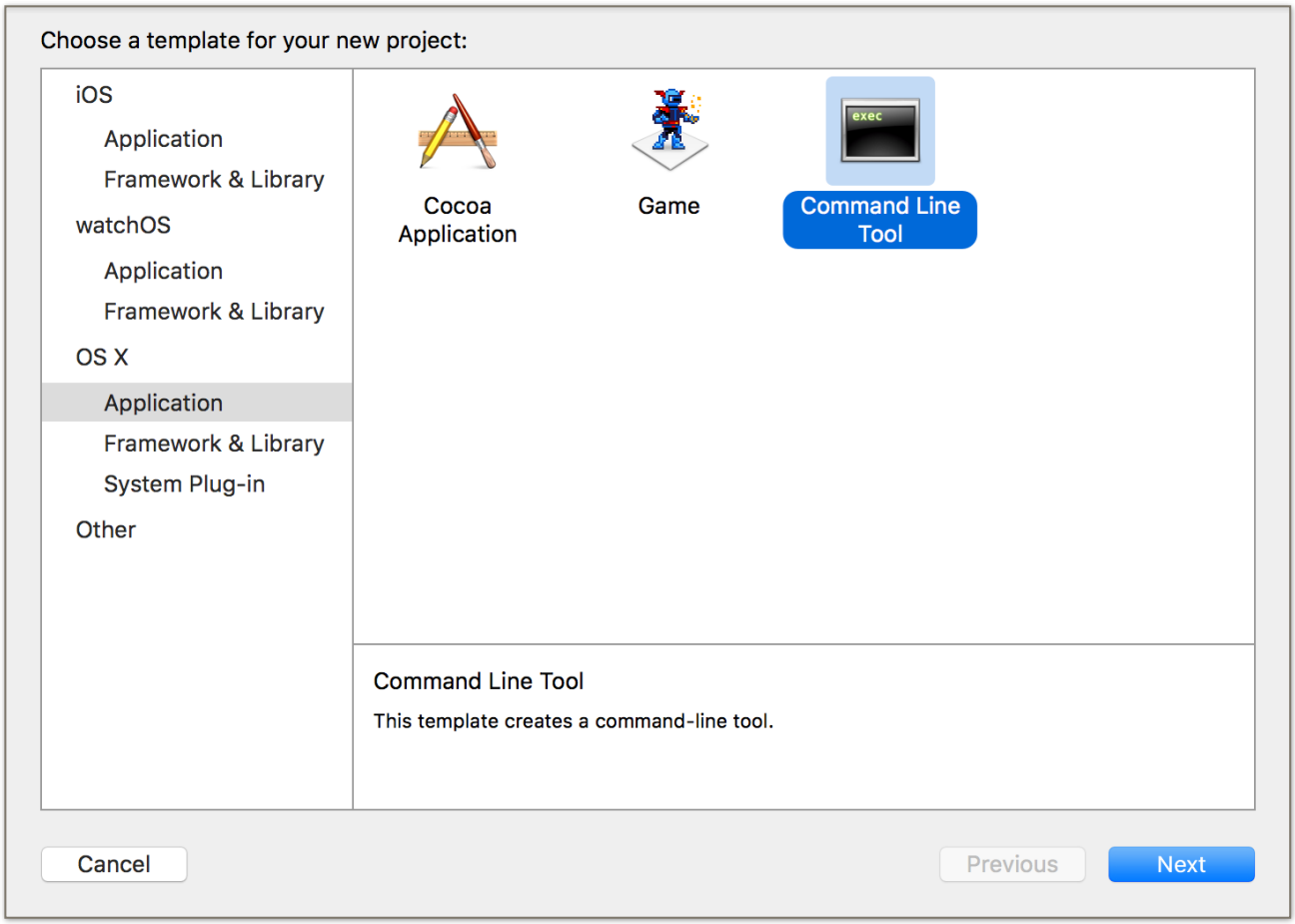
Hello, iOS
4 Objective-C程序结构[1]

这一节开始介绍和Objective-C相关的代码知识。我们默认读者已经阅读了之前的文章，了解面向对象编程的基本知识，并且具有初步的C语言能力。让我们从代码的程序结构开始谈起。

对于Objective-C程序，与java语言类似的，一般以类作为结构基础。这其中包括了**类的接口**、**类的实现**和**实例程序**三个部分。一般地，这三个部分会分别存在于.m,.h和主文件当中，但是根据需要也可以放在同一个文件中。我们以实际的代码案例[2]来说明。这一次我们将设计一个OS X命令行程序，选择Command Line Tool，之后的步骤与以往相似。

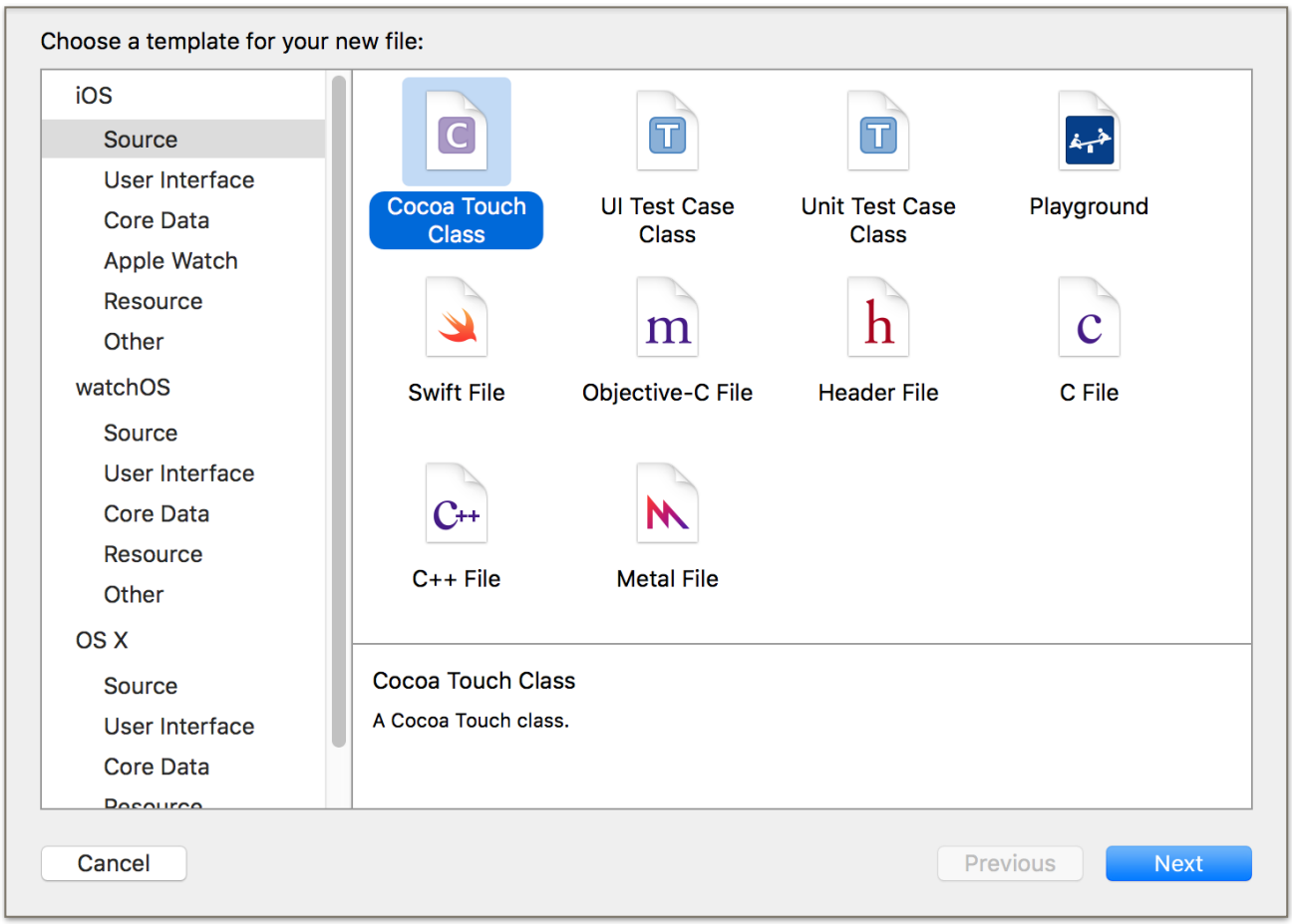


新建工程之后，我们向文件部分添加新的类，右键选择New File，然后选择Cocoa Touch Class，并选择其为NSObject的子类(Subclass Of栏选择NSObject)。我们把这个类叫做Student。

---

[1] 2015年9月2日，基于OS X 10.11 beta **8**， iOS 9 beta 5， Xcode 7 beta 5.

[2] 本节的样例项目名为Demo4。



我们为学生类定义一些属性和方法，代码如图[3]。

```
9  #import <Foundation/Foundation.h>
10
11 @interface Student : NSObject {
12     NSString *name;
13     NSInteger age;
14 }
15
16 - (NSString *)name;
17 - (NSInteger)age;
18 - (void)setName:(NSString *)input;
19 - (void)setAge:(NSInteger)input;
20
21 @end
22
```

我们为这个类定义了2个属性和4个方法。属性定义部分，语法和C语言类似，而且如果你看过了前面的文章也应该了解，这两个类型分别表示字符串和整数类型。下面是4个方法的定义，以-开始的是对象方法，也就是这个类实例化的对象可以使用的方法，而+开头的是类方法，即这个类可以直接使用的方法。其中以void开头的表示没有返回值的方法。

注意第11行，这里是接口的声明部分。冒号前是这个类的名字，以大写字母开头，并采用驼峰命名法。冒号后是这个类的父类，一般在iOS开发中，不知道去哪里找父类的类就默认NSObject是它的父类(比较专业的说法是NSObject是系统类)。

---

[3] 这里我们采取了一种较为简单的定义方法。如果你有java的编程经验，会想到为每个属性设计getter和setter方法。但是在Objective-C中，我们一般不用为其单独设计这两种方法，而是直接用点语法访问并读写的，样例中这样写是为了后面的代码更加直观设计的。

```
9  #import "Student.h"
10
11 @implementation Student
12
13 - (NSString *)name {
14     return name;
15 }
16
17 - (NSInteger)age {
18     return age;
19 }
20
21 - (void)setName:(NSString *)input {
22     name = input;
23 }
24
25 - (void)setAge:(NSInteger)input {
26     age = input;
27 }
28
29 @end
30
```

接下来我们设计实现部分的代码。需要注意的是，为了逻辑的合理性和封装的要求，只要声明了的方法，就一定要在实现部分对其进行实现。

只要你对C语言的函数有所了解，就可以看懂在21行和25行的两个方法里，分别把传入的值赋给原来的变量(属性)了。

还有一个例外是如果你学过C++的话，每次构造都有对应的析构函数，在iOS的开发过程中，一般省略这个步骤。苹果在iOS 5后引入了ARC机制[4]，未来会讲到这个不知道方便了多少程序员的好东西。

现在我们在主函数中，测试学生类的各项功能。

```
9  #import <Foundation/Foundation.h>
10 #import "Student.h"
11
12 int main(int argc, const char * argv[]) {
13     @autoreleasepool {
14         Student *stu = [[Student alloc] init];
15         [stu setName:@"李狗蛋"];
16         [stu setAge:15];
17         NSLog(@"学生姓名: %@", [stu name]);
18         NSLog(@"学生年龄: %ld", (long)[stu age]);
19     }
20     return 0;
21 }
22
```

第10行，我们引入学生类的文件，注意Objective-C中的引入关键字是import，与C语言的include有所区别。第13行，autoreleasepool块中的所有内容均遵守ARC机制，如果理解有困难可以先跳过。14行实例化了一个学生类的对象，alloc是内存申请，init是初始化。

在Objective-C中，方法的使用我们之前曾经提到过，格式是用中括号包起来谁去做什么。可以看到15、16两行分别执行了设置名字和设置年龄两个方法[5]。需要注意的是在Objective-C中，涉及字符的操作前都带有一个@符号。17、18两行中的NSLog是系统打印日志的意思，其使用方法与C语言中的printf有相似的地方，但是它同时还可以表示出应用程序本身的情况。17行，NSString用%@输出。18行，出于安全性的考虑。NSInteger在打印的时候推荐转换成long，然后用%ld来输出。

程序运行的结果如图所示，可以看出log出的打印内容和printf出的内容有些许不同。

```
2015-09-02 06:02:33.601 Demo4[1199:286752] 学生姓名: 李狗蛋
2015-09-02 06:02:33.602 Demo4[1199:286752] 学生年龄: 15
Program ended with exit code: 0
```

[4] Automatic Reference Counting，自动引用计数器，作用是自动释放没有被引用(指针指向)的内存。

[5] 也可以直接用点语法直接设置两个属性，即：

```
stu.name = @“李狗蛋"; stu.age = 15;
```

下面我们简单总结一下Objective-C的语法格式：

接口：
@interface 类名: 父类名 {
变量的定义;
}
方法的定义;
@end

实现：
@implementation 类名
方法的实现
@end

对于方法，我们经常会见到多参的方法，定义的格式是**[接收者 方法名1:参数1 方法名2:参数2 … 方法名n:参数n];**。这个方法的名称是“方法名1:方法名2:…方法名n”。

Objective-C中大量使用了@符号，我简单总结了@的用法如下表：

指令	含义	举例
@“char”	定义字符串常量	@“李狗蛋”
@class ClassName	定义一个类	@class Student
@defs(class)	返回结构的类型列表	struct ClassA {@defs(ClassB)}
@autoreleasepool	创建自动释放池	@autoreleasepool {}
@end	一个部分的结束	@end
@implementation	类的实现	@implementation ClassA
@interface	类的接口	@interface ClassA: NSObject
@private	私有域	@private {int num;}
@protected	保护域	@protected {int num;}
@public	公开域	@public {int num;}
@property (list)name	属性的定义	@property (nonatomic)int age;
@protocol	创建协议对象	@protocol (Copying) {…} if ([ClassA conformsTo:(protocol)])
@protocol name	创建协议	@protocol Copying
@selector(method)	选择对象	if ([ClassA respondTo:@selector(alloc)])
@synchronized(object)	定义一个同步，在某一时刻，它只能被一个线程占用	@synchronized(self)
@synthesize name	提供getter/setter方法	@synthesize name;
@try	捕获异常	@try {NSString *name;}
@catch(exception)	抓取(处理)异常	@catch(NSException *exc) {…}
@finally	无论是否抛出异常都会执行	@finally {[name release];}
@throw	抛出异常	@throw exc;

Objective-C和其他语言类似保留了许多关键字[6]。同时还有一些已定义的标识符，如下表：

标识符	含义
_cmd	方法内自动定义的本地变量，包含本方法的选择程序
func	在函数或方法内自动定义的的本地字符串变量，包含本函数或方法的名称
BOOL	布尔值，一般赋值为YES或NO
Class	类对象类型
id	通用对象类型
IMP	指向了返回id类型值的方法的指针
nil	空对象
Nil	空类对象
NO	(BOOL)0
NSObject	<Foundation/NSObject.h>中定义的所有类的根类
Protocol	存储协议信息的类的名称
SEL	已经编译的选择程序
self	本类
super	父类
YES	(BOOL)1

这一节，我们了解了Objective-C语言的基本结构以及基本信息，下一节我们将更深入地研究这种语言的特性。因为这一节的概念非常多，所以阅读上可能会有些困难，或者某些内容可能在课内知识更加丰富之后才会有更深刻的理解。同时也因为本节列举的信息比较多，由于本人知识有限，有错误也请多多批评指出。

第4节 EOF

[6] Objective-C保留的关键字有： _Bool, _Complex, _Imaginary, auto, break, bycopy, byref, case, char, const, continue, default, do, double, else, enum, extern, float, for, goto, if, in, inline, inout, int, long, oneway, out, register, restrict, return, self, short, signed, sizeof, static, struct, super, switch, typedef, union, unsigned, void, volatile, while.