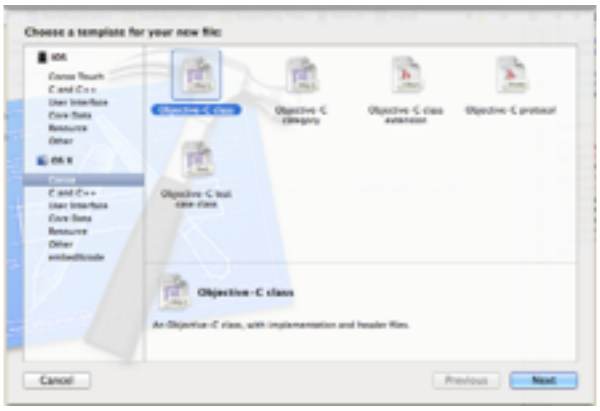
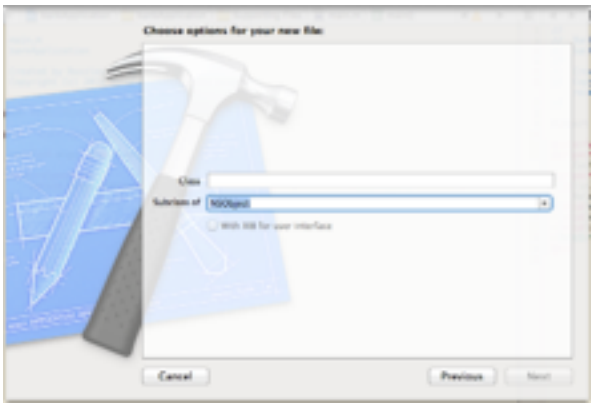
Using Object Oriented tools and techniques

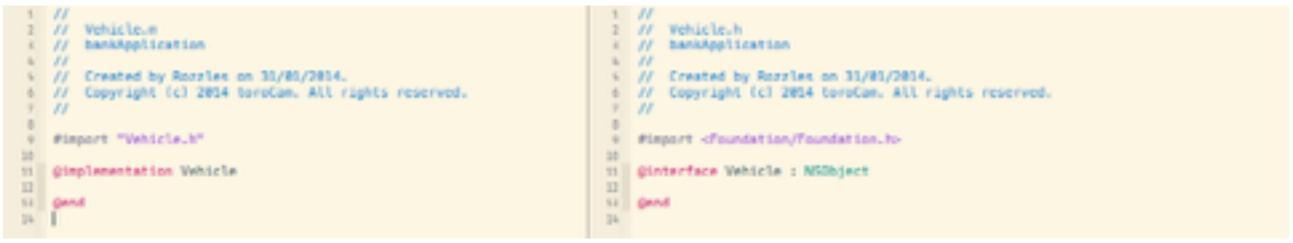
This is a brief description on how to implement Object Oriented tools and techniques in *'Objective C'* using the Xcode IDE.
In Objective C the superclass root of almost all classes is NSObject. Through NSObject, objects inherit a basic interface to the runtime system and the ability to behave as Objective-C objects. To create a simple class in Objective C click File > New > File and select Objective C class.
The next dialog will ask you what your new class to be a Subclass of, the default is NSObject



which contains the most basic requirements of an Objective C class so unless you have any specific need to be a subclass of something else click next to make your Superclass.
Create the elements you'd like all your subclasses to contain - for this example I'll use a Vehicle as


Choose options for your new file:
Class
Subclass of
NSObject
With XIB for user interface
Cancel
Previous
Next

an example.
This gives me a simple Vehicle class Interface and Implementation file.

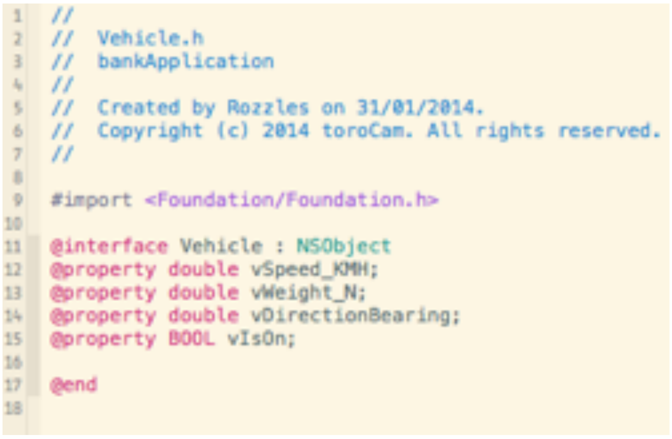


```
//
//  Vehicle.m
//  bankApplication
//
//  Created by Rozzles on 31/01/2014.
//  Copyright (c) 2014 toroCom. All rights reserved.
//

#import "Vehicle.h"

@implementation Vehicle

@end
```

```
//
//  Vehicle.h
//  bankApplication
//
//  Created by Rozzles on 31/01/2014.
//  Copyright (c) 2014 toroCom. All rights reserved.
//

#import <Foundation/Foundation.h>

@interface Vehicle : NSObject

@end
```

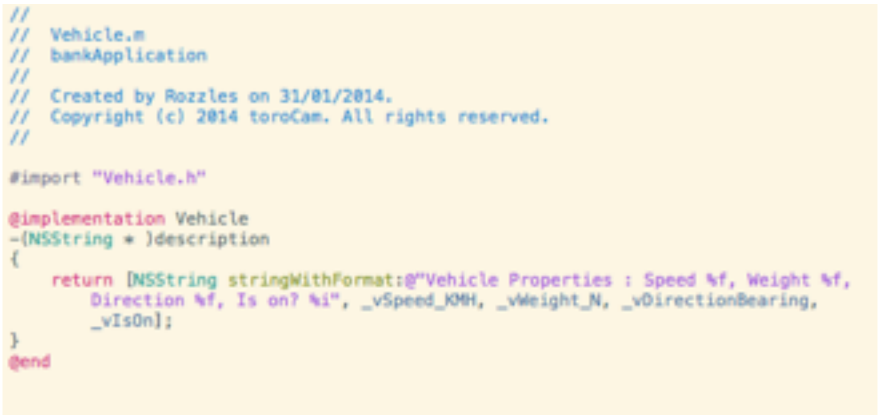
I need to use another Object Oriented concept to describe what all vehicles have on common - this is called Abstraction in which I focus on the bare essentials of a Vehicle that all vehicles have in common - Trucks, Lorries, Cars or Planes.
All my vehicles move so I need to define the **speed** of the vehicle
All my vehicles have a **weight** so I need to define that
All my vehicles travel in a **direction**, so I should include this.

All my vehicles have a state of being **Switched on or Switched off** - however not all of them have engines ect. so I should define this further down the hierarchy.
Given this I may write the class like so:



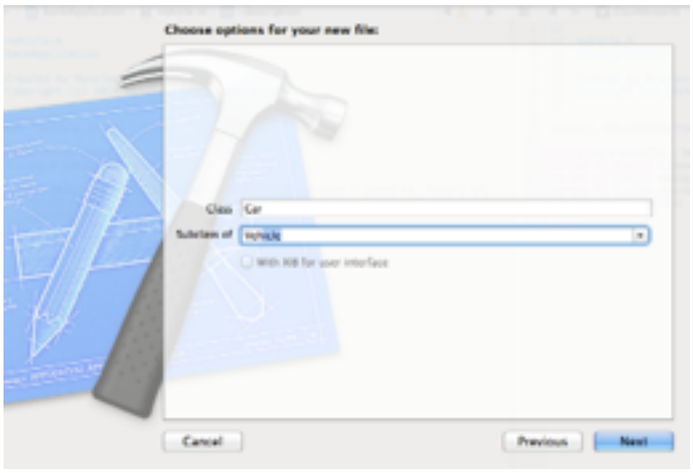
```
//
//  Vehicle.h
//  bankApplication
//
//  Created by Rozzles on 31/01/2014.
//  Copyright (c) 2014 toroCam. All rights reserved.
//

#import <Foundation/Foundation.h>

@interface Vehicle : NSObject
@property double vSpeed_KMH;
@property double vWeight_N;
@property double vDirectionBearing;
@property BOOL vIsOn;

@end
```

Now every subclass of Vehicle has these properties.
Say I wanted all of my vehicles to share common methods too, for instance I want my vehicles to describe themselves by their properties unlike how their superclass -NSObject describes them with an address.
I could override the description method of NSObject to provide a more useful description like so:



```
//
//  Vehicle.m
//  bankApplication
//
//  Created by Rozzles on 31/01/2014.
//  Copyright (c) 2014 toroCam. All rights reserved.
//

#import "Vehicle.h"

@implementation Vehicle
-(NSString * )description
{
    return [NSString stringWithFormat:@"Vehicle Properties : Speed %f, Weight %f,
        Direction %f, Is on? %i", _vSpeed_KMH, _vWeight_N, _vDirectionBearing,
        _vIsOn];
}
@end
```

As shown, I have now created a subclass of NSObject with properties and overrided a method from the superclass to replace it with something more useful.
I may also want to create some subclasses ob Vehicle like so:



Choose options for your new file:

Class: Car
Subclass of: vehicle
With XIB for user interface

The class '*car*' is now a subclass of both NSObject and vehicle. It has the properties I defined in Vehicle.h vSpeed_KMH, vWeight_N vDirectionBearing and vIsOn;