这里要做一点补充说明。首先，你定义了两个 Fraction 对象：aFraction 和 bFraction，并将它们的值分别设为 1/4 和 1/2，然后又定义了名为 resultFraction 的一个 Fraction，这个变量将用来存储相加操作的结果。

以下代码

```
resultFraction = [aFraction add: bFraction];
```

首先发送 add:消息到 aFraction 类，同时将类 Fraction bFraction 作为它的参数。

在这个方法里，新的 Fraction 对象被创建并且加出了结果，该结果存储在通过方法返回的 Fraction 对象 result 中，然后将它保存在变量 resultFraction 中。你可能注意到你并未在 main 中为 resultFraction 创建或初始化一个 Fraction 对象。这是因为 add:方法中已经创建了一个对象，并通过方法返回一个对象的引用，这个引用存储在 resultFraction 中。所以，resultFraction 最终保存了一个 Fraction 的对象引用，该对象是在 add:方法中创建的。这一段是很重要的！在你完全理解它之前，值得反复回顾。

7.7.1 扩展类的定义和接口文件

你可能不需要处理分数，但是这些例子告诉你如何通过加入新方法来定义和扩展一个类。其他处理分数的人将使用你的 Fraction.h 接口文件，并且使用这个文件足够编写他自己的处理分数的程序。如果他需要添加新方法，通过直接扩展类定义或者定义自己的子类并添加自己的新方法直接扩展该类，可以实现该目的。下一章将学习如何实现它。

7.8 练习

1. 将下列方法加到 Fraction 类，以扩展关于分数的算术运算。在每个例子中都约简结果。

```
// 减去消息接收者的参数
-(Fraction *) subtract: (Fraction *) f;
// 消息接收者乘以参数
-(Fraction *) multiply: (Fraction *) f;
// 消息接收者除以参数
```

```
-(Fraction *) divide: (Fraction *) f;
```

2. 从 Fraction 类中修改 Print 方法，使之能够接收一个可选的 BOOL 参数，它表明是否应该约简分数并显示它。如果要约简它（参数为 YES），一定不要对它进行永久更改。
3. Fraction 类对负分数适用吗？例如，-1/4 和-1/2 能得出正确结果吗？如果想出答案，编写测试程序进行尝试。
4. 修改 Fraction 类的 print 方法，以便显示比 1 大的分数，例如，5/3 能显示为 1 2/3。
5. 从代码 7.2 中移除@synthesize 指令并修改程序使新的变量命名通过编译。
6. 第 4 章“数据类型和表达式”练习 6，为复数创建一个新类名为 Complex。创建一个新的方法名为 add:，用来求两个复数之和。要将复数相加，你只需要分别对实数部分和虚数部分求和，如下：

```
(5.3 + 7i) + (2.7 + 4i) = 8 + 11i
```

add:方法可以存储并返回一个新的复数，基于如下方法描述：

```
- (Complex *) add: (Complex *) complexNum;
```

7. 第 4 章练习 6 的 Complex 类在这一章的练习 6 中进行了扩展，请单独创建接口文件 Complex.h 和实现文件 Complex.m。再创建一个单独的测试程序文件进行测试。