Properties

* Property is a robust way to handle Object’s data.
* We need to define a property if you want single piece of data (or a class object member) to be visible to other classes.

**Definition of simple Property**

**@**interface SimpleProperties : UIViewController

{

int count;

}

@property (read write) int count;

@end

In the class interface we need two things

1. Define a class variable as we normally do.
2. Define a property using the directive @property.

* In the above example ‘readwrite’ is property attribute.
* Property attribute you define decides how the property behaves.

After you define the property on the class interface, you need to also define the property implementation.

@implementation SimpleProperties

@synthesize count

@end

@synthesize will create automatically setter & getter methods for the property.

**Property Declarations:**

* Atomic: Means blocking access to accessors a single access both.

If two threads try to modify the value of property, their access to property is not so simultaneous. Using this behavior helps to avoid potential problems (Ex Values not expected to pass coming two setters).

On the other hand non atomic accessor render much factor, but make no warranty as to the simultaneous access of threads to access property.

Both are similar and use multi-threading .In this case non has been selected for factor access and atomic for safer and robust access.

Non atomic is not thread safe and multi-tasking is allowed.

**Strong:** Class/Object values.

Reference count of object is always increasing i.e. memory is modified.

**Weak:** Primitive Values.

Reference count of object is same i.e. memory remains unchanged.

**Copy:** Duplicates values at initialization.

**Assign:** Primitive types.

In .h file

@property (non atomic, Strong )NSString \*str;

In .m file

@implemetation

@synthesize str; // Compile time feature generate setter/getter methods.