2、动态方法解析

通常而言，对象在接收到消息后，会通过IMP去找到并执行响应方法，而动态方法解析则是会在消息启动转发机制之前执行。具体表现在当对象接收到未知的消息时，先会调用所属的类方法或者实例方法。在这个方法中我们可以为未知的消息添加处理方法，当然了，我们所添加的处理方法肯定是我们已经实现的，至于如何添加，我们会在后边讲解。

```
+ (BOOL)resolveClassMethod:(SEL)sel OBJC_AVAILABLE(10.5, 2.0, 9.0, 1.0);
+ (BOOL)resolveInstanceMethod:(SEL)sel OBJC_AVAILABLE(10.5, 2.0, 9.0, 1.0);
```

```
+ (BOOL)resolveInstanceMethod:(SEL)sel
{
    NSString *method1 = NSStringFromSelector(sel);
    if ([method1 isEqualToString:@"getC"]) {
        class_addMethod([self class], @selector(printText),(IMP)textMethod , "v@:");
    }
    return YES;
}
```

如果有的方法在解析这一步不想进行处理，系统会自动将方法选择器传送到消息转发机制，此时需要上述方法返回为NO

3、消息转发

(1).重定向

通过前边的动态方法解析之后，会走到这一步。当然，在消息转发之前其实还有一步，被称之为消息重定向。即我们通过重写一个方法去替换掉原本该消息的接收者，将消息重定向到另一个对象。重写方法为:

```
- (id)forwardingTargetForSelector:(SEL)aSelector{
    if(aSelector == @selector(printText)){
        return _otherPerson;
    }
    return [super forwardingTargetForSelector:aSelector];
}
```

在这个方法中传递进来的是发送的消息，返回的是要重定向到的对象，并记得在最后调用super的方法，除此之外，要重定向的对象也必须要实现该消息。