内容不能改变。为了提高效率和节省内存，如果两个直接量字符串有相同的字符序列，Java 虚拟机就将它们存储在一个对象中。这个独特的对象称为限定字符串对象。
6. 正则表达式（缩写 `regex`）是一个描述模板的字符串，用于匹配一系列字符串。可以通过指定一个模板来匹配、替代或者分隔字符串。
7. `StringBuilder/StringBuffer` 类可以用来替代 `String` 类。`String` 对象是不可变的，但是可以向 `StringBuilder/StringBuffer` 对象中添加、插入或追加新的内容。如果字符串的内容不需要任何改变，就使用 `String` 类；如果可能改变的话，则使用 `StringBuilder/StringBuffer` 类。

## 测试题

在线回答本章节的测试题，位于 [www.cs.armstrong.edu/liang/intro10e/quiz.html](http://www.cs.armstrong.edu/liang/intro10e/quiz.html)。

## 编程练习题

### 10.2 ~ 10.3 节

\*10.1（时间类 `Time`）设计一个名为 `Time` 的类。这个类包含：

- 表示时间的数据域 `hour`、`minute` 和 `second`。
- 一个以当前时间创建 `Time` 对象的无参构造方法（数据域的值表示当前时间）。
- 一个构造 `Time` 对象的构造方法，这个对象有一个特定的时间值，这个值是以毫秒表示的、从 1970 年 1 月 1 日午夜开始到现在流逝的时间段（数据域的值表示这个时间）。
- 一个构造带特定的小时、分钟和秒的 `Time` 对象的构造方法。
- 三个数据域 `hour`、`minute` 和 `second` 各自的 `get` 方法。
- 一个名为 `setTime(long elapsedTime)` 的方法使用流逝的时间给对象设置一个新时间。例如，如果流逝的时间为 555550000 毫秒，则转换为 10 小时、10 分钟、10 秒。

画出该类的 UML 图并实现这个类。编写一个测试程序，创建两个 `Time` 对象（使用 `new Time()` 和 `new Time(555550000)`），然后显示它们的小时、分钟和秒。

✎ 提示：前两个构造方法可以从流逝的时间中提取出小时、分钟和秒。对于无参构造方法，当前时间可以使用 `System.currentTimeMillis()` 获取当前时间，如程序清单 2-7 所示。

10.2（`BMI` 类）将下面的新构造方法加入到 `BMI` 类中：

```
/** Construct a BMI with the specified name, age, weight,
 * feet, and inches
 */
public BMI(String name, int age, double weight, double feet,
            double inches)
```

10.3（`MyInteger` 类）设计一个名为 `MyInteger` 的类。这个类包括：

- 一个名为 `value` 的 `int` 型数据域，存储这个对象表示的 `int` 值。

- 一个为指定的 `int` 值创建 `MyInteger` 对象的构造方法。
- 一个返回 `int` 值的 `get` 方法。
- 如果值分别为偶数、奇数或素数，那么 `isEven()`、`isOdd()` 和 `isPrime()` 方法都会返回 `true`。
- 如果指定值分别为偶数、奇数或素数，那么相应的静态方法 `isEven(int)`、`isOdd(int)` 和 `isPrime(int)` 会返回 `true`。
- 如果指定值分别为偶数、奇数或素数，那么相应的静态方法 `isEven(MyInteger)`、`isOdd(MyInteger)` 和 `isPrime(MyInteger)` 会返回 `true`。
- 如果该对象的值与指定的值相等，那么 `equals(int)` 和 `equals(MyInteger)` 方法返回 `true`。
- 静态方法 `parseInt(char[])` 将数字字符构成的数组转换为一个 `int` 值。
- 静态方法 `parseInt(String)` 将一个字符串转换为一个 `int` 值。

画出该类的 UML 图并实现这个类。编写客户程序测试这个类中的所有方法。

10.4 (`MyPoint` 类) 设计一个名为 `MyPoint` 的类，表示一个带 `x` 坐标和 `y` 坐标的点。该类包括：

- 两个带 `get` 方法的数据域 `x` 和 `y` 分别表示它们的坐标。
- 一个创建点 (0,0) 的无参构造方法。
- 一个创建特定坐标点的构造方法。
- 一个名为 `distance` 的方法，返回从该点到 `MyPoint` 类型的指定点之间的距离。
- 一个名为 `distance` 的方法，返回从该点到指定 `x` 和 `y` 坐标的指定点之间的距离。

画出该类的 UML 图并实现这个类。编写一个测试程序，创建两个点 (0,0) 和 (10,30.5)，并显示它们之间的距离。

#### 10.4 ~ 10.8 节

- \*10.5 (显示素数因子) 编写一个程序，提示用户输入一个正整数，然后以降序显示它的所有最小因子。例如：如果整数为 120，那么显示的最小因子为 5、3、2、2、2。使用 `StackOfIntegers` 类存储因子（例如：2、2、2、3、5），获取之后按倒序显示这些因子。
- \*10.6 (显示素数) 编写一个程序，然后以降序显示小于 120 的所有素数。使用 `StackOfIntegers` 类存储这些素数（例如：2、3、5、...），获取之后按倒序显示它们。
- \*\*10.7 (游戏：ATM 机) 使用编程练习题 9.7 中创建的 `Account` 类来模拟一台 ATM 机。创建一个有 10 个账户的数组，其 `id` 为 0, 1, ..., 9，并初始化收支为 100 美元。系统提示用户输入一个 `id`。如果输入的 `id` 不正确，就要求用户输入正确的 `id`。一旦接受一个 `id`，就显示如运行示例所示的主菜单。可以选择 1 来查看当前的收支，选择 2 表示取钱，选择 3 表示存钱，选择 4 表示退出主菜单。一旦退出，系统就会提示再次输入 `id`。所以，系统一旦启动就不会停止。

```
Enter an id: 4 
Main menu
1: check balance
2: withdraw
3: deposit
4: exit
Enter a choice: 1 
The balance is 100.0
```

```
Main menu
1: check balance
2: withdraw
3: deposit
4: exit
Enter a choice: 2 ↵
Enter an amount to withdraw: 3 ↵

Main menu
1: check balance
2: withdraw
3: deposit
4: exit
Enter a choice: 1 ↵
The balance is 97.0

Main menu
1: check balance
2: withdraw
3: deposit
4: exit
Enter a choice: 3 ↵
Enter an amount to deposit: 10 ↵

Main menu
1: check balance
2: withdraw
3: deposit
4: exit
Enter a choice: 1 ↵
The balance is 107.0

Main menu
1: check balance
2: withdraw
3: deposit
4: exit
Enter a choice: 4 ↵

Enter an id:
```

\*\*\*10.8 (财务：税款类 Tax) 编程练习题 8.12 使用数组编写一个计算税款的程序。设计一个名为 Tax 的类，该类包含下面的实例数据域。

- `int filingStatus` (四种纳税人状态之一)：0——单身纳税人、1——已婚共缴纳税人或合法寡妇、2——已婚单缴纳税人、3——家庭纳税人。使用公共静态常量 `SINGLE_FILER(0)`、`MARRIED_JOINTLY_OR_QUALIFYING_WIDOW(ER)(1)`、`MARRIED_SEPARATELY(2)` 和 `HEAD_OF_HOUSEHOLD(3)` 表示这些状态。
- `int[][] brackets`：存储每种纳税人的纳税等级。
- `double[] rates`：存储每种纳税等级的税率。
- `double taxableIncome`：存储可征税收入。

给每个数据域提供 `get` 和 `set` 方法，并提供返回税款的 `getTax()` 方法。该类还提供一个无参构造方法和构造方法 `Tax(filingStatus,brackets,rates,taxableIncome)`。

画出该类的 UML 图并实现这个类。编写一个测试程序，使用 Tax 类对所给四种纳税人打印 2001 年和 2009 年的税款表，可征税收入范围在 50 000 美元和 60 000 美元之间，间隔区间为 1000 美元。2009 年的税率参见表 3-2，2001 年的税率参见表 10-1。

表 10-1 2001 年美国联邦个人所得税税率表

税率	单身纳税人	已婚共缴纳税人或符合条件的丧偶人士	已婚单缴纳税人	家庭纳税人
15%	\$27 050 以下	\$45 200 以下	\$22 600 以下	\$36 250 以下
27.5%	\$27 051 ~ \$65 550	\$45 201 ~ \$109 250	\$22 601 ~ \$54 625	\$36 251-\$93 650
30.5%	\$65 551 ~ \$136 750	\$109 251 ~ 166 500	\$54 626 ~ \$83 250	\$93 651-\$151 650
35.5%	\$136 751 ~ \$29 7350	\$166 501 ~ \$297 350	\$83 251 ~ \$148 675	\$151 651-\$297 350
39.1%	\$297 351 及以上	\$297 351 及以上	\$148 676 及以上	\$297 351 及以上

\*\*10.9 (课程类 Course) 如下改写 Course 类:

- 程序清单 10-6 中数组的大小是固定的。对它进行改进, 通过创建一个新的更大的数组并复制当前数组的内容来实现数组大小的自动增长。
- 实现 dropStudent 方法。
- 添加一个名为 clear() 的新方法, 然后删掉选某门课程的所有学生。

编写一个测试程序, 创建一门课程, 添加三个学生, 删除一个学生, 然后显示这门课程的学生。

\*10.10 (Queue 类) 10.6 节给出了一个用于 Stack 的类。设计一个名为 Queue 的类用于存储整数。像栈一样, 队列具有元素。在栈中, 元素以“后进先出”的方式获得。在队列中, 元素以“先进先出”的方式获取。该类包含:

- 一个名为 element 的 int[] 类型的数据域, 保存队列中的 int 值。
- 一个名为 size 的数据域, 保存队列中的元素个数。
- 一个构造方法, 使用默认的容量 8 来创建一个 Queue 对象。
- 方法 enqueue(int v), 用于将 v 加入到队列中。
- 方法 dequeue(), 用于从队列中移除元素并返回该元素。
- 方法 empty(), 如果队列是空的话, 该方法返回 true。
- 方法 getSize(), 返回队列的大小。

画出该类的 UML 图并实现这个类, 使之初始数组的大小为 8。一旦元素个数超过了大小, 数组大小将会翻倍。如果一个元素从数组的开始部分移除, 你需要将数组中的所有元素往左边改变一个位置。编写一个测试程序, 增加从 1 到 20 的 21 个成员, 然后将这些数字移除并显示它们。

\*10.11 (几何: Circle2D 类) 定义 Circle2D 类, 包括:

- 两个带有 get 方法的名为 x 和 y 的 double 型数据域, 表明圆的中心点。
- 一个带 get 方法的数据域 radius。
- 一个无参构造方法, 该方法创建一个 (x,y) 值为 (0,0) 且 radius 为 1 的默认圆。
- 一个构造方法, 创建带指定的 x、y 和 radius 的圆。
- 一个返回圆面积的方法 getArea()。
- 一个返回圆周长的方法 getPerimeter()。
- 如果给定的点 (x,y) 在圆内, 那么方法 contains(double x, double y) 返回 true, 如图 10-21a 所示。
- 如果给定的圆在这个圆内, 那么方法 contains(Circle2D circle) 返回 true, 如图 10-21b 所示。
- 如果给定的圆和这个圆重叠, 那么方法 overlaps(Circle2D circle) 返回 true, 如图 10-21c 所示。

画出该类的 UML 图并实现这个类。编写测试程序, 创建一个 Circle2D 对象 c1(new Circle2D(2,2,5.5)), 显示它的面积和周长, 还要显示 c1.contains(3,3)、c1.contains(new Circle2D (4,5,10.5)) 和 c1.overlaps(new Circle2D(3,5,2.3))。



a) 点在圆内

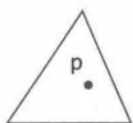
b) 一个圆在另一个圆内

c) 一个圆和另一个圆重叠

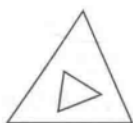
图 10-21

\*\*\*10.12 (几何: Triangle2D 类) 定义 Triangle2D 类, 包含:

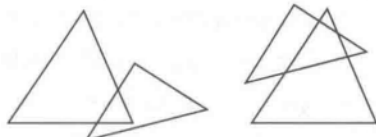
- 三个名为 p1、p2 和 p3 的 MyPoint 类型数据域, 这三个数据域都带有 get 和 set 方法。MyPoint 在编程练习题 10.4 中定义。
- 一个无参构造方法, 该方法创建三个坐标为 (0,0)、(1,1) 和 (2,5) 的点组成的默认三角形。
- 一个创建带指定点的三角形的构造方法。
- 一个返回三角形面积的方法 getArea()。
- 一个返回三角形周长的方法 getPerimeter()。
- 如果给定的点 p 在这个三角形内, 那么方法 contains(MyPoint p) 返回 true, 如图 10-22a 所示。
- 如果给定的三角形在这个三角形内, 那么方法 contains(Triangle2D t) 返回 true, 如图 10-22b 所示。
- 如果给定的三角形和这个三角形重叠, 那么方法 overlaps(Triangle2D t) 返回 true, 如图 10-22c 所示。



a) 点在三角形内



b) 一个三角形在另一个三角形内

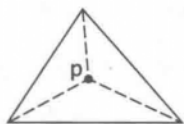


c) 一个三角形和另一个三角形重叠

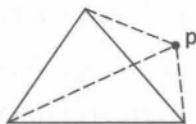
图 10-22

画出该类的 UML 图并实现这个类。编写测试程序, 使用构造方法 `new Triangle2D(new MyPoint(2.5,2), new MyPoint(4.2,3), new MyPoint(5,3.5))` 创建一个 Triangle2D 对象 t1, 显示它的面积和周长, 并显示 `t1.contains(3,3)`、`t1.contains(new Triangle2D(new MyPoint(2.9,2), new MyPoint(4,1), MyPoint(1,3.4)))` 和 `t1.overlaps(new Triangle2D(new MyPoint(2,5.5), new MyPoint(4,-3), MyPoint(2,6.5)))` 的结果。

提示: 关于计算三角形面积的公式请参见编程练习题 2.19。为了检测一个点是否在三角形中, 画三条虚线, 如图 10-23 所示。如果点在三角形中, 每条虚线应该和边相交一次。如果虚线和边相交两次, 那么这个点肯定在这个三角形外。找到两条线交点的算法, 参加编程练习题 3.25。



a) 点在三角形内



b) 一个点在三角形外

图 10-23

\*10.13 (几何: MyRectangle2D 类) 定义 MyRectangle2D 类, 包含:

- 两个名为 x 和 y 的 double 型数据域表明矩形的中心点, 这两个数据域都带有 get 和 set 方法 (假设这个矩形的边与 x 轴和 y 轴平行)。
- 带 get 和 set 方法的数据域 width 和 height。

- 一个无参构造方法，该方法创建一个 (x,y) 值为 (0,0) 且 width 和 height 为 1 的默认矩形。
- 一个构造方法，创建带指定的 x、y、width 和 height 的矩形。
- 方法 `getArea()` 返回矩形的面积。
- 方法 `getPerimeter()` 返回矩形的周长。
- 如果给定的点 (x,y) 在矩形内，那么方法 `contains(double x, double y)` 返回 true，如图 10-24a 所示。
- 如果给定的矩形在这个矩形内，那么方法 `contains(MyRectangle2D r)` 返回 true，如图 10-24b 所示。
- 如果给定的矩形和这个矩形重叠，那么方法 `overlaps(MyRectangle2D r)` 返回 true，如图 10-24c 所示。



a) 点在矩形内      b) 一个矩形在另一个矩形内      c) 一个矩形和另一个矩形重叠      d) 点被包围在矩形中

图 10-24

画出该类的 UML 图并实现这个类。编写测试程序，创建一个 `MyRectangle2D` 对象 `r1(new MyRectangle2D(2,2,5.5,4.9))`，显示它的面积和周长，然后显示 `r1.contains(3,3)`、`r1.contains(new MyRectangle2D(4,5,10.5,3.2))` 和 `r1.overlaps(new MyRectangle2D(3,5, 2.3,5.4))` 的结果。

\*10.14 (MyDate 类) 设计一个名为 `MyDate` 的类。该类包含：

- 表示日期的数据域 `year`、`month` 和 `day`。月份是从 0 开始的，即 0 表示一月份。
- 一个无参构造方法，该方法创建当前日期的 `MyDate` 对象。
- 一个构造方法，创建以从 1970 年 1 月 1 日午夜开始流逝的毫秒数为时间的 `MyDate` 对象。
- 一个构造方法，创建一个带指定年、月、日的 `MyDate` 对象。
- 三个数据域 `year`、`month` 和 `day` 的 `get` 方法。
- 一个名为 `setDate(long elapsedTime)` 使用流逝的时间为对象设置新数据的方法。

画出该类的 UML 图并实现这个类。编写测试程序，创建一个测试程序，创建两个 `Date` 对象（使用 `new Date()` 和 `new Date(34355555133101L)`），然后显示它们的小时、分钟和秒。

✎ 提示：前两个构造方法将从逝去的时间中提取出年、月、日。例如：如果逝去的时间是 56155550000 毫秒，那么年就是 1987，月就是 9，而天是 18。可以使用编程练习题 9.5 中讨论的 `GregorianCalendar` 类来简化编程。

\*10.15 (几何：边界矩形) 边界矩形是指包围一个二维平面上一系列点的矩形，如图 10-24d 所示。编写一个方法，为二维平面上一系列点返回一个边界矩形，如下所示：

```
public static MyRectangle2D getRectangle(double[][] points)
```

`Rectangle2D` 类在编程练习题 10.13 中定义。编写一个测试程序，提示用户输入 5 个点，然后显示边界矩形的中心、宽度以及高度。下面是一个运行示例：

```
Enter five points: 1.0 2.5 3.4 5.6 7.8 9.10
The bounding rectangle's center (5.0, 6.25), width 8.0, height 7.5
```

## 10.9 节

\*10.16 (被 2 或 3 整除) 找出能被 2 或 3 整除的前 10 个数字，这些数字有 50 个十进制位数。

- \*10.17 (平方数) 找出大于 `Long.MAX_VALUE` 的前 10 个平方数。平方数是指形式为  $n^2$  的数。例如, 4、9 以及 16 都是平方数。找到一种方法, 使你的程序能快速运行。
- \*10.18 (大素数) 编写程序找出五个大于 `Long.MAX_VALUE` 的素数。
- \*10.19 (Mersenne 素数) 如果一个素数可以写成  $2^p - 1$  的形式, 那么该素数就称为 Mersenne 素数, 其中的  $p$  是一个正整数。编写程序找出  $p \leq 100$  的所有 Mersenne 素数, 然后显示如下所示的输出。(必须使用 `BigInteger` 来存储数字, 因为它太大了, 不能用 `long` 来存储。程序可能需要运行几个小时。)

$p$	$2^p - 1$
2	3
3	7
5	31
...	

- \*10.20 (近似  $e$ ) 编程练习题 5.26 使用下面数列近似计算  $e$ :

$$e = 1 + \frac{1}{1!} + \frac{1}{2!} + \frac{1}{3!} + \frac{1}{4!} + \cdots + \frac{1}{i!}$$

为了得到更好的精确度, 在计算中使用 25 位精度的 `BigDecimal`。编写程序显示当  $i=100, 200, \dots, 1000$  时  $e$  的值。

- 10.21 (被 5 或 6 整除) 找出能被 5 或 6 整除的大于 `Long.MAX_VALUE` 的前 10 个数字。

#### 第 10.10 ~ 10.11 节

- \*\*10.22 (实现 `String` 类) Java 库中提供了 `String` 类, 给出你自己对下面方法的实现 (将新类命名为 `MyString1`):

```
public MyString1(char[] chars);
public char charAt(int index);
public int length();
public MyString1 substring(int begin, int end);
public MyString1 toLowerCase();
public boolean equals(MyString1 s);
public static MyString1 valueOf(int i);
```

- \*\*10.23 (实现 `String` 类) 在 Java 库中提供了 `String` 类, 给出你自己对下面方法的实现 (将新类命名为 `MyString2`):

```
public MyString2(String s);
public int compare(String s);
public MyString2 substring(int begin);
public MyString2 toUpperCase();
public char[] toChars();
public static MyString2 valueOf(boolean b);
```

- 10.24 (实现 `Character` 类) 在 Java 库中提供了 `Character` 类, 给出你自己对这个类的实现 (将新类命名为 `MyCharacter`)。

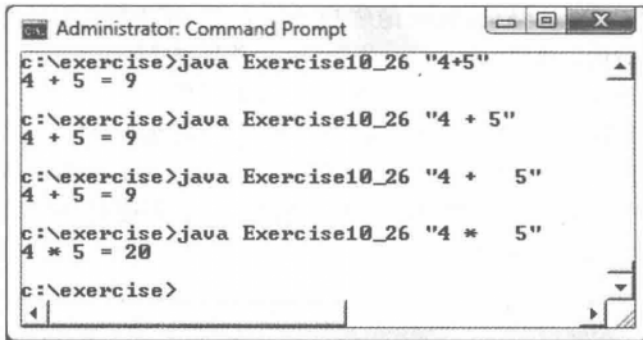
- \*\*10.25 (新的字符串 `split` 方法) `String` 类中的 `split` 方法会返回一个字符串数组, 该数组是由分隔符分隔开的子串构成的。但是, 这个分隔符是不返回的。实现下面的新方法, 方法返回字符串数组, 这个数组由用匹配字符分隔开的子串构成, 子串也包括匹配字符。

```
public static String[] split(String s, String regex)
```

例如, `split("ab#12#453", "#")` 会返回 `ab`、`#`、`12`、`#` 和 `453` 构成的 `String` 数组, 而 `split("a?b?gf#e", "[?#]")` 会返回 `a`、`?`、`b`、`?`、`gf`、`#` 和 `e` 构成的字符串数组。

- \*10.26 (计算器) 修改程序清单 7-9, 接收一个字符串的表达式, 其中操作符和操作数由 0 到多个空格隔开。例如, `3+4` 和 `3 + 4` 都是可以接受的表达式。下面是一个运行示例:





```
Administrator: Command Prompt
c:\exercise>java Exercise10_26 "4+5"
4 + 5 = 9
c:\exercise>java Exercise10_26 "4 + 5"
4 + 5 = 9
c:\exercise>java Exercise10_26 "4 + 5"
4 + 5 = 9
c:\exercise>java Exercise10_26 "4 * 5"
4 * 5 = 20
c:\exercise>
```

**\*\*10.27** (实现 `StringBuilder` 类) 在 Java 库中提供了 `StringBuilder` 类。给出你自己对下面方法的实现 (将新类命名为 `MyStringBuilder1`):

```
public MyStringBuilder1(String s);
public MyStringBuilder1 append(MyStringBuilder1 s);
public MyStringBuilder1 append(int i);
public int length();
public char charAt(int index);
public MyStringBuilder1 toLowerCase();
public MyStringBuilder1 substring(int begin, int end);
public String toString();
```

**\*\*10.28** (实现 `StringBuilder` 类) 在 Java 库中提供了 `StringBuilder` 类。给出你自己对下面方法的实现 (将新类命名为 `MyStringBuilder2`):

```
public MyStringBuilder2();
public MyStringBuilder2(char[] chars);
public MyStringBuilder2(String s);
public MyStringBuilder2 insert(int offset, MyStringBuilder2 s);
public MyStringBuilder2 reverse();
public MyStringBuilder2 substring(int begin);
public MyStringBuilder2 toUpperCase();
```