

```
109  /** Get the number of days in a month */
110  public static int getNumberOfDaysInMonth(int year, int month) {
111      if (month == 1 || month == 3 || month == 5 || month == 7 ||
112          month == 8 || month == 10 || month == 12)
113          return 31;
114
115      if (month == 4 || month == 6 || month == 9 || month == 11)
116          return 30;
117
118      if (month == 2) return isLeapYear(year) ? 29 : 28;
119
120      return 0; // If month is incorrect
121  }
122
123  /** Determine if it is a leap year */
124  public static boolean isLeapYear(int year) {
125      return year % 400 == 0 || (year % 4 == 0 && year % 100 != 0);
126  }
127 }
```

该程序没有检测用户输入的有效性。例如：如果用户输入的月份不在 1 到 12 之间，或者年份在 1800 年之前，那么程序就会显示出错误的日历。为避免出现这样的错误，可以添加一个 if 语句在打印日历前检查输入。

该程序可以打印一个月的日历，还可以很容易地修改为打印整年的日历。尽管它现在只能处理 1800 年 1 月以后的月份，但是可以稍作修改，便能够打印 1800 年之前的月份。

6.11.4 逐步求精的优势

逐步求精将一个大问题分解为小的易于处理的子问题。每个子问题可以使用一个方法来实现。这种方法使得问题更加易于编写、重用、调试、修改和维护。

1. 更简单的程序

打印日历的程序比较长。逐步求精方法将其分解为较小的方法，而不是在一个方法中写很长的语句序列。这样简化了程序，使得整个程序易于阅读和理解。

2. 重用方法

逐步求精提高了一个程序中的方法重用。isLeapYear 方法只定义了一次，从 getTotalNumberOfDays 和 getNumberOfDaysInMonth 方法中都进行了调用。这减少了冗余的代码。

3. 易于开发、调试和测试

因为每个子问题在一个方法中解决，而一个方法可以分别的开发、调试以及测试。这隔离了错误，使得开发、调试和测试更加容易。

编写大型程序时，可以使用自顶向下或自底向上的方法。不要一次性地编写整个程序。使用这些方法似乎浪费了更多的开发时间（因为要反复编译和运行程序），但实际上，它会更节省时间并使调试更容易。

4. 更方便团队合作

当一个大问题分解为许多子问题，各个子问题可以分配给不同的编程人员。这更加易于编程人员进行团队工作。

关键术语

actual parameter (实际参数)
ambiguous invocation (歧义调用)

argument (实参)
divide and conquer (分治)

formal parameter (ie. parameter) (形式参数即形参)	modifier (修饰符)
information hiding (信息隐藏)	parameter (参数)
method (方法)	pass-by-value (按值传递)
method abstraction (方法抽象)	scope of variable (变量的作用域)
method overloading (方法重载)	stepwise refinement (逐步求精)
method signature (方法签名)	stub (待完善方法)

本章小结

1. 程序模块化和可重用性是软件工程的中心目标之一。Java 提供了很多有助于完成这一目标的有效结构。方法就是一个这样的结构。
2. 方法头指定方法的修饰符、返回值类型、方法名和参数。本章所有的方法都使用静态修饰符 `static`。
3. 方法可以返回一个值。返回值类型 `returnValueType` 是方法要返回的值的的数据类型。如果方法不返回值, 则返回值类型就是关键字 `void`。
4. 参数列表是指方法中参数的类型、次序和数量。方法名和参数列表一起构成方法签名 (method signature)。参数是可选的, 也就是说, 一个方法可以不包含参数。
5. `return` 语句也可以用在 `void` 方法中, 用来终止方法并返回到方法的调用者。在方法中, 有时用于改变正常流程控制是很有用的。
6. 传递给方法的实际参数应该与方法签名中的形式参数具有相同的数目、类型和顺序。
7. 当程序调用一个方法时, 程序控制就转移到被调用的方法。被调用的方法执行到该方法的 `return` 语句或到达方法结束的右括号时, 将程序控制还给调用者。
8. 在 Java 中, 带返回值的方法也可以当作语句调用。在这种情况下, 调用函数只要忽略返回值即可。
9. 方法可以重载。这就意味着两个方法可以拥有相同的方法名, 只要它们的方法参数列表不同即可。
10. 在方法中声明的变量称作局部变量。局部变量的作用域是从声明它的地方开始, 到包含这个变量的块结束为止。局部变量在使用前必须声明和初始化。
11. 方法抽象是把方法的应用和实现分离。用户可以在不知道方法是如何实现的情况下使用方法。方法的实现细节封装在方法内, 对调用该方法的用户隐藏。这称为信息隐藏或封装。
12. 方法抽象将程序模块化为整齐、层次分明的形式。将程序写成简洁的方法构成的集合会比其他方式更容易编写、调试、维护和修改。这种编写风格也会提高方法的可重用性。
13. 当实现一个大型程序时, 可以使用自顶向下或自底向上的编码方法。不要一次性编写完整个程序。这种方式似乎浪费了更多的编码时间 (因为要反复编译和运行这个程序), 但实际上, 它会更节省时间并使调试更容易。

测试题

本章节的测试题位于 www.cs.armstrong.edu/liang/intro10e/quiz.html。

编程练习题

☞ 注意: 本章练习中学生常犯的错误是, 没有实现符合要求的方法, 尽管主程序的输出是正确的。这类错误的示例参见: www.cs.armstrong.edu/liang/CommonMethodErrorJava.pdf。

6.2 ~ 6.9 节

- 6.1 (数学: 五角数) 一个五角数被定义为 $n(3n-1)/2$, 其中 $n=1, 2, \dots$ 。所以, 开始的几个数字就是 1, 5, 12, 22, ..., 编写下面的方法返回一个五角数:

```
public static int getPentagonalNumber(int n)
```

编写一个测试程序显示前 100 个五角数，每行显示 10 个。

*6.2 (求一个整数各位数字之和) 编写一个方法，计算一个整数各位数字之和。使用下面的方法头：

```
public static int sumDigits(long n)
```

例如：sumDigits(234) 返回 9(2+3+4)。

✎ 提示：使用求余操作符 % 提取数字，用除号 / 去掉提取出来的数字。例如：使用 234%10 (=4) 抽取 4。然后使用 234/10 (=23) 从 234 中去掉 4。使用一个循环来反复提取和去掉每位数字，直到所有的位数都提取完为止。

编写程序提示用户输入一个整数，然后显示这个整数所有数字的和。

**6.3 (回文整数) 使用下面的方法头编写两个方法：

```
// Return the reversal of an integer, i.e., reverse(456) returns 654
public static int reverse(int number)
```

```
// Return true if number is a palindrome
public static boolean isPalindrome(int number)
```

使用 reverse 方法实现 isPalindrome。如果一个数字的反向倒置数和它的顺向数一样，这个数就称作回文数。编写一个测试程序，提示用户输入一个整数值，然后报告这个整数是否是回文数。

*6.4 (反向显示一个整数) 使用下面的方法头编写方法，反向显示一个整数：

```
public static void reverse(int number)
```

例如：reverse(3456) 返回 6543。编写一个测试程序，提示用户输入一个整数，然后显示它的反向数。

*6.5 (对三个数排序) 使用下面的方法头编写方法，按升序显示三个数：

```
public static void displaySortedNumbers(
    double num1, double num2, double num3)
```

编写测试程序，提示用户输入三个数字，调用方法以升序显示他们。

*6.6 (显示图案) 编写方法显示如下图案：

```

      1
     2 1
    3 2 1
   ...
  n n-1 ... 3 2 1
```

该方法头为：

```
public static void displayPattern(int n)
```

*6.7 (财务应用程序：计算未来投资价值) 编写一个方法，计算按照给定的年数和利率计算未来投资价值，未来投资是用编程练习题 2.21 中的公式计算得到的。

使用下面的方法头：

```
public static double futureInvestmentValue(
    double investmentAmount, double monthlyInterestRate, int years)
```

例如：futureInvestmentValue(10000, 0.05/12, 5) 返回 12833.59。

编写一个测试程序，提示用户输入投资额 (例如 1000)、利率 (例如 9%)，然后打印年份从 1 到 30 年的未来投资价值，如下所示：

The amount invested:	1000	Enter
Annual interest rate:	9	Enter
Years	Future Value	
1	1093.80	
2	1196.41	
...		
29	13467.25	
30	14730.57	

6.8 (摄氏度和华氏度之间的转换) 编写一个类，包含下面两个方法：

```
/** Convert from Celsius to Fahrenheit */
public static double celsiusToFahrenheit(double celsius)

/** Convert from Fahrenheit to Celsius */
public static double fahrenheitToCelsius(double fahrenheit)
```

转换公式如下：

华氏度 = (9.0 / 5) * 摄氏度 + 32
摄氏度 = (5.0 / 9) * (华氏度 - 32)

编写一个测试程序，调用这两个方法来显示如下表格：

摄氏度	华氏度	华氏度	摄氏度
40.0	104.0	120.0	48.89
39.0	102.2	110.0	43.33
...			
32.0	89.6	40.0	4.44
31.0	87.8	30.0	-1.11

6.9 (英尺和米之间的转换) 编写一个类，包含如下两个方法：

```
/** Convert from feet to meters */
public static double footToMeter(double foot)

/** Convert from meters to feet */
public static double meterToFoot(double meter)
```

转换公式如下：

米 = 0.305 * 英尺
英尺 = 3.279 * 米

编写一个测试程序，调用这两个方法以显示下面的表格：

英尺	米	米	英尺
1.0	0.305	20.0	65.574
2.0	0.610	25.0	81.967
...			
9.0	2.745	60.0	196.721
10.0	3.050	65.0	213.115

6.10 (使用 isPrime 方法) 程序清单 6-7 提供了测试某个数字是否是素数的方法 isPrime(int number)。使用这个方法求小于 10000 的素数个数。

6.11 (财务应用程序：计算佣金) 编写一个方法，利用编程练习题 5.39 中的方案计算佣金。方法头如下所示：

```
public static double computeCommission(double salesAmount)
```

编写一个测试程序，显示下面表格：

销售总额	酬金
10000	900.0
15000	1500.0
...	
95000	11100.0
100000	11700.0

6.12（显示字符）使用下面的方法头，编写一个打印字符的方法：

```
public static void printChars(char ch1, char ch2, int
    numberPerLine)
```

该方法打印 ch1 到 ch2 之间的字符，每行按指定个数打印。编写一个测试程序，打印从 '1' 到 'Z' 的字符，每行打印 10 个。字符之间使用一个空格字符隔开。

*6.13（数列求和）编写一个方法计算下列级数：

$$m(i)=\frac{1}{2}+\frac{2}{3}+\cdots+\frac{i}{i+1}$$

编写一个测试程序，显示下面的表格：

i	m(i)
1	0.5000
2	1.1667
...	
19	16.4023
20	17.3546

*6.14（估算 π）π 可以使用下面的数列进行计算：

$$m(i)=4\left(1-\frac{1}{3}+\frac{1}{5}-\frac{1}{7}+\frac{1}{9}-\frac{1}{11}+\cdots+\frac{(-1)^{i+1}}{2i-1}\right)$$

编写一个方法，对于给定的 i 返回 m(i)，并且编写一个测试程序，显示如下表格：


i	m(i)
1	4.0000
101	3.1515
201	3.1466
301	3.1449
401	3.1441
501	3.1436
601	3.1433
701	3.1430
801	3.1428
901	3.1427

*6.15（财务应用程序：打印税表）程序清单 3-5 给出计算税款的程序。使用下面的方法头编写一个计算税款的方法：

```
public static double computeTax(int status, double taxableIncome)
```

使用这个方法编写程序，打印可征税收入从 50 000 美元到 60 000 美元，收入间隔为 50 美元的所有婚姻状态纳税人的纳税表，如下所示：

Taxable Income	Single	Married Joint	Married Separate	Head of a House
50000	8688	6665	8688	7353
50050	8700	6673	8700	7365
...				
59950	11175	8158	11175	9840
60000	11188	8165	11188	9853

 提示：使用 `Math.round`（即 `Math.round(computeTax(status, taxableIncome))`）将税收舍入为整数。

*6.16（一年的天数）使用下面的方法头编写一个方法，返回一年的天数：

```
public static int numberOfDaysInAYear(int year)
```

编写一个测试程序，显示从 2000 年到 2020 年每年的天数。

6.10 ~ 6.11 节

*6.17（显示 0 和 1 构成的矩阵）编写一个方法，使用下面的方法头显示 $n \times n$ 的矩阵：

```
public static void printMatrix(int n)
```

每个元素都是随机产生的 0 或 1。编写一个测试程序，提示用户输入 n ，显示如下所示的 $n \times n$ 矩阵：

```
Enter n: 3
0 1 0
0 0 0
1 1 1
```

**6.18（检测密码）一些网站对于密码具有一些规则。编写一个方法，检测字符串是否是一个有效密码。假定密码规则如下：

- 密码必须至少 8 位字符。
- 密码仅能包含字母和数字。
- 密码必须包含至少两个数字。

编写一个程序，提示用户输入一个密码，如果符合规则，则显示 `Valid Password`，否则显示 `Invalid Password`。

*6.19（`MyTriangle` 类）创建一个名为 `MyTriangle` 的类，它包含如下两个方法：

```
/** Return true if the sum of any two sides is
 * greater than the third side. */
public static boolean isValid(
    double side1, double side2, double side3)
/** Return the area of the triangle. */
public static double area(
    double side1, double side2, double side3)
```

编写一个测试程序，读入三角形三边的值，若输入有效，则计算面积；否则显示输入无效。三角形面积的计算公式在编程练习题 2.19 中给出。

*6.20（计算一个字符串中字母的个数）编写一个方法，使用下面的方法头计算字符串中的字母个数：

```
public static int countLetters(String s)
```

编写一个测试程序，提示用户输入字符串，然后显示字符串中的字母个数。

*6.21（电话按键盘）国际标准的字母 / 数字匹配图如编程练习题 4.15 所示，编写一个方法，返回给定大写字母的数字，如下所示：

```
int getNumber(char uppercaseLetter)
```

编写一个测试程序，提示用户输入字符串形式的电话号码。输入的数字可能会包含字母。程序将字母（大写或者小写）翻译成一个数字，然后保持其他字符不变。下面是该程序的运行示例：

```
Enter a string: 1-800-Flowers
1-800-3569377
```

```
Enter a string: 1800flowers
18003569377
```

- **6.22**（数学：平方根的近似求法）有几种实现 Math 类中 sqrt 方法的技术。其中一个称为巴比伦法。它通过使用下面公式的反复计算近似地得到：

```
nextGuess = (lastGuess + n / lastGuess) / 2
```

当 nextGuess 和 lastGuess 几乎相同时，nextGuess 就是平方根的近似值。最初的猜测值可以是任意一个正值（例如 1）。这个值就是 lastGuess 的初始值。如果 nextGuess 和 lastGuess 的差小于一个很小的数，比如 0.0001，就可以认为 nextGuess 是 n 的平方根的近似值；否则，nextGuess 就成为 lastGuess，近似过程继续执行。实现下面的方法，返回 n 的平方根。

```
public static double sqrt(long n)
```

- *6.23**（指定字符的出现次数）使用下面的方法头编写一个方法，找到一个字符串中指定字符的出现次数。

```
public static int count(String str, char a)
```

例如，count("Welcome", 'e') 返回 2。编写一个测试程序，提示用户输入一个字符串以及一个字符，显示该字符在字符串中出现的次数。

6.10 ~ 6.12 节

- **6.24**（显示当前日期和时间）程序清单 2-7 显示当前时间。改进这个例子，显示当前的日期和时间。在程序清单 6-12 中的日历例子，可以提供一些如何求年、月和日的思路。

- **6.25**（将毫秒数转换成小时数、分钟数和秒数）使用下面的方法头，编写一个将毫秒数转换成小时数、分钟数和秒数的方法。

```
public static String convertMillis(long millis)
```

该方法返回形如“小时：分钟：秒”的字符串。例如：convertMillis(5500) 返回字符串 0:0:5，convertMillis(100000) 返回字符串 0:1:40，convertMillis(555550000) 返回字符串 154:19:10。

综合题

- **6.26**（回文素数）回文素数是指一个数同时为素数和回文数。例如：131 是一个素数，同时也是一个回文素数。数字 313 和 757 也是如此。编写程序，显示前 100 个回文素数。每行显示 10 个数并且准确对齐，数字中间用空格隔开。如下所示：

```
2 3 5 7 11 101 131 151 181 191
313 353 373 383 727 757 787 797 919 929
...
```

- **6.27**（反素数）反素数（反转拼写的素数）是指一个非回文素数，将其反转之后也是一个素数。例如：17 是一个素数，而 71 也是一个素数，所以 17 和 71 是反素数。编写程序，显示前 100 个反素

数。每行显示 10 个，并且数字间用空格隔开，如下所示：

```
13 17 31 37 71 73 79 97 107 113
149 157 167 179 199 311 337 347 359 389
...
```

- **6.28 (梅森素数)** 如果一个素数可以写成 $2^p - 1$ 的形式，其中 p 是某个正整数，那么这个素数就称作梅森素数。编写程序，找出 $p \leq 31$ 的所有梅森素数，然后显示如下的输出结果：

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

- **6.29 (双素数)** 双素数是指一对差值为 2 的素数。例如：3 和 5 就是一对双素数，5 和 7 是一对双素数，而 11 和 13 也是一对双素数。编写程序，找出小于 1000 的所有双素数。显示结果如下所示：

```
(3, 5)
(5, 7)
...
```

- **6.30 (游戏：双骰儿赌博)** 掷双骰子游戏是赌场中非常流行的骰子游戏。编写程序，玩这个游戏的一个变种，如下所示：

掷两个骰子。每个骰子有六个面，分别表示值 1, 2, ..., 6。检查这两个骰子的和。如果和为 2、3 或 12（称为掷骰子 (craps)），你就输了；如果和是 7 或者 11（称作自然 (natural)），你就赢了；但如果和是其他数字（例如：4、5、6、8、9 或者 10），就确定了一个点。继续掷骰子，直到掷出一个 7 或者掷出和刚才相同的点数。如果掷出的是 7，你就输了。如果掷出的点数和你前一次掷出的点数相同，你就赢了。

程序扮演一个独立的玩家。下面是一些运行示例。

```
You rolled 5 + 6 = 11
You win
```

```
You rolled 1 + 2 = 3
You lose
```

```
You rolled 4 + 4 = 8
point is 8
You rolled 6 + 2 = 8
You win
```

```
You rolled 3 + 2 = 5
point is 5
You rolled 2 + 5 = 7
You lose
```

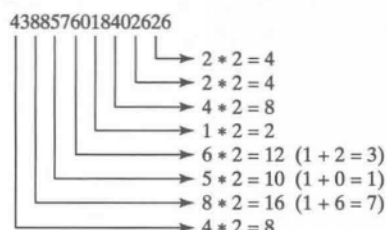
- **6.31 (财务应用程序：信用卡号的合法性)** 信用卡号遵循下面的模式。一个信用卡号必须是 13 到 16 位的整数。它的开头必须是：

- 4，指 Visa 卡
- 5，指 Master 卡
- 37，指 American Express 卡
- 6，指 Discover 卡

在 1954 年，IBM 的 Hans Luhn 提出一种算法，该算法可以验证信用卡号的有效性。这个算法在确定输入的卡号是否正确，或者这张信用卡是否被扫描仪正确扫描方面是非常有用的。遵

循这个合法性检测可以生成所有的信用卡号，通常称之为 Luhn 检测或者 Mod 10 检测，可以如下描述（为了方便解释，假设卡号为 4388576018402626）：

1) 从左到右对每个数字翻倍。如果对某个数字翻倍之后的结果是一个两位数，那么就将其两位加在一起得到一位数。



2) 现在将第一步得到的所有一位数相加。

$$4+4+8+2+3+1+7+8=37$$

3) 将卡号里从左到右在奇数位上的所有数字相加。

$$6+6+0+8+0+7+8+3=38$$

4) 将第二步和第三步得到的结果相加。

$$37+38=75$$

5) 如果第四步得到的结果能被 10 整除，那么卡号是合法的；否则，卡号是不合法的。例如，号码 4388576018402626 是不合法的，但是号码 4388576018410707 是合法的。

编写程序，提示用户输入一个 long 型整数的信用卡号码，显示这个数字是合法的还是非法的。使用下面的方法设计程序：

```
/** Return true if the card number is valid */
public static boolean isValid(long number)

/** Get the result from Step 2 */
public static int sumOfDoubleEvenPlace(long number)

/** Return this number if it is a single digit, otherwise,
 * return the sum of the two digits */
public static int getDigit(int number)

/** Return sum of odd-place digits in number */
public static int sumOfOddPlace(long number)

/** Return true if the digit d is a prefix for number */
public static boolean prefixMatched(long number, int d)

/** Return the number of digits in d */
public static int getSize(long d)

/** Return the first k number of digits from number. If the
 * number of digits in number is less than k, return number. */
public static long getPrefix(long number, int k)
```

下面是程序的运行示例：（你也可以通过将输入作为一个字符串读入，以及对字符串进行处理来验证信用卡卡号。）

```
Enter a credit card number as a long integer:
4388576018410707
4388576018410707 is valid
```

```
Enter a credit card number as a long integer:
4388576018402626
4388576018402626 is invalid
```

*6.32 (游戏: 赢取双骰子赌博游戏的机会) 修改编程练习题 6.30 使该程序运行 10 000 次, 然后显示赢得游戏的次数。

*6.33 (当前日期和时间) 调用 `System.currentTimeMillis()` 返回从 1970 年 1 月 1 号 0 点开始至今为止的毫秒数。编写程序, 显示日期和时间。下面是运行示例:

```
Current date and time is May 16, 2012 10:34:23
```

*6.34 (打印日历) 编程练习题 3.21 使用 Zeller 一致性原理来计算某天是星期几。使用 Zeller 的算法简化程序清单 6-12 以获得每月开始的第一天是星期几。

6.35 (几何问题: 五边形的面积) 五边形的面积可以使用下面的公式计算:

$$\text{面积} = \frac{5 \times s^2}{4 \times \tan\left(\frac{\pi}{5}\right)}$$

编写一个方法, 使用下面的方法头来返回五边形的面积。

```
public static double area(double side)
```

编写一个主方法, 提示用户输入五边形的边, 然后显示它的面积。下面是一个运行示例:

```
Enter the side: 5.5 Enter
The area of the pentagon is 52.04444136781625
```

*6.36 (几何问题: 正多边形的面积) 正多边形是一个 n 条边的多边形, 它的每条边的长度都相等, 而且所有角的角度也相等 (即多边形既是等边又等角的)。计算正多边形面积的公式是:

$$\text{面积} = \frac{n \times s^2}{4 \times \tan\left(\frac{\pi}{n}\right)}$$

使用下面的方法头编写方法, 返回正多边形的面积:

```
public static double area(int n, double side)
```

编写一个 main 方法, 提示用户输入边的个数以及正多边形的边长, 然后显示它的面积。下面是一个运行示例:

```
Enter the number of sides: 5 Enter
Enter the side: 6.5 Enter
The area of the polygon is 72.69017017488385
```

6.37 (格式化整数) 使用下面的方法头编写一个方法, 用于将整数格式化为指定宽度:

```
public static String format(int number, int width)
```

方法为数字 `number` 返回一个带有一个或多个以 0 作为前缀的字符串。字符串的位数就是宽度。比如, `format(34,4)` 返回 0034, `format(34,5)` 返回 00034。如果数字宽于指定宽度, 方法返回该数字的字符串表示。比如, `format(34,1)` 返回 34。

编写一个测试程序, 提示用户输入一个数字以及宽度, 显示通过调用 `format(number,width)` 返回的字符串。

*6.38 (生成随机字符) 使用程序清单 6-10 中的方法 `RandomCharacter` 打印 100 个大写字母及 100 个一位数字, 每行显示 10 个。

6.39 (几何: 点的位置) 编程练习题 3.32 显示如何测试一个点是否在一个有向直线的左侧、右侧, 或在该直线上。使用下面的方法头编写该方法:

```
/** Return true if point (x2, y2) is on the left side of the
 * directed line from (x0, y0) to (x1, y1) */
public static boolean leftOfTheLine(double x0, double y0,
    double x1, double y1, double x2, double y2)

/** Return true if point (x2, y2) is on the same
 * line from (x0, y0) to (x1, y1) */
public static boolean onTheSameLine(double x0, double y0,
    double x1, double y1, double x2, double y2)

/** Return true if point (x2, y2) is on the
 * line segment from (x0, y0) to (x1, y1) */
public static boolean onTheLineSegment(double x0, double y0,
    double x1, double y1, double x2, double y2)
```

编写一个程序，提示用户输入三个点赋给 p0、p1 和 p2，显示 p2 是否在从 p0 到 p1 的直线的左侧、右侧、直线上，或者线段上。下面是一些运行示例：

```
Enter three points for p0, p1, and p2: 1 1 2 2 1.5 1.5 --Enter
(1.5, 1.5) is on the line segment from (1.0, 1.0) to (2.0, 2.0)
```

```
Enter three points for p0, p1, and p2: 1 1 2 2 3 3 --Enter
(3.0, 3.0) is on the same line from (1.0, 1.0) to (2.0, 2.0)
```

```
Enter three points for p0, p1, and p2: 1 1 2 2 1 1.5 --Enter
(1.0, 1.5) is on the left side of the line
from (1.0, 1.0) to (2.0, 2.0)
```

```
Enter three points for p0, p1, and p2: 1 1 2 2 1 -1 --Enter
(1.0, -1.0) is on the right side of the line
from (1.0, 1.0) to (2.0, 2.0)
```