OBJECTIVE-C DATA ENCAPSULATION

All Objective-C programs are composed of the following two fundamental elements:

- **Program statements *code*:** This is the part of a program that performs actions and they are called methods.
- **Program data:** The data is the information of the program which is affected by the program functions.

Encapsulation is an Object-Oriented Programming concept that binds together the data and functions that manipulate the data and that keeps both safe from outside interference and misuse. Data encapsulation led to the important OOP concept of **data hiding**.

Data encapsulation is a mechanism of bundling the data and the functions that use them, and **data abstraction** is a mechanism of exposing only the interfaces and hiding the implementation details from the user.

Objective-C supports the properties of encapsulation and data hiding through the creation of user-defined types, called **classes**. For example:

```
@interface Adder : NSObject
{
    NSInteger total;
}

- (id)initWithInitialNumber:(NSInteger)initialNumber;

- (void)addNumber:(NSInteger)newNumber;

- (NSInteger)getTotal;

@end
```

The variable total is private and we cannot access from outside the class. This means that they can be accessed only by other members of the Adder class and not by any other part of your program. This is one way encapsulation is achieved.

Methods inside the interface file are accessible and are public in scope.

There are private methods, which are written with the help of **extensions**, which we will learn in upcoming chapters.

Data Encapsulation Example:

Any Objective-C program where you implement a class with public and private members variables is an example of data encapsulation and data abstraction. Consider the following example:

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

@interface Adder : NSObject
{
    NSInteger total;
}

- (id)initWithInitialNumber:(NSInteger)initialNumber;

- (void)addNumber:(NSInteger)newNumber;

- (NSInteger)getTotal;

@end
```

```
@implementation Adder

-(id)initWithInitialNumber:(NSInteger)initialNumber{
    total = initialNumber;
    return self;
}

- (void)addNumber:(NSInteger)newNumber{
    total = total + newNumber;
}

- (NSInteger)getTotal{
    return total;
}

@end

int main(int argc, const char * argv[])
{
    NSAutoreleasePool * pool = [[NSAutoreleasePool alloc] init];
    Adder *adder = [[Adder alloc]initWithInitialNumber:10];
    [adder addNumber:5];
    [adder addNumber:4];
    NSLog(@"The total is %ld",[adder getTotal]);
    [pool drain];
    return 0;
}
```

When the above code is compiled and executed, it produces the following result:

```
2013-09-22 21:17:30.485 DataEncapsulation[317:303] The total is 19
```

Above class adds numbers together and returns the sum. The public members **addNum** and **getTotal** are the interfaces to the outside world and a user needs to know them to use the class. The private member **total** is something that is hidden from the outside world, but is needed for the class to operate properly.

Designing Strategy:

Most of us have learned through bitter experience to make class members private by default unless we really need to expose them. That's just good **encapsulation**.

It's important to understand data encapsulation since it's one of the core features of all Object-Oriented Programming *OOP* languages including Objective-C.

Loading [MathJax]/jax/output/HTML-CSS/fonts/TeX/fontdata.js