.h files (Headers)

Defining an interface for a class

Syntax:

```
#import <someOtherHeader.h>
@interface MyClass : ParentClass {
      varType1 varName1;
      varType2 varName2;
}
@property(nonatomic, retain) varType1 varName1;
-(return type) methodName1:(paramType) paramName;
-(return type) methodName2:(t1) p1 moreName:(t2)p2;
@end
```

Explanation:

`#import <someOtherHeader.h>`
Imports another class's header file. Note that <> is used for standard headers, whereas "" is used for headers that you write.

`@interface MyClass : ParentClass {`
Declares a class named MyClass ,which extends class ParentClass

`varType1 varName1;`
Declares a variable as a member of MyClass

`@property(nonatomic, retain) varType1 varName1;`
Declares methods called getVarName1() and setVarName1()

`-(return type) methodName1:(paramType) paramName;`
Declares a method called methodName with a single parameter.
Note that Objective-C method syntax is significantly different from Java/C. A method's "name" is interleaved with its parameters. For instance, the "name" of the last method above is:

```
methodName2:moreName:
```

The parameters to a method are inserted after the colons. More info on this is contained in Messages.

`@end`
Ends the `@interface` block.

Full Example:

```
#import <UIKit/UIKit.h>
@interface webBrowserClass : UIViewController {
      UIWebView * myWebView;
      UIButton * goButton;
      UIButton * backButton;
      UITextField * address;
      UILabel * textOutput;
}

@property(nonatomic, retain) IBOutlet UIWebView * myWebView;
@property(nonatomic, retain) IBOutlet UIButton * goButton;
@property(nonatomic, retain) IBOutlet UIButton * backButton;
@property(nonatomic, retain) IBOutlet UITextField address;
@property(nonatomic, retain) IBOutlet UILabel * textOutput;

-(IBAction) backButtonWasPressed;
-(IBAction) goButtonWasPressed;
-(NSString) processUrl:(NSString*)theUrl;
@end
```

Notes:

`IBOutlet` is a special keyword that allows integration with interface builder, it defines a variable as an "outlet", which can be linked with UI elements in interface builder.

`IBAction` is a special keyword that allows integration with interface builder, it defines a method as an "action", which can be linked with events being sent from UI elements in interface builder. It is set as the return type of the method, however the return type is actually `void`.

.m files

Implementing a class

Syntax:

```
#import “myClass.h”

@implementation myClass
@synthesize varName1;
-(return type) methodName1:(paramType) paramName {
      ...
      ...
      ...
}
-(return type) methodName2:(t1) p1 moreName:(t2)p2 {
      ...
      ...
      ...
}
@end
```

Explanation:

`#import “myClass.h”`
Imports the definition of `myClass`, as defined in the previous page.

`@implementation myClass`
Tells the compiler that the following source code is implementing `myClass`.

`@synthesize varName1;`
For every `@property` in your .h file, a corresponding `@synthesize` must be put into the .m file. The `@property` macro declares getter/setter methods, whereas the `@synthesize` macro implements those methods.

`-(return type) methodName1:(paramType) paramName {`
Method implementations follow the same syntax rules as Method Declarations.

`@end`
Ends the `@implementation` block.

Full Example:

```
#import “webBrowserClass.h”

@implementation webBrowserClass

@synthesize myWebView;
@synthesize goButton;
@synthesize backButton;
@synthesize address;
@synthesize textOutput;

-(IBAction) backButtonWasPressed {
      [myWebView goBack];
}

-(IBAction) goButtonWasPressed {
      NSString * theAddress = [address getText];
      theAddress = [self processUrl:theAddress];
      NSURL *url = [NSURL URLWithString:theAddress];
      NSURLRequest req = [NSURLRequest requestWithURL:url];
      [myWebView loadRequest:req];
}

-(NSString *) processUrl:(NSString*)theUrl {
      return @”http://” + theUrl;
}
@end
```

Notes:

No `return` statements are necessary as the `IBAction` defines the methods as `void` return type.

Variables do not have to be defined as a member of a class, local variables can be defined as well, as is shown in `goButtonWasPressed`. `theAddress` is a local variable, and will be garbage collected at the end of the method.

The `processUrl` method simply adds an “http://” to the beginning of the url. In Objective-C, the “String” class is called `NSString`. `NSString` constants are created by preceding a string with an “@” symbol. For example; `@”String Constant”`

Messages (aka methods, functions):
Sending Messages to Objects

Syntax:
```
[object message];
[object interleavedMessage:param1 continuedMessage:param2 ];
[object1 message1:[object2 message2]];
```

Explanation:
To call a method in Objective-C, one uses the square brackets with the object and message names inside.

Method names in Objective-C are different from other languages, in that method names are interleaved with the parameters given to that method. For instance, the name of the second message above is:
`interleavedMessage:continuedMessage:`

However, the parameters are inserted after the colons, resulting in;
`interleavedMessage:param1 continuedMessage:param2`

Equivalent syntax in Java/C++ would look like;
object.message(param1, param2);

Declaration of methods inserts not only the name of parameters, but also the return type and types of variables. For example;

```
-(void) methodName1:(int)param1;

-(void) methodName2:(int)param1 withParam:(NSString*)param2;
```

Misc.
Random stuff that might be useful.

To output a string to the debugger console, use `NSLog( NSString * );`
Again, `NSString` constants are created via the `@"String Constant"` syntax.
Example: `NSLog( @"This prints out!" );`

To open the debugger console, in Xcode, press APPLE-SHIFT-R. This is useful not only for viewing the output from your program, but when (not if, unfortunately) your program crashes, or even performs unexpectedly, the debugger console may contain useful information.

Many, many, classes start with the letters “NS”. This is for historical reasons, but if you are trying to find a class to do something, slap NS on the beginning.