如果不覆写方法，则

```
Square *mySquare = [[Square alloc] init];
```

分配了一个新的 Square 对象，但是没有为存储在实例变量中的 Rectangle 对象 rect 分配存储空间。

一个解决方案就是覆写 init 或添加 initWithSide:之类的新方法来分配空间。这个方法可以为 Rectangle rect 分配存储空间，并相应地设置它的边。

在 Square 类中定义方法时，可以使用 Rectangle 的方法。例如，下面说明了如何实现 area 方法：

```
-(int) area
{
  return [rect area];
}
```

其他方法的实现作为练习留给大家（参见本章练习 5）。

11.5 练习

1. 扩展代码清单 11-1 中的 MathOps 分类，使之包含一个 invert 方法，这个方法返回一个 Fraction，它是接收者的倒置。
2. 向类 Fraction 添加一个名为 Comparison 的分类。根据以下声明，在这个分类中添加两个方法：

   ```
   -(BOOL) isEqualTo: (Fraction *) f;
   -(int) compare: (Fraction *) f;
   ```

   如果两个分数相同，第一个方法应该返回 YES；否则，返回 NO。注意分数的比较方式（如，比较 3/4 和 6/8 应当返回 YES）。

   如果接收者小于参数传递来的分数，则第二个方法应当返回-1；如果二者相等，应返回 0；如果接收者大于参数，则应当返回 1。
3. 通过添加遵守非正式协议 NSCompairsonMethods（本章前面所列出的）的方法来扩展 Fraction 类。根据该协议实现前 6 个方法（isEqualTo:、isLessThanOrEqualTo:、isLessThan:、isGreaterThanOrEqualTo:、isGreaterThan:、isNotEqualTo:），并测试它们。

4. 函数 sin()、cos()和 tan()是 C 标准库的一部分（与 scanf()一样)。这些函数在系统头文件<math.h>中声明了，当导入 Foundation.h 时，这些就会被自动导入到你的程序中。

这些函数分别可以用来计算用弧度表示的 double 参数的 sine、cosine 或者 tangent 值返回的结果，也是一个双精度的浮点值。所以，

```
result = sin (d);
```

用于计算 d 的 sine 值，角度 d 的值用弧度表示。为第 6 章“选择结构”中的 Calculator 类添加一个名为 Trig 的分类。根据以下声明，为这个分类添加一些方法来计算 sine、cosine 和 tangent 的值：

```
-(double) sin;
-(double) cos;
-(double) tan;
```

5. 根据本章对合成对象的讨论以及以下接口部分：

```
@interface Square: NSObject
-(instancetype *) initWithSide: (int) s;
-(void) setSide: (int) s;
-(int) side;
-(int) area;
-(int) perimeter;
@end
#import "Rectangle.h"
@implementation Square
{
    Rectangle   *rect;
}
// 这里插入 Square 的方法
...
@end
```

编写 Square 的实现部分，以及用来检验其方法的测试程序。注意：记住需要覆写 init 方法，initWithSide:将作为重新设计过的初始化方法。