OBJECTIVE-C EXTENSIONS

A class extension bears some similarity to a category, but it can only be added to a class for which you have the source code at compile time *theclassiscompiledatthesametimeastheclassextension*.

The methods declared by a class extension are implemented in the implementation block for the original class, so you can't, for example, declare a class extension on a framework class, such as a Cocoa or Cocoa Touch class like NSString..

Extensions are actually categories without the category name. It's often referred as **anonymous categories**.

The syntax to declare a extension uses the @interface keyword, just like a standard Objective-C class description, but does not indicate any inheritance from a subclass. Instead, it just adds parentheses, as shown below

```
@interface ClassName ()

@end
```

Characteristics of extensions

- An extension cannot be declared for any class, only for the classes that we have original implementation of source code.
- An extension is adding private methods and private variables that are only specific to the class.
- Any method or variable declared inside the extensions is not accessible even to the inherited classes.

Extensions Example

Let's create a class SampleClass that has an extension. In the extension, let's have a private variable internalID.

Then, let's have a method getExternalID that returns the externalID after processing the internalID.

The example is shown below and this wont work on online compiler.

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

@interface SampleClass : NSObject
{
    NSString *name;
}

- (void)setInternalID;
- (NSString *)getExternalID;

@end


@interface SampleClass()
{
    NSString *internalID;
}

@end

@implementation SampleClass

- (void)setInternalID{
    internalID = [NSString stringWithFormat:
    @"UNIQUEINTERNALKEY%dUNIQUEINTERNALKEY",arc4random()%100];
}

- (NSString *)getExternalID{
    return [internalID stringByReplacingOccurrencesOfString:
    @"UNIQUEINTERNALKEY" withString:@""];
}

@end

int main(int argc, const char * argv[])
{
    NSAutoreleasePool * pool = [[NSAutoreleasePool alloc] init];
    SampleClass *sampleClass = [[SampleClass alloc]init];
    [sampleClass setInternalID];
    NSLog(@"ExternalID: %@",[sampleClass getExternalID]);
    [pool drain];
    return 0;
}
```

Now when we compile and run the program, we will get the following result.

```
2013-09-22 21:18:31.754 Extensions[331:303] ExternalID: 51
```

In the above example, we can see that the internalID is not returned directly. We here remove the UNIQUEINTERNALKEY and only make the remaining value available to the method getExternalID.

The above example just uses a string operation, but it can have many features like encryption/decryption and so on.