CS 696 Mobile Application Development
Fall Semester, 2010
Doc 2 Objective C - Basics
Aug 31, 2010

References

The Objective-C 2.0 Programming Language, http://developer.apple.com/iphone/library/documentation/Cocoa/Conceptual/ObjectiveC/Introduction/introObjectiveC.html#//apple_ref/doc/uid/TP30001163

Objective-C Hello World

```
#import <Cocoa/Cocoa.h>

int main(int argc, char *argv[])
{
	NSLog(@"Hello World!");
	return 0;
}
```

History

Early 1980's

Brad Cox & Tom Love combine C and Smalltalk messaging

Goal - software components

1986 - Objective-C book published

1988 - NeXT uses Objective-C to implement NeXTstep user interface

1996 - Apple purchase NeXT,

Objective-C becomes bases for Mac OS X

2007 - iPhone OS written in Objective-C

2010 Aug - TIOBE index ranks Objective-C 9'th in popularity

TIOBE programming popularity index: http://www.tiobe.com/index.php/content/paperinfo/tpci/index.html

Objective-C Overview

- Strict superset of C
    - C programs are legal Objective-C programs
    - Apple's Objective-C support C++
- Single Inheritance
- Protocols (java interfaces)
- Categories (Extending classes)
- Properties
- Smalltalk messaging syntax
- Exception Handing
- Dynamic runtime
- Objects created on heap
- Reflection
- Blocks

Syntax Additions to C

Anonymous object

Classes

Selectors

Message expressions

Protocol, Category syntax

Objective-C Message Syntax

Java

```
Rectangle sample = new Rectangle();
sample.setWidth(4);
sample.setHeight(5);
sample.setHeightWidth(4,5);
int area = sample.area();
```

Objective-C

```
Rectangle * sample = [[Rectangle alloc] init];
[sample setWidth:4];
[sample setHeight:5];
[sample setHeight:5 width: 4];
int area = [sample area];
[sample release];
```

Message Syntax

```
[receiver message]
[receiver message: argument]
[receiver message: arg1 and: arg2]
[receiver message: arg1 and: arg2 with: arg3]
```

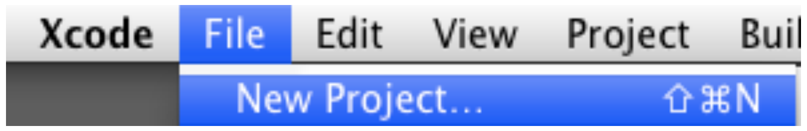
Base Date Types

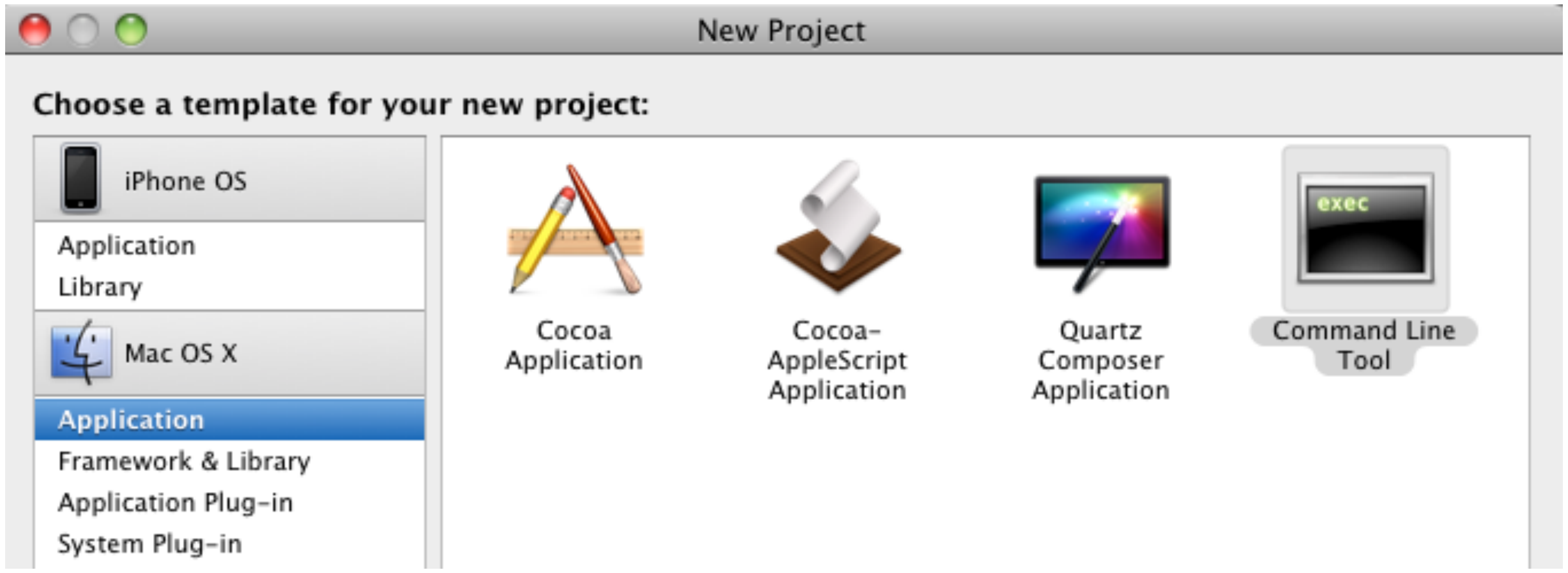
int	at least 16 bits
short int	smaller than int; at least 16 bits
long int	at least 32 bits
long long int	at least 64 bits
unsigned int	at least 16 bits
float	at least 6 digits of precision
double	at least 10 digits of precision
long double	at least 10 digits of precision
char	Single character
unsigned char	
signed char	
BOOL	0, 1, TRUE, FALSE, YES, NO
float _Complex	Complex number
double _Complex	Extended accuracy complex number
long double _Complex	Extra-extended accuracy complex number
void	

BOOL

```
BOOL isHome = YES;
isHome = NO;
isHome = FALSE;
isHome = 0;
isHome = TRUE;
isHome = 1;
if (isHome)
    NSLog(@"home");
else {
    NSLog(@"away");
}
```

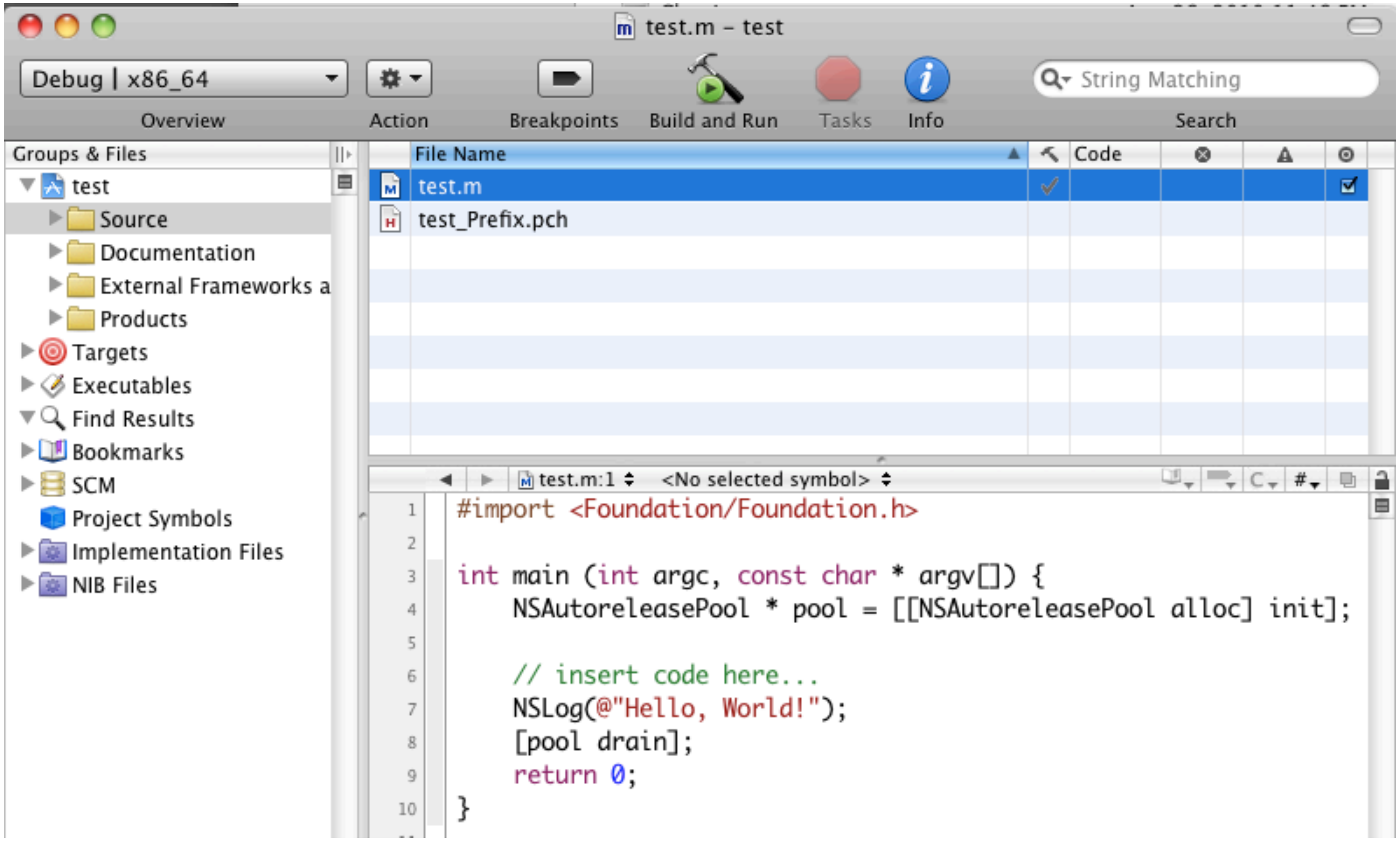
How to Run Programs





Choose Command Line tool to write straight Objective-C code. When we start iPhone applications we will use a different template for projects.

Xcode Editor



```
#import <Foundation/Foundation.h>

int main (int argc, const char * argv[]) {
    NSAutoreleasePool * pool = [[NSAutoreleasePool alloc] init];

    // insert code here...
    NSLog(@"Hello, World!");
    [pool drain];
    return 0;
}
```

Strings

NSString	C String
`NSString* greeting = @"Hi mom";`	`char* greeting = "Hi Dad";`
Objective-C Class in Foundation framework Used in iPhone development	Objective-C array of char

Where are the docs for NSString?

Search for NSString in Xcode

Google for NSString

    or go to Apple Dev center and find NSString class

Read String Programming Guide

Creating Strings

```
NSString * start = @"Start";
NSString * all = [start stringByAppendingString:@" and the rest"];
NSLog(all);
```

Formatting Strings

```
NSString * formatted = [NSString stringWithFormat:@"Name: %@, Age: %f",
                                                  @"Sam", 12.3];

//Name: Sam, Age: 12.300000
```

Some formats

%@	Object
%d, %i	signed int
%u	unsigned int
%f	float/double
%x, %X	hexadecimal int
%o	octal int
%zu	size_t
%p	pointer
%e	float/double (in scientific notation)
%g	float/double (as %f or %e, depending on value)
%s	C string (bytes)
%S	C string (unichar)
%c	character
%C	unichar
%lld	long long

See "String Format Specifiers" in Apples "String Programming Guide"

NSLog uses formatting

```
float fromString = [@"  123.45  " floatValue];
NSLog(@"Result: %f", fromString);
```

Output

```
2010-08-21 16:42:38.129 examples[29557:a0b] Result: 123.449997
```

Reading From File

names.txt

```
First: Roger,Last: Whitney
First: Sam,Last: Spade
```

```
NSString *path = @"/Users/whitney/Desktop/names.txt";
NSError *error;
NSString *stringFromFileAtPath = [NSString
                        stringWithContentsOfFile: path
                        encoding: NSUTF8StringEncoding
                        error:&error];
if (stringFromFileAtPath == nil) {
     NSLog(@"Error reading file at %@\n%@",
     path, [error localizedFailureReason]);
}
NSLog(@"Contents:%@", stringFromFileAtPath);
```

Scanning a String

names.txt

```
First: Roger,Last: Whitney
First: Sam,Last: Spade
```

```
NSScanner *theScanner;
NSString *firstName;
NSString *lastName;

theScanner = [NSScanner scannerWithString: stringFromFileAtPath];

while ([theScanner isAtEnd] == NO)
    {
    if ([theScanner scanString: @"First:" intoString: NULL] &&
        [theScanner scanUpToString: @","  intoString: &firstName] &&
        [theScanner scanString: @"," intoString: NULL] &&
        [theScanner scanString: @"Last:" intoString: NULL] &&
        [theScanner scanUpToString: @"\n" intoString: &lastName] )
        {
        NSLog(@"First: %@: Last: %@", firstName, lastName );
        }
    }
```

stringFromFileAtPath is from last slide

Derived Data Types

```
int single[] = { 0, 1, 2 }
int byOrder[]  = {[2] = 4, [5] = 1, [0] = 8 }
int matrix[3][2] = { {1, 2} , {4, 5}, {6, 7 }}


struct point { x; y } corner = { 10, 20 };


union overlap { int integer; float floater } example = { 10 }


enum Days{Sunday,Monday,Tuesday,Wednesday,Thursday,Friday,Saturday} exam;
exam = Monday;
```

Standard C Control Structures

for
do
while
if
switch

nil

Java's null that responds to messages

```
Rectangle* test = nil;
int area = [test area];
// runs without error
```

Rules

message returns	[nil message] returns
object	nil
pointer, int, long, double (numberic types)	0
struct in register	all values 0.0