

I PHONE

Class notes

By
Mr . Balu sir



**SRI RAGHAVENDRA
XEROX**

Software languages Material Available

Beside Bangalore Iyyager Bakery .Opp. CDAC, Balkampet Road,
Ameerpet,Hyd

 9951596199

iPhone

26/09/2013

RS = 150/-

Smartphones & tablets are became most popular in the market because those all are using operating system.

which OS we are using in mobiles & tablets those are called mobile operating system.

* History of mobile OS :-

→ From 1979 - 1999 all mobiles are working with the help of embedded system only.

→ In the year of 1993 first smartphone OS is released with the name called IBM Siemen.

→ In the year of 1996, Palm OS introduced in the market

→ In the year of 1996, Microsoft released Windows CE OS for mobiles.

→ In the year of 1999, Nokia introduced S40 OS for their own products.

→ In the year of 2000, Symbian OS is introduced in the market with 2G technology.

→ In the year of 2001 enhancement of Palm OS is introduced.

→ In the year of 2002, Microsoft introduced enhancement of Windows CE OS.

→ In the year of 2002, BlackBerry introduced their own OS.

→ In the year of 2005, Nokia released "Maemo OS".

→ In the year of 2007, Apple corporation release their own OS called iOS, with 3G features.

- In the year 2008, google introduced Android os.
- In the year of 2009, Samsung introduced BADA OS.
- In the year of 2010, Microsoft introduced Windows OS. for windows mobile.
- In the year of 2012, Mozilla Corporation, introduced Firefox OS for mobiles.
- In the year of 2013, BlackBerry released RIM (Research In Motion) os.

* Common Operating System for Mobiles :-

- 1) Android from Google, It is a free and an open source.
- 2) BADA from Samsung electronics, it is a closed source.
- 3) Symbian from Nokia, it is an open licensed source.
- 4) Windows phone from Microsoft, it is a closed source.
- 5) BlackBerry from QNX Software, it is a closed source.
- 6) Brew from Qualcomm, it is a closed source.
- 67) Windows 8 from Microsoft, closed source.

iOS (iPhone Operating System) :-

- 1) It is a own OS of Apple Corporation.
- 2) By using this OS only, Apple devices are works like iPhone, iPad, iPod, Apple TV.
- 3) The original name of "iOS" is "OSX" & it was renamed as iOS on "june 7th 2010".
- 4) The first version of iOS is released on June 29th 2007. (Version 1.0).
- 5) On July 1st 2008 iOS 2nd version is released i.e. 2.0.
- 6) On June 17th 2009 iOS 3rd version is released i.e. 3.0.
- 7) On Feb 2nd ~~2010~~ 2010 1st generation of iPod release in the market i.e 3.1.
- 8) 1st generation iPad is released with 3.2 version.
- 9) On June 21st 2010, iOS 4.0 is released with multitask features.
- 10) The iPod 2nd generation & iPhone 3G implemented based on 4.0 only.
- 11) On June 6th 2011, iOS released 5.0 with Apple TV and many more features like imessage chat, icloud, twitter integration.
- 12) iOS 5.0 can support all iPad version, iPhone 3G, 3GS, 4, 4S models also along with iPad 3rd & 4th generation.
- 13) On Sept 19th 2012, iOS 6.0 is introduced with Siri, facebook integration, maps & lots of new features. (Siri for blind people).

- 14) On Sept. 18th 2013, iOS 7.0 introduced in Market with enhancement of upcoming versions.
- 15) To develop any kind of iOS appⁿ, we need MAC OS with 10.4 or higher configurations.
- 16) For iOS related appⁿ, we require to use IDE i.e. Xcode (latest versions supporting for iOS 7.0).

'Objective C'

Objective C is a general purpose high level object oriented programming language.

Objective C programming language designed by using C and Smalltalk-80 programming languages.

Smalltalk is a second generation object oriented programming language.

Objective C programming language follows Smalltalk message passing style (syntax) along with 'C' programming language concept.

Objective C is a main programming language used by Apple for the OS X & iOS operating system.

In 1970's Dennis Ritchie was developed C programming language but, this language is not popular until Unix OS is not released.

In 1980's Brad J. Cox designed Objective C language based on Smalltalk-80 & C language.

In the year of 1988, Objective C language developed its library & development environment called NEXTSTEP by NeXT software.

In 1992, Objective C language became open source.

In 1994, NeXT Computer & Sun Microsystem released a standardized specification of the NeXTstep system called OPENSTEP.

Free Software Foundation implementation of OPENSTEP is called GNUstep.

A Linux based GNUstep development environment is called LinuxStep.

On December 20, 1996, Apple Computer handover NeXT software, NeXTSTEP, OPENSTEP environment to release Apple's OS called OSX.

In 2007, Apple Released an updated to objective C language is called Objective-C 2.0.

Apple version of development environment is called Cocoa.

27/09/2013

When we are implementing any program by using Objective C then, extension of the file is '.m'.

Objective C related app" can be designed, compiled & run by using terminal & IDE also.

If we require to use the IDE, then must be required to go for 'Xcode' only.

Xcode IDE can be find from the system with the help of 'finder.'

When we open Xcode IDE then, in left side panel we having a option called "create new project" or go to 'file' → new → project

After choosing new-project we having a option called "choose a template for your new project".

Select 'OS X' option from left side panel & select the type as "command line tool"

After selecting "Next" button we need to provide product name i.e. app" name, company name & select type as "foundation".

* Objective 'C' programming basic standard :-

1]

```
#import <Foundation/Foundation.h>
int main(int argc, const char * argv[])
{
    NSAutoreleasePool * pool=[[NSAutoreleasePool alloc] init];
    NSLog(@"%@",@"Welcome iPhone");
    [pool drain];
    return 0;
}
```

- 1) Foundation is a framework which provides standard i/o related all predefined classes or frameworks or interface information.
- 2) argc is a variable of type Integer which holds total no. of arguments values which is passed to main fun?
- 3) argv is a variable of type char * which holds actual argument values which is passed to main function.

* NSAutoreleasePool :-

It is a predefined interface or class which is available in foundation framework.

pool is a object of type NSAutoreleasePool which holds reference of pool memory

By using alloc method we are creating pool of memory for current appl?

By using 'init' method, dynamically created memory will initialize.

'drain' is a predefined method. By using this method we are deallocated dynamically created pool memory.

example 2 :

```
#import <Foundation/Foundation.h>
int main (int argc , const char * argv [])
{
    @autoreleasepool {
        int a,b;
        NSLog(@"Enter 2 values");
        //printf("Enter 2 values");
        scanf ("%d %d", &a,&b);
        NSLog(@"Sum of %d + %d = %d",a,b,a+b);
        //printf ("\nsum of %d + %d = %d",a,b,a+b);
    }
}
```

To compile the program we require to use "Build" option.

MacBook → command + B

Windows → window + B

Product → Build.

To run or execute the program we require to use "Run" option.

MacBook → command + R

Windows → window + R

Product → Run.

while program is running if we require to stop then go for "stop" option.

Mac Book → Command + ⏎

Windows → window + ⏎

Product → Stop.

Generally Xcode containing three layout views i.e. Navigator view, debug area, utilities.

Navigator view is left side panel, debug area is bottom panel, utility is Right side panel i.e. cocoa libraries.

* NSLog :-

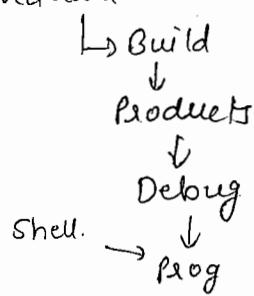
It is a predefined method which is available in foundation framework.

By using this predefined method we can print the data on console i.e. debug window.

When we are working with NSLog method, it can take any no. of argument, but first argument must NSString type. That's why we require to indicate "@" symbol.

Generally in C programming language when we are using printf it takes char *, that's why "@" specification doesn't required.

User/home/library/developer/xcode/Deriveddata 30/09/2013
/Prog1.



After constructing any application in iOS executable file or shell file can be located in User/home/library/developer/xcode/deriveddata.

* Datatypes :-

1) In Objective C we having ANSI C related datatypes and Next Step related datatype also.

2) In Objective C ANSI C related datatypes are

char - %c

short int - %hi, %hx, %ho.

unsigned short int - %hu, %hx, %ho.

int - %i, %x, %o.

unsigned int - %u, %x, %o.

long int - %li, %lx, %lo → long long int - %lli, %llx, %lo.

float - %f, %e, %g, %a.

double - %f, %e, %g, %a.

long double - %Lf, %Lf, %Lg.

BOOL

id

- 3) By using BOOL datatype we required to store boolean values i.e. TRUE or FALSE, or YES or NO.
- 4) id is a generic datatype in Objective C which can hold any kind of data values.
- 5) By using id datatype it is possible to implement dynamic casting i.e. at runtime it can hold any type of data.

Example :-

- 1) Open Xcode IDE by using finder
- 2) Create a new Xcode project
- 3) From 'choose template' option select "application" under OSX.
- 4) Select the app" type as 'Command line tool'.
- 5) Give the product name as "Prog 2".
Select the type as "Foundation".
- 6) Then save the project template in any selected directory.

Code in 'main.m'

```
#import <Foundation/Foundation.h>
int main (int argc, const char * argv[])
{
    NSAutoreleasePool * pool = [[NSAutoreleasePool alloc] init];
    BOOL isBool = YES;
    NSLog(@"%@", isBool);
    NSLog(@"%@", isBool ? @"YES" : @"NO");
    //printf("%s", isBool ? "YES": "No");
    char achar = 'a';
    unsigned char anUnsignedchar = 68;
    NSLog(@"%@", "The letter %c is ASCII number %hd", achar, achar);
```

```
NSLog(@"%@", anUnsignedChar);
```

short aShort = -89768;

```
unsigned short anUnsignedShort = 65535;
```

```
NSLog(@"%@", aShort);
```

```
NSLog(@"%@", anUnsignedShort);
```

```
int anInt = -2147483648;
```

```
unsigned int anUnsignedInt = 42944967295;
```

```
NSLog(@"%@", anInt);
```

```
NSLog(@"%@", anUnsignedInt);
```

long along = -922337203685477680;

```
unsigned long anUnsignedLong = 18443567055236856;
```

```
NSLog(@"%@", along);
```

```
NSLog(@"%@", anUnsignedLong);
```

long long along long = -92346789221012893;

```
unsigned long long anUnsignedLongLong = 18448422001098437;
```

```
NSLog(@"%@", alongLong);
```

```
NSLog(@"%@", anUnsignedLongLong);
```

```
[pool drain];
```

```
return 0;
```

```
}
```

Ex3 : Prog3 → for finding size of the datatype.

```
#import <Foundation/Foundation.h>
int main(int argc, const char *argv[])
{
    NSAutoreleasePool *pool = [[NSAutoreleasePool alloc] init];
    NSLog(@"%@", @"Size of char : %ld", sizeof(char)); → 1
    NSLog(@"%@", @"Size of unsigned char : %ld", sizeof(unsigned char)); → 2
    NSLog(@"%@", @"Size of short : %ld", sizeof(short)); → 2
    NSLog(@"%@", @"Size of U_short : %ld", sizeof(U_short)); → 2
    _____ int _____ (int); → 4
    _____ int _____ (uint); → 4
    _____ long _____ (long); → 8
    _____ ulong _____ (ulong); → 8
    _____ long long _____ (long long); → 8
    _____ ulong long _____ (ulong long); → 8
    _____ float _____ (float); → 4
    _____ double _____ (double); → 8
    _____ long double _____ (long double); → 16
    _____ bool _____ (bool); → 1
    _____ id _____ (id); → 8
    [pool drain];
    return 0;
}
```

* Datatype Range allocate predefined Macros :-

Signed char min : SCHAR_MIN
max : SCHAR_MAX

Unsigned char min : UCHAR_MIN
max : UCHAR_MAX

short int min : SHRT_MIN
max : SHRT_MAX

unsigned short int min : USHRT_MIN
max : USHORT_MAX.

signed int min : INT_MIN
max : INT_MAX.

Unsigned int min : UINT_MIN
max : UINT_MAX.

long int min : LONG_MIN
max : LONG_MAX.

Unsigned long int min : ULONG_MIN
max : ULONG_MAX.

long long int min : LLONG_MIN
max : LLONG_MAX

Unsigned long long int min : ~~ULLONG~~ ULLONG_MIN
max : ULLONG_MAX

float min : FLT_MIN
~~#~~ max : FLT_MAX

Double min : DBL_MIN
max : DBL_MAX.

MAX Array Index : SIZE_MAX.

* Control Flow Statements :-

01/10/2013

Objective C will support three types of control flow statements
i.e. selection statement. → if, else, else-if, switch.

iteration statement → while, for, do-while.

jumping statements → break, continue, goto.

example:

Create command line tool appl' with the name prog4.

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

```
int main (argc int argc, const char * argv).
```

```
{
```

```
    NSAutoreleasePool * pool = [[NSAutoreleasePool alloc] init];
```

```
    short int a,b,c,d;
```

```
    NSLog(@"Enter value of a");
```

```
    scanf ("%hi", &a);
```

```
    NSLog(@"Enter value of b");
```

```
    scanf ("%hi", &b);
```

```
    NSLog(@"Enter value of c");
```

```
    scanf ("%hi", &c);
```

```
    NSLog(@"Enter value of d");
```

```
    scanf ("%hi", &d);
```

```
    if (a==b || a==c || a==d)
```

```
{
```

```
        NSLog(@"%d is duplicate value");
```

```
}
```

```
else if (b==c || b==d)
```

```
{
```

```
        NSLog(@"%d is duplicate value");
```

```
}
```

```
else if (c==d)
```

```
{
```

```
        NSLog(@"%d is duplicate value");
```

```
}
```

```
else
{
    NSLog(@"There is no duplicate value");
}
[pool drain];
return 0;
}
```

Example 6: Prog 5:

Sum of 1st & last digit.

```
#import <Foundation/Foundation.h>
int main(int argc, const char * argv)
{
    @autoreleasepool
    {
        int value;
        short int sum=0, temp;
        NSLog(@"Enter the value");
        scanf("%d", &value);
        sum = value%10;
        value = value/10;
        while (value != 0)
        {
            temp = value%10;
            value = value/10;
        }
        NSLog(@"sum of %hi + %hi = %i", temp, sum, temp+sum);
    }
    return 0;
}
```

Ex-6: Program 6

Changing the case of letters:

```
#import <Foundation/Foundation.h>
int main(int argc, const char * argv[])
{
    @autoreleasepool
    {
        char str[100];
        NSInteger index;
        printf("Enter a string");
        gets(str);
        // scanf("%s", str);
        for (index=0; str[index] != '\0'; index++)
        {
            if (str[index] >= 'A' && str[index] <= 'Z')
                str[index] = str[index] + 32;
            else if (str[index] >= 'a' && str[index] <= 'z')
                str[index] = str[index] - 32;
        }
        // NSLog(@"%@", str);
        printf("String data: %s", str);
    }
    return 0;
}
```

O/P: Welcome to Naresh IT

⇒ WELCOME TO NARESH IT.

Ex:7 Prog - function.

* functions :-

In C programming language we having four types of user defined fun' & those all functions are supported in objective C also. i.e.

- 1) No return type with no parameter.
- 2) Return type without parameter.
- 3) Return type with parameter.
- 4) No return type with parameter.

import <Foundation/Foundation.h>.

```
Void abc()
{
    NSLog(@"%@", @"Hello");
}

Void sum(int x, int y)
{
    NSLog(@"%@", @"sum of x+y : %i", x+y);
}

int sumValue()
{
    int x = 10, y = 20;
    return (x+y);
}

int addnum(int x, int y)
{
    return (x+y);
}

int main (argc int argc, const char * argv[])
{
    NSAutoreleasePool * pool = [[NSAutoreleasePool alloc] init];
```

```

    abc();
    int a,b;
    NSLog(@"Enter 2 values:");
    scanf("%d %d", &a, &b);
    sum(a,b);
    int s;
    S = sumvalue();
    NSLog(@"sumvalue1 is :%d",S);
    S = addnum(120,200);
    NSLog(@"sum value2 is :%d",S);
    [pool drain];
    return 0;
}

```

* Object Oriented Concept in Objective C

- 1) Whenever a programming language supports oops concept then it is called Object Oriented Programming language.
- 2) In any OOP language we having four OOPS concepts.
 i.e. a) Abstraction.
 b) Encapsulation.
 c) Polymorphism.
 d) Inheritance.

a) Abstraction:-

It is a concept of providing the information about an entity without any implementation.

~~Encap.~~

b) Encapsulation:

It is a concept of providing the security for the data with the help of private or protected variables.

c) Polymorphism:

It is a concept of implementation of method multiple times with same name.

d) Inheritance:

Deriving the properties of base class to derived class is called inheritance.

In Objective C, we having two types of inheritance only i.e. 1) Single inheritance
2) Multilevel inheritance.

- 1) When we are deriving a class from single base class it is called single inheritance.
- 2) Deriving a class from an existing derived class is called multilevel inheritance.

Whenever we are working with oops concepts we need classes & objects.

* Class: It is an user defined datatype which is constructed according to realtime environment.
i.e. realtime object model information.

A class is an entity of data members & member functions (variables & methods).

A class is a blueprint of an entity. without any physical memory.

Object: An object is a physical instance of a class.

when we are creating an object for class then only memory will constructed.

Any kind of objective C application c having three parts

i) interface section.

ii) Implementation section.

iii) Program section.

02/01/2013

1) Interface Section:-

This section contains class declaration & method declaration only., it doesn't having any implementation part.

Objective C related classes contains two types of members & two types of methods i.e. non static or instance variables , static ~~or~~ variables.

For every object of the class, if the physical memory is constructed for a member. then it is called non static or instance members.

For complete class related object only once, if memory is created then it is called static member.

For every object if corresponding method is available then it is called instance method.

For complete class related object ,if only one method is available then it is called static method.

* Syntax to create a class :

```
@ interface class_name : Super class
{
    Data members ;
}
member functions ;
@ end.
```

inherited from superclass

* Syntax to create instance method :

- (return type) method name ;

* Syntax to create static method :

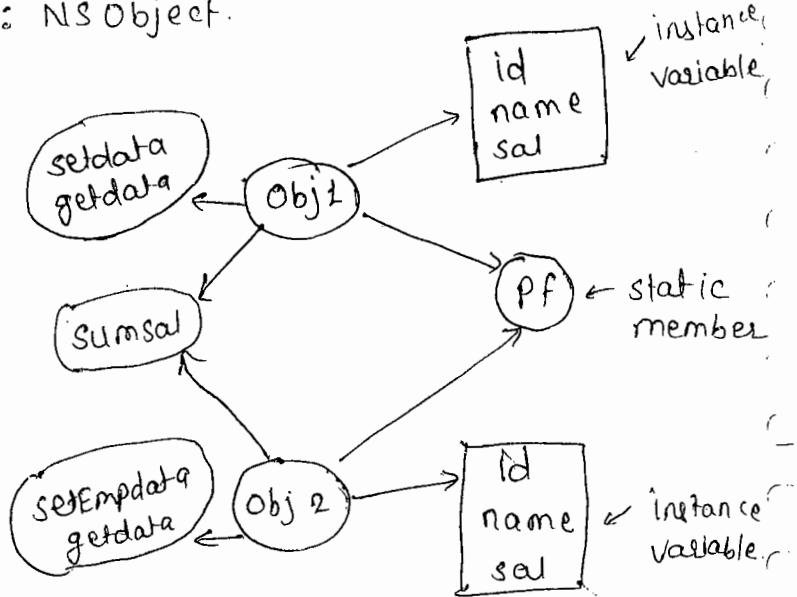
- + (return type) method name ;

Note: '-' sign will indicates instance methods ,

'+' sign will indicates static methods .

Ex: @ interface Empclass : NSObject.

```
{
    int id;
    char name[36];
    int sal;
    static float pf;
}
-(void)setEmpdata;
-(void) getData;
+(int) sumSal;
@ end.
```



2) Implementation Section :

This part contains implementation part of the class.

Method implementation always required to construct within the implementation part.

* Syntax :

@ implementation class_name.

 methods

@ end.

Ex: @ implementation EMPCLASS.

- (Void) SetEmpData
{
 // code
}

- (void) getData
{
 // code
}

+ (int) SumSal.
{
 // code
 return statement;
}

@ end.

3) Program Section :

This section contains object creation & method calling.

~~Give~~

Example:

- * Create a command line tool app with the name called class Ex1.

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

```
@interface EMPCLASS : NSObject
```

```
{ int EMPid;
```

```
char EMPname[36];
```

```
int EmpSal;
```

```
}
```

```
- (void) SetEMPData;
```

```
- (void) GetData;
```

```
- (int) SumSal; (int) g2;
```

```
- (int) GetSalary;
```

```
@end
```

```
@implementation EMPCLASS
```

```
- (void) SetEmpData
```

```
{
```

```
Emp id = 101;
```

```
strcpy(EMPname, "Rajesh");
```

```
EMPSal = 15000;
```

```
}
```

```
- (void) GetEmpData
```

```
{
```

```
// char * str = (char *) malloc(36, sizeof(char));
```

```
NSLog(@"Enter Emp id: ");
```

```
scanf("%d", &Empid);
```

```
NSLog(@"Enter Emp Name: ");
```

```
fflush(stdin);
```

```
scanf("%s", EMPname);
```

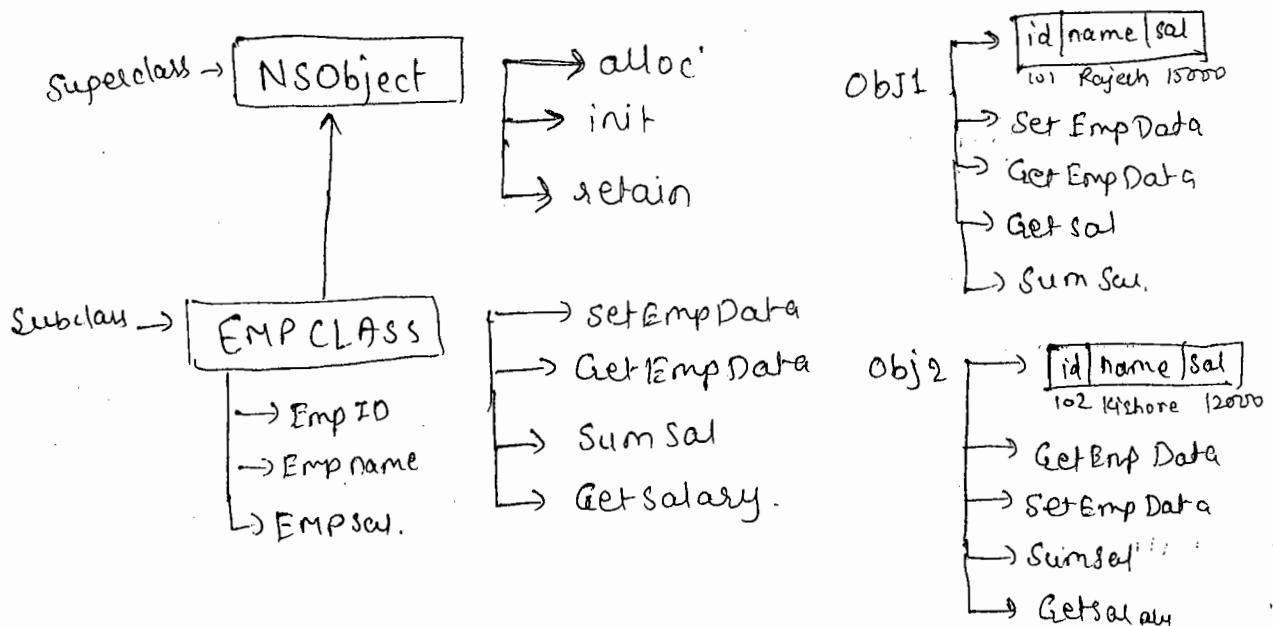
```
NSLog(@"Enter EMP salary: ");
```

```
scanf("%d", &EMPSal);
```

```
}
```

```
- (int) sumSal: (int) s2;  
{  
    return (EMPsal + s2);  
}  
- (int) GetSalary  
{  
    return Empsal;  
}  
@ end
```

```
int main (int argc, const char * argv [ ] )  
{  
    @autoreleasepool {  
        static method  
        EMPCLASS * Obj1 = [[EMPCCLASS alloc] init];  
        ↓  
        instance method.  
        [Obj1 SetEmpData];  
  
        EMPCLASS * Obj2 = [[EMPCCLASS alloc] init];  
        [Obj2 GetEMPData];  
  
        int totalSal;  
        totalsal= [Obj1 sumSal: [Obj2 GetSalary]];  
        NSLog(@"Total Salary : %i", totalSal);  
        [Obj1 retain];  
        [Obj2 retain];  
    }  
    return 0;  
}
```



$$\text{Total} = [\text{Obj1 sumsal} : [\text{Obj2 getsal}]] \\ 27000 = \downarrow \quad + \quad \downarrow \\ 15000 \quad + \quad 12000$$

—XOX—

03/10/2013

Note:

When we are working class method or static method then always it should be called by using class name only.

Class methods can access static variables of class. i.e. Static members only we can access.

Instance method required to call with object name only.

Instance method can access instance variable of class.

NSObject is a root or superclass of any iOS app, which can provide foundation framework related all methods.

When we are creating any subclass from NSObject class then automatically NSObject related methods will included.

~~@attr~~

* @autoreleasepool :

This block will create NSAutoreleasePool related object automatically. & when control is passing outside of the block then it call 'drain' method.

* Working with multiple file interface :-

When we are working with any iOS related app, then code will be implemented in multiple files.

Always interface section should be required to place in '.h' file & implementation section must be required to place in '.m' file.

example:

Create an app of type commandline tool with the name class Ex 2.

⇒ Add a Objective C file to the current app.



* Code in FPSClass.h

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

```
@interface FPSClass : NSObject
```

```
{ int ft;  
    float ins;  
}
```

```
- (void) Set ft : (int) i *;  
- (void) Set ins : (float) f;  
- (int) Get ft;  
- (float) Get ins;  
- (void) Set Fps Data : (int) i with ins : (float) f ;  
- (void) Get Fps Data;  
- (FpsClass*) Add : (FpsClass*) obj A with object : (FpsClass*) obj B;  
@ end.
```

* Code in Fps Class.m :-

```
# import "FpsClass.h" // user defined header file.  
@ implementation FpsClass  
- (void) Set ft (int) i  
{  
    ft = i;  
}  
- (void) Set ins (float) f  
{  
    ins = f;  
    if (ins >= 12.0f)  
    {  
        ft = ft + (int) ins / 12; // ft += (int) ins / 12;  
        ins = ins - (float) ins / 12 * 12; // ins -= (int) ins / 12 * 12;  
    }  
}  
- (int) Get ft  
{  
    return ft;  
}  
- (float) Get ins  
{  
    return ins;  
}
```

```

-(void) SetFpsData : (int) i withIns : (float) f
{
    ft = i;
    ins = f;
    if (ins >= 12.0f)
    {
        ft += (int)ins / 12;
        ins -= (int)ins / 12 * 12;
    }
}

-(void) GetFpsData
{
    NSLog(@"Obj 2 result: %d %.2f", ft, ins);
}

-(FpsClass*) Add : (FpsClass*) objA withObject : (FpsClass*) objB
{
    ft = Obj-A → ft + ObjB → ft;
    ins = ObjA → ins + ObjB → ins;
    if (ins >= 12.0f)
    {
        ft += (int)ins / 12;
        ins -= (int)ins / 12 * 12;
    }
    return self;
}

@end.

```

* Code in main.m :-

```

#import <Foundation/Foundation.h>
#import "FpsClass.h"
int main ( @int argc, const char* argv )
{
    @autoreleasepool
    {
        FpsClass* obj1 = [[FpsClass alloc] init];
    }
}

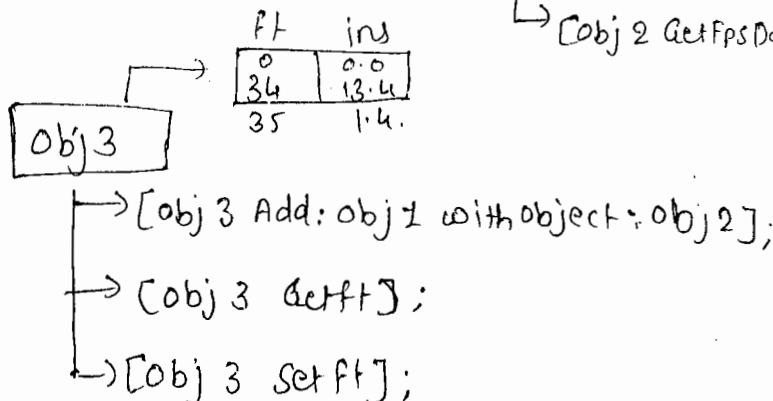
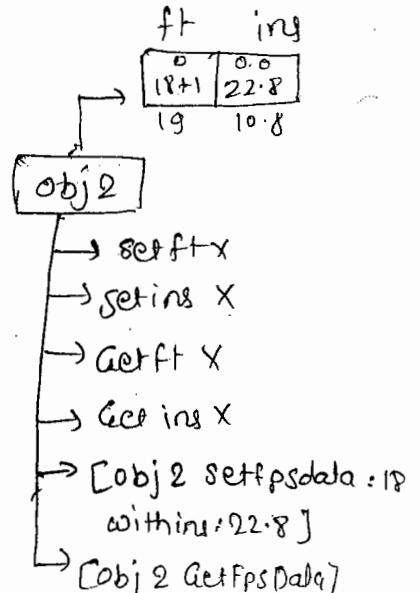
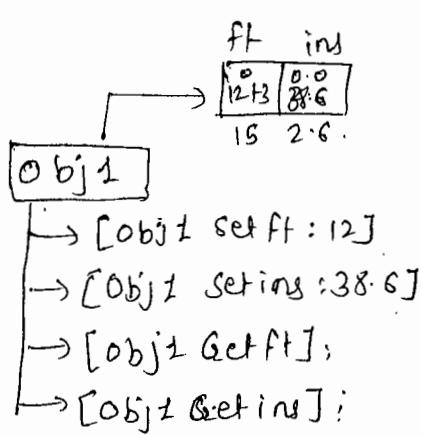
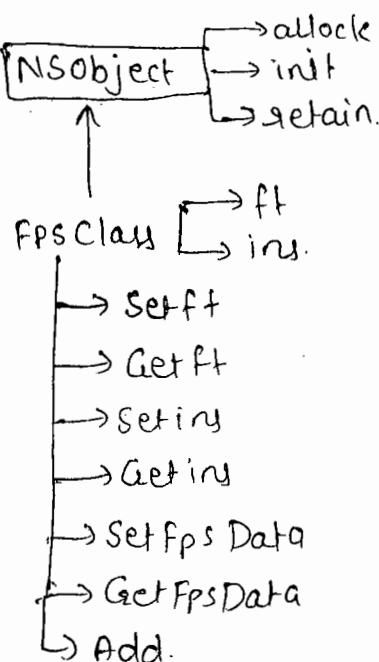
```

```

[Obj1. set ft : 12];
[Obj1 Set ins : 38.6];
NSLog(@"Obj1 result: %d %.2f", [Obj1 Getft], [Obj1 Getins]);
FPSClass * Obj2 = [[FPSClass alloc] init];
[Obj2 SetFpsData : 18.0 with ins : 22.8];
[Obj2 GetFpsData];

FPSClass * Obj3 = [[FPSClass alloc] init];
Obj3 = [Obj3 Add: Obj1 withObject: Obj2];
NSLog(@"Obj3 result: %d %.2f", [Obj3 Getft], [Obj3 Getins]);
[Obj1 retain];
[Obj2 retain];
[Obj3 retain];
}
return 0;
}

```



04/10/2013

By using Setter methods we required to assign instance variable data.

By using Getter methods we can access instance variable data.

'Self' is a keyword which always refers current object.

'super' is a keyword which always refers base class object.

~~XXXX~~

* Create a command line appl" with the name ClassEx3.

=> Create a new class with the name 'MathClass'.

* Code in MathClass.h

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

```
@ interface MathClass : NSObject.
```

```
{  
    double result;  
}
```

```
- (void) SetResult: (double) r;
```

```
- (void) GetResult;  
    double
```

```
- (void) Add : (double) v2 ;
```

```
- (void) Sub : (double) v2 ;
```

```
- (void) Mul : (double) v2 ;
```

```
- (void) div : (double) v2 ;
```

```
- (void) Mod : (double) v2 ;
```

```
@ end.
```

* Code in MathClass.m :

```
#import "MathClass.h"
@implementation MathClass
-(void) SetResult :(double)a
{
    result = a;
}
-(double) GetResult
{
    return result;
}
-(void) Add :(double)V2
{
    result += V2; //result = result + V2;
}
-(void) Sub :(double)V2
{
    result -= V2;
}
-(void) Mul :(double)V2
{
    result *= V2;
}
-(void) Div :(double)V2
{
    result /= V2;
}
-(void) Mod :(double)V2
{
    result = fmod(result, V2);
}
@end.
```

* Code in main.m:

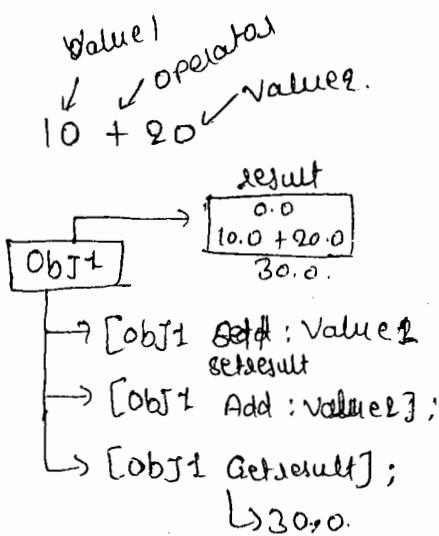
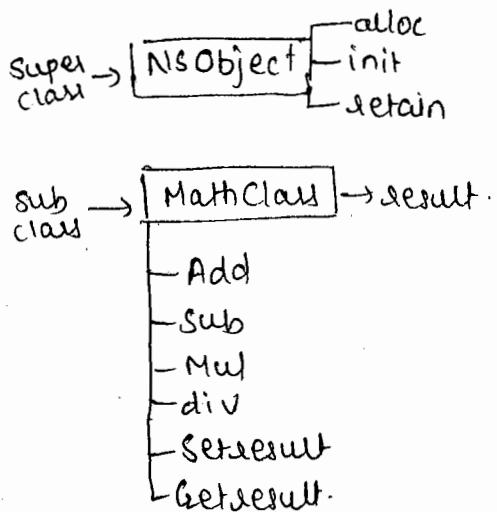
```
# import <Foundation/Foundation.h>
# import "MathClass.h"

int main (int argc, const char * argv[])
{
    NSAutoreleasePool * pool = [[NSAutoreleasePool alloc] init];
    double value1, value2;
    char operator;
    BOOL operatorIsValid = TRUE;
    NSLog(@"Enter the value");
    scanf ("%lf %c %lf", &value1, &operatorIsValid, &value2);
    MathClass * obj = [[MathClass alloc] init];
    [obj setResult : value1];
    switch (operator)
    {
        case '+': [obj add : value2];
                     break;
        case '-': [obj sub : value2];
                     break;
        case '*': [obj mul : value2];
                     break;
        case '/': [obj div : value2];
                     break;
        case '%': [obj mod : value2];
                     break;
        default:
            operatorIsValid = FALSE;
            break;
    }
    if (operatorIsValid == TRUE)
    {
        NSLog(@"%.2f %c %.2f", value1, operator, value2,
              [obj getResult]);
    }
}
```

```

else {
    NSLog(@"Invalid input data");
    [obj1 retain];
    [pool drain];
    return 0;
}

```



- * #import <filename.h> is required to use when we are loading predefined frameworks or header files.
- * #import "filename.h" is required when we are importing user defined header files from current project directory.

"@ property" directive:

05/10/2013

By using this directive we can provide accessor method (setter or getter) for any class related members in interface section.

Generally Getter & Setter methods are called as accessor method which required to make enable by using "@synthesize" directive.

@property directive we require to place interface section, @synthesize directive we require to place implementation section.

'@property' directive having multiple attributes like → ① nonatomic :-

- 'nonatomic' means multiple threads can access that variable.
- 'nonatomic' is thread unsafe & fast in performance.
- 'nonatomic' is not a default attribute so, we need to use 'nonatomic' keyword.
- When we are working with non-atomic member then result can be unexpected because multiple threads accessing same member.

for. ex:- @property (nonatomic, retain) NSString *Name;
 @synthesize name;

② atomic :-

- atomic means only one thread can access that member & until current thread is not completed another thread cannot be accessed.

→ By default any member's default attribute is 'atomic' only.

→ atomic attribute is safe but it is slow in process.

e.g. @property (retain) NSString *Name; (atomic is by default)
 @synthesize name;

③ 'retain':

→ when we are applying 'retain' attribute then old value will retain & new value will assign.

→ 'retain' attribute will send the information that, old value need to be released before assigning new value.

④ 'read only':

→ This attribute we require to use when we need to apply 'read only' property to a member.

→ When we are applying 'read only' property then compiler doesn't generate getter & setter method automatically.

→ If we are apply read only attribute then, getter method only required to place in implementation section.

→ If we are making synthesize for read only attribute then it provides only getter method, if we are trying to access the data to assign by using '=' operator then it gives an error.

⑤ 'read write':

→ This attribute will generate setter & getter methods to modify the member of a class.

- This attribute is a default property of a class.
- When we are applying readwrite attribute then setter & getter both methods available in implementation block.

⑥ strong(iOS 4 = retain) :-

- This attribute will says to the compiler that, hold the member in heap area until it is not retain.
- By default all instance variables & local variables are 'strong' only.
- Generally, in cocoa framework it is required to use 'strong' attribute.

⑦ weak(iOS 4 = unsafe_unretained) :-

- It says to the compiler, keep this member as long it is possible.
- On weak members we can't call retain or release object methods.
- When the weak member is deallocated then automatically pointer will set to null.

⑧ assign:

- This attribute will work like assignment operator.
- assign property having setter methods only.
- By using this attribute we can generate 'c' primitive type datatypes.

⑨ copy:

- This attribute is required for mutable Objects.
- Copy attribute will specify that old value required to copy with new data.

* Create a commandline tool appl" with name class Ex 4.
⇒ Create a new class with the name EmpClass

* Code in EmpClass.h :-

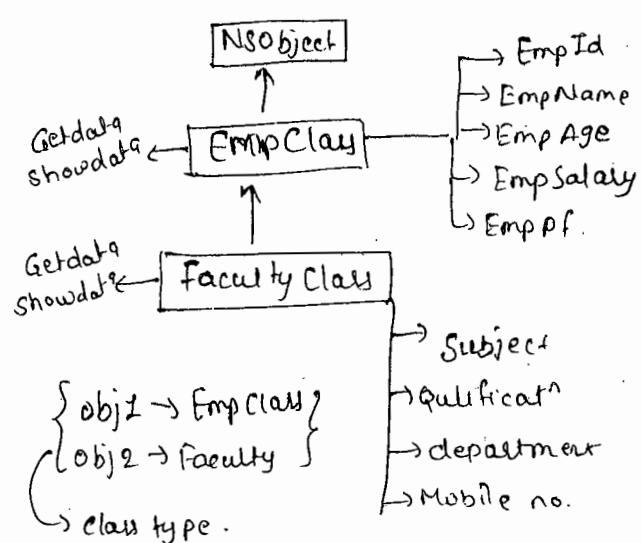
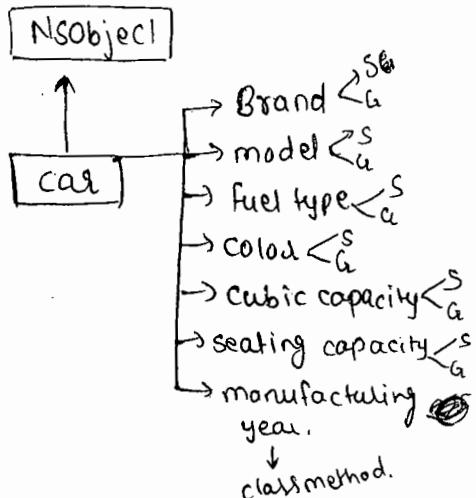
```
#import <Foundation/Foundation.h>
@interface EmpClass : NSObject
{
    int EmpId;
    NSString *name;
    short int Age;
    unsigned int Salary;
}
@property (nonatomic, readwrite) int EmpId;
@property (nonatomic, retain) NSString *name;
@property (readwrite) short int age;
@property unsigned int Salary;
- (int) SumSal : (EmpClass *) obj B;
@end.
```

* Code in EmpClass.m :-

```
#import "Emp.h"
@implementation EmpClass
@synthesize EmpId, Age;
@synthesize name;
@synthesize Salary;
- (int) SumSal : (EmpClass *) obj B.
{
    return (Salary + obj B->Salary);
}
@end.
```

* Code in main.m :-

```
#import <Foundation/Foundation.h>
#import <"Emp Class.h">
int main (int argc, const char * argv[])
{
    @autoreleasepool {
        EmpClass * obj1 = [[EmpClass alloc] init];
        [obj1 setEmpId:102];
        [obj1 setName:@"Rajesh"];
        [obj1 setAge:26];
        [obj1 setSalary:12300];
        NSLog(@"EMP ID: %d", [obj1 EmpID]);
        NSLog(@"Emp Name : %@", [obj1 EmpName]);
        NSLog(@"EMP Age : %i", [obj1 Age]);
        NSLog(@"Emp Salary : %.4f", [obj1 Salary]);
        int tsal;
        tsal = [obj1 sumSal : obj2];
        NSLog(@"Total Salary : %.d", tsal);
        [obj2 release];
        [obj1 release];
    }
    return 0;
}
```



07/10/2013

- * Create a command line tool appl' with the name ClxyFx 5.
Create a new class with the name called Car subclass of
P type NSObject.

1

- * Code in Car.h

```
#import <Foundation/Foundation.h>
@interface Car : NSObject
{
    NSString * brand;
    —" — model;
    —" — fuel;
    —" — color;
    short int cubic;
    short int scapacity;
}
+ (int) getmyear;
- (void) SetBrand : (NSString *) bname;
- (NSString *) GetBrand;
- (void) SetModel : (NSString) mname;
- (NSString *) GetModel;
- (void) SetFuel : (NSString) fnames;
- (NSString *) GetFuel;
- (void) SetColor : (NSString) cnames;
- (NSString *) GetColor;
- (void) SetCubic : (int) C;
- (int) GetCubic;
- (void) SetScapacity: (int) S;
- (int) Getscapacity;
- (void) displaydata;
@end.
```

* Code in Car.m :

```
#import "Car.h"
@implementation car;
static int myear;
+(int) getmyear
{
    myear=2013;
    return myear;
}
-(void) SetBrand:(NSString *)bname
{
    brand=bname;
}
-(NSString *) GetBrand
{
    return brand;
}
-(void) SetModel:(NSString *)mname
{
    model=mname;
}
-(NSString *) GetModel
{
    return b.model;
}
-(void) SetColor:(NSString *)sname
{
    color=cname;
}
-(NSString *) GetColor
{
    return color;
}
```

```
- (Void) setfuel (NSString*) fname
{
    fuel = fname;
}

-(NSString*) Getfuel
{
    return fuel;
}

-(void) SetCubic :(int)c
{
    cubic = c;
}

-(int) GetCubic
{
    return cubic;
}

-(Void) SetScapacity :(int)s
{
    scapacity = s;
}

-(int) GetScapacity
{
    return scapacity;
}

-(void) displaydata
{
    NSLog(@"%@", brand, model, fuel, color, cubic,
            scapacity, myear);
}

@ end.
```

* Code in main.m.

```
#import <Foundation/Foundation.h>
#import "Car.h"
int main (int argc, const char * argv[])
{
    @autoreleasepool
    {
        Car* Obj1 = [[Car alloc] init];
        [Obj1 SetBrand : @"Hyundai"];
        [Obj1 SetModel : @"i20"];
        [Obj1 SetFuel : @"Petrol"];
        [Obj1 SetColor : @"Red"];
        [Obj1 SetCubic : 1086];
        [Obj1 SetScapacity : 5];
        NSLog(@"%@", Obj1);
        [Obj1 retain];
        Car* Obj2 = [[Car alloc] init];
        [Obj2 SetBrand : @"Hyundai"];
        [Obj2 SetModel : @"i20"];
        [Obj2 SetFuel : @"Diesel"];
        [Obj2 SetColor : @"Red"];
        [Obj2 SetCubic : 1396];
        [Obj2 SetScapacity : 5];
        Obj2.displaydata();
        Obj2.retain;
        return 0;
    }
}
```

* Multilevel Inheritance *

- Deriving a class from an existing derived class is called Multilevel Inheritance.
- When we are working with multilevel inheritance then derived class contains all the functionality of superclss & baseclass also.

* Overloading :-

Writing a method multiple times with same name & different objects is called overloading.

* Overriding :-

Reimplementation of base class method into subclass is called overriding.

- Generally when we need to provide extending functionality of superclss method into subclass then recommended to go for overriding.
- By using overriding we can provide complete new behaviour of superclss method into subclass & it is possible to extend new functionalities.
- When we are extending the superclss method into subclass then must be required to call Superclss method from subclass.

- * Create a commandline tool appl" with name classEx6.
- Create a new class called Emp subclass of type NSObject.
- Create a new class called Faculty subclass of type Emp.

* code in EMP.h :-

```
#import <Foundation/Foundation.h>
@interface EMP : NSObject
{
    int empId;
    NSString* empName;
    short int empAge;
    unsigned int empSal;
    float empPF;
}
-(void)getData;
-(void)showData;
@end.
```

code in Emp.m :-

```
#import "EMP.h"
@implementation EMP
-(void)getData
{
    char name[30];
    NSLog(@"Enter EMP ID:");
    scanf("%d", &empId);
    NSLog(@"Enter Name:");
    scanf("%s", name);
    empName = [NSString stringWithUTF8String:name];
    NSLog(@"Enter Age:");
    scanf("%hi", &empAge);
    NSLog(@"Enter Salary:");
    scanf("%u", &empSal);
    empPF = (float)empSal / 10;
}
```

```
- (void) ShowData  
{  
    NSLog(@"%@", "ID: %d NAME: %@ SAL: %.2f Age: %.2f Pf: %.2f",  
        empId, empName, empSal, empAge, empPf);  
}
```

@ end.

* Code in FACULTY.h :-

```
#import "EMP.h"  
@ interface FACULTY : EMP
```

```
{  
    NSString * facSub;  
    NSString * facQuali;  
    NSString * facDept;  
    NSString * facMobile;  
}
```

```
- (void) getData;  
- (void) ShowData;
```

@ end.

* Code in FACULTY.m :-

```
#import "FACULTY.h"  
@ implementation FACULTY
```

```
- (void) getData
```

```
{  
    char sub[20];  
    char @quali[10];  
    char dept[10];  
    char mobile[10];  
    [super getData];
```

```
    NSLog(@" Enter Sub: ");  
    scanf("%s", sub);
```

```
    facSub = [NSString stringWithUTF8String: sub];
```

```

    NSLog(@"Enter Quli:");
    scanf("%s", quli);
    facQuli = [NSMutableString stringWithUTF8String:quli];
    NSLog(@"Enter Dept:");
    scanf("%s", dept);
    facDept = [NSMutableString stringWithUTF8String:dept];
    NSLog(@"Enter Mobile:");
    scanf("%s", mobile);
    facMobile = [NSMutableString stringWithUTF8String:mobile];
}

-(void) ShowData {
    [super ShowData];
    printf("Sub: %s Quli: %s Dept: %s Ph: %s, [%facSub UTF8String]\n",
           , [%facQuli UTF8String]
           , [%facDept UTF8String]
           , [%facMobile UTF8String]);
}

@end.

```

* Code in main.m :-

```

# Import <Foundation/Foundation.h>
# Import "Emp.h"
# import "FACULTY.h"

int main(int argc, const char * argv[])
{
    @autoreleasepool
    {
        EMP *obj1 = [[EMP alloc] init];
        [obj1 getData];
        [obj1 showData];

        FACULTY *obj2 = [[FACULTY alloc] init];
        [obj2 getData];
        [obj2 showData];
    }
}

```

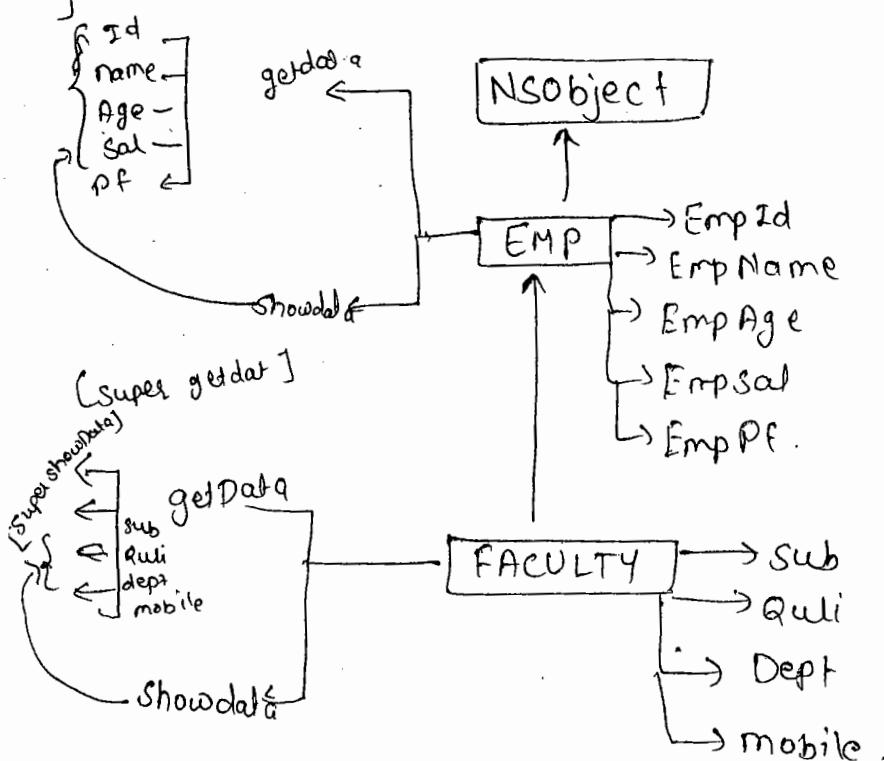
```

if([obj1 isKindOfClass:[EMP <#class>]]==TRUE)
{
    NSLog(@"Obj1 type is EMP");
}

if ([FACULTY isKindOfClass:[NSObject <#class>]]==TRUE)
{
    NSLog(@"FACULTY is subclass-of NSObject");
}

[Obj1 retain];
[Obj2 retain];
}
return 0;
}

```



08/10/2013

- * Procedure to override NSObject methods in derived classes.
- * Create a commandline tool appl' with the name BankBox.
- ⇒ Create a new class with the name BankAccount subclass of type NSObject.
- ⇒ Create a new class with name Saving Account subclass of type BankAccount.

* Code in BankAccount.h :-

```
#import <Foundation/Foundation.h>
@interface BankAccount : NSObject
{
    long int accountNo;
    double accountBalance;
}
@property (readwrite) long int accountNo;
@property (readwrite) double accountBalance;
+ (BankAccount*)alloc;
+ (int)noOfObjectCount;
-(void)displayInfo;
@end.
```

* Code in BankAccount.m :

```
#import "BankAccount.h"
@implementation BankAccount
@synthesize accountNo;
@synthesize accountBalance;
static int objectCount=0;
+ (BankAccount*)alloc
{
    ++objectCount;
    return([super alloc]); // NSObject alloc
```

```
+ (int) noOfObjectCount  
{  
    return objectCount;  
}  
- (void) displayInfo  
{  
    NSLog(@"Account No : %ld Balance : %.1f", accountNo,  
          accountBalance);  
}  
@ end.
```

* Code in Saving Account.h

```
#import <Foundation/Foundation.h>  
#import "BankAccount.h"  
@interface SavingAccount : BankAccount  
{  
    float intRate;  
}  
@property (readwrite) float intRate;  
+ (SavingAccount*) alloc;  
- (void) displayInfo;  
- (double) calcIntRate;  
@ end.
```

* Code in Saving Account.m :-

```
#import "SavingAccount.h"  
@implementation SavingAccount  
@synthesize intRate;  
+ (SavingAccount**) alloc  
{  
    return ([super alloc]);  
}
```

```

-(void) displayInfo
{
    NSLog(@"%@", "Account No: %ld Balance: %lf intRate: %f",
            accountNo, accountBalance, intRate);
}

-(double) calcIntRate
{
    return (accountBalance * intRate);
}

@ end.

```

* Code in main.m :-

```

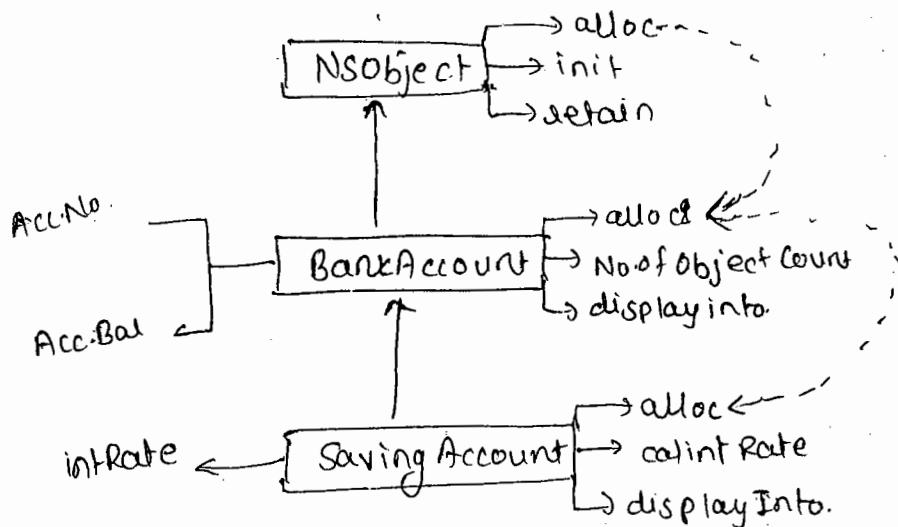
#import <Foundation/Foundation.h>
#import "BankAccount.h"
#import "Saving Account.h"
int main( int argc, const char * argv[] )
{
    @autoreleasepool
    {
        BankAccount * obj1 = [[BankAccount alloc] init];
        [obj1 setAccountNo: 123456];
        [obj1 setAccountBalance: 3428.22];
        [obj1 displayInfo];

        Saving Account * obj2 = [[Saving Account alloc] init];
        [obj2 setAccountNo: 325674];
        [obj2 setAccountBalance: 45000]; [obj2 setIntRate: 0.02f];
        [obj2 displayInfo];

        NSLog(@"%@", "Int Rate Amount: %lf", [obj2 calcIntRate]);
        int ObjCount = [BankAccount noofObjectCount];
        NSLog(@"%@", "Object Count : %d", ObjCount);

        [obj2 retain];
        [obj1 retain];
    }
    return 0;
}

```



Bank Account * Obj = [[BankAccount alloc] init];

Acc No	Acc Bal.
123456	0.0

[BankAcc alloc].

↓

[NSObject alloc]

[Obj1 setAccNo : 123456]

[Obj1 setAccBal : 1234.54];

Saving Acc * Obj2 = [[Saving alloc] init];

[Bank Acc alloc]
 ↓
 [Obj2 setAccNo : 789012]; intRate ← [Saving Acc. alloc]
 ↓
 [Obj2 setAccBal : 45000]; Acc No ← [Bank Acc. alloc]
 ↓
 Saving Acc [Obj2 setIntRate : 0.02f]; Acc Bal ← [Bank Acc. alloc]
 ↓
 [Obj2 displayInfo]; memory ← [NSObject alloc]

* Controlling Access to instance Variables:-

- 1) By default instance variables in a class can be accessed by methods of the class.
- 2) This is the default behaviour of any class.
- 3) By default any class can contain three types of access controlling. i.e.
 - a) Protected :- Access is allowed only by the methods of the class & its subclass.

b) Private: Access is ^{not} restricted to methods of that class.
Access is not available to subclass.

c) Public: Direct access available to methods of the class,
subclass & code in another module or classes.

09/10/2013

Categories in Objective C

- By using categories it is possible to add new methods to existing class in place without using subclasses. (Inheritance)
- To implement the categories we doesn't require existing class related .h & .m file.
- Categories can be refer like class extension also.
- A category can be declare for any classes even if we don't having original implementation source code such as for standard cocoa or cocoa touch classes.
- Objective C categories provide the ability to add functionality to an object without subclassing or changing the actual object.
- Generally categories required to use to add the methods to existing classes like NSSetting or your own custom objects also.
- Subclassing is one way to add the functionality to an object but avoiding unnecessary code subclassing by using a category will help to reduce the amount of code & keep your project more organised.

- In following case always recommended to go for categories.
 - 1) Extending an existing cocoa class.
 - 2) Class Extension
 - 3) Private methods on a class.

* Syntax to Create Categories :-

@interface Class-Name (Category)

 Data Members;

 Member Functions;

@end.

→ Always category related interface section should be exist within the header files only.

→ Implementation section must be exist within .m file only.

@ implementation Class-Name (Category)

 method code

@end.

Example :

Adding getData & showData methods to NSObject.

→ Create a commandline tool appl" with name CatExt.

→ Add the new category file to the project.

→ Select file → new → file of type "Objective-C category"

→ Give the Category name as NSObjectExtention. & select category on NSObject.

* Code in NSObject + nsobjectExten.h.

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

```
@interface NSObject(nsobjectExten)
```

```
- (void) getData;
```

```
- (void) showData;
```

```
@end.
```

* Code in NSObject + nsobjectExten.m :-

```
#import "NSObject + nsobjectExten.h"
```

```
@implementation NSObject(nsobjectExten)
```

```
- (void) getData
```

```
{
```

```
    NSLog(@"from getData");
```

```
}
```

```
- (void) showData
```

```
{
```

```
    NSLog(@"Hello@ Show Data");
```

```
}
```

```
@end.
```

* Code in main.m :-

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

```
#import <NSObject + nsobjectExten.h>
```

```
int main(int argc, const char * argv[])
```

```
{
```

```
    @autoreleasepool
```

```
{
```

```
        NSObject *obj = [[NSObject alloc] init];
```

```
        [obj getData];
```

```
        [obj showData];
```

```
        [obj retain];
```

```
    }
```

```
    return 0;
```

* Creating a category & custom class.

* Create a commandline tool appl' with the name EmpCat.
→ Create a new class with the name Emp.

* Code in Emp.h :-

```
# import <foundation/Foundation.h>
@interface EMP : NSObject {
    short int EmpId;
    NSString * EmpName;
    unsigned int empSal;
}
@property (readwrite) short int EmpId;
@property (copy) NSString * EmpName;
@property unsigned int empSal;
@end.
```

* Code in EMP.m :-

```
#import "EMP.h"
@implementation EMP
@synthesize EmpId = _EmpId;
@synthesize EmpName = _EmpName;
@synthesize empSal;
@end.
```

Create new "Objective-c category" file with name
~~EMP~~' EmpExten category on EMP.

* Code in EMP + EmpExten.h :-

```
# import "EMP.h"  
@ interface EMP (EmpExten)  
    -(void) showData;  
    -(void) getData :(int)i withName: (NSString *)  
        name withSal : (unsigned int)s;  
@ end.
```

* Code in Emp + EmpExten.m :-

```
@ implementation EMP (EmpExten)  
-(void) getData :(int)i withName: (NSString *) name withSal  
    (unsigned int)s  
{  
    EmpId = i;  
    EmpName = name;  
    EmpSal = s;  
}  
-(void) showData  
{  
    NSLog(@"%@", @"hi %@", @"u", EmpId, EmpName, EmpSal);  
}  
@ end.
```

* Code in main.m :-

```
# import <Foundation/Foundation.h>  
# import <EMP + EmpExten.h>  
int main (int argc, const char * argv [ ] )  
{  
    @autoreleasepool  
    {  
        EMP *obj1 = [[EMP alloc] init];  
        [obj1 getData: 102 withName: @"Ray" withSal: 12345];  
        [obj1 showData];  
        [obj1 retain];  
    }  
    return 0;  
}
```

10/10/2013

- When we are working with the categories it will allow to add the methods to existing class only i.e. it is not possible to add any instance variable to the categories.
- In implementation if we required instance variables to any existing classes then, it must be required to go for class extension.

* Class Extension :-

- Class extension is a similar concept to categories with more powerful features.
- When we are working with class extension it will allow to add new instance variables & methods to existing class.
- When we are trying to add class extension to the current project then it will create '.h' file only.
(if it is category .h & .m will also be created).
- Class extension must be required to add for custom classes only. i.e. it doesn't allow for 'cocoa' classes.
(categories will allows for both type).
- When we are working with class extension, must be required the implementation part of existing class.
(category doesn't required).
- When we are working with class Extension, methods must be implemented in original class related .m file only.
(category will allows within their own .m file)

* Syntax to Create Class Extension:-

```
@ interface class_Name ()  
{  
    Instance variables  
    methods  
}  
@ end.
```

example:-

- * Create a commandline tool appl' with the name ClassExt.
- ⇒ Add a new class with the name called EMP, subclass of type NSObject.
- ⇒ Create a new class Extension file with name NEWEMP on EMP class.

* Code in EMP.h :-

```
# import <Foundation/Foundation.h>  
@ interface EMP : NSObject  
{  
    short int eId;  
    NSString *eName;  
    unsigned int esal ;  
}  
@ property (readwrite) short int eId;  
@ property (copy) NSString *eName;  
@ property (readwrite) unsigned int esal;  
-(void) display Data ;  
@ end.
```

* Code in EMP.m :-

```
#import "EMP.h"
#import "EMP_NEEMP.h"

@interface EMP
@synthesiz eId, eSal, eName, epno;
-(void) SetData : (int)i withName: (NSString*)name andSal :
    (unsigned int)s withPhno : (NSString*)p
{
    eId = i;
    eName = Name;
    eSal = s;
    epno = P;
}

-(void) display Data
{
    NSLog(@"ID: %d, Name: %@ Sal: %u Ph: %@", eId, eName,
          eSal, epno);
}

@end
```

* Code in EMP_NEEMP.h :-

```
#import "EMP.h"

@interface EMP
{
    NSString *epno;
}

@property (copy) NSString *epno;

-(void) SetData : (int)i withName: (NSString*)name andSal :
    (unsigned int)s withPhno : (NSString*)p;
```

* Code in main.m :-

```
#import <Foundation/Foundation.h>
#import "EMP.h"
#import "EMP_NEW_EMP.h"
```

```
int main(int argc, const char * argv[])
{
    @autoreleasepool
    {
        EMP* Obj1 = [[EMP alloc] init];
        [Obj1 setData:122 withName:@"Rajesh" andSal:12345
            withPno:@1001001000];
        [Obj1 displayData];
        [Obj1 retain];
    }
    return 0;
}
```

* Protocol :-

- Protocol will provide set of declaration of the methods which can be implemented by any other classes.
- It is recommended to go for protocols when we are not providing full fledged functionality of any interface. Maximum of cocoa related UIKit ~~is~~ framework having multiple of protocols those required to implement when we are using any UI components like 'textbox', buttons, UI table or any other controllers.

→ When we are working with protocols, we required to follow three procedures i.e.

- ① Creating a protocol
- ② Adopting the protocol.
- ③ Checking type of protocol.

* Syntax to Create a protocol :-

@ protocol Protocol-Name < Class-Name >

declaration

@ end.

* Syntax to Adopt the protocol :-

@ interface NEW-CLASS : BASE CLASS < protocol name >

declaration

@ end.

* Example :

Create a commandline tool appl' with the name
Protocol Ex.

Create a new protocol file with the name StreetLegal.

** Code in StreetLegal.h :-

import < foundation / Foundation.h >

@ Protocol StreetLegal < NSObject >

- (void) signalStop;

- (void) signalLeftTurn;

- (void) signalRightTurn;

@ end.

Create a new class with the name Car, subclass of NSObject.

* Code in Car.h :-

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

@interface Car : NSObject<StreetLegal>
{
}

-(void) signalStop;
-(void) signalLeftTurn;
-(void) signalRightTurn;
@end.
```

* Code in Car.m :-

```
#import "Car.h"

@implementation Car

-(void) signalStop
{
    NSLog(@"Car is Stop");
}

-(void) signalLeftTurn
{
    NSLog(@"Car has taken left turn");
}

-(void) signalRightTurn
{
    NSLog(@"Car has taken right turn");
}

@end
```

Create a new class with Name Bus subclass of NSobject.

* Code in Bus.h :-

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

@interface Bus : NSObject<StreetLegal>
-(void) signalStop;
-(void) signalLeftTurn;
-(void) signalRightTurn;
@end.
```

* Code in Bus.m :-

```
#import "Bus.h"
@implementation Bus
-(void) signalStop
{
    NSLog(@"Bus Stop");
}
-(void) signalLeftTurn
{
    NSLog(@"Bus took left");
}
-(void) signalRightTurn
{
    NSLog(@"Bus took Right");
}
@end.
```

* Code in main.m :-

```
#import <Foundation/Foundation.h>
#import "Bus.h"
#import "Car.h"

int main(int argc, const char * argv[])
{
    @autoreleasepool
    {
        id myobj = [[Bus alloc] init];
        // id<StreetLegal>myobj = [[Bus alloc] init];
        if ([[myobj isKindOfClass:[Car Car class]])]
        {
            [myobj signalStop];
            [myobj signalLeftTurn];
            [myobj signalRightTurn];
        }
        else if ([[myobj isKindOfClass:[Bus class]]]) or else
        {
            [myobj signalStop];
            [myobj signalLeftTurn];
            [myobj signalRightTurn];
        }
        [myobj retain];
    }
    return 0;
}
```

11/10/2013

Delegates

- * Delegate is a generic type of Objective-C which allow to communicate the data betⁿ the objects.
- when we require to pass the data from one class obj to another class object , then it must be possible by using delegates only.
- Delegates are implemented by using protocols only.
- when we are working with cocoa UI controllers then from one control to another control if we required to communicate the data then it must be required to implement by using delegates only.

* Syntax to create a delegate method :-

@protocol DelegateName <NSObject>

 Methods Declaration ;

@ end.

* Adopting the delegate to the class :-

@ interface NEW_CLASS : BASE_CLASS

{

 id <Delegate Name> delegate;

 declaration

}

@ property (nonatomic, assign) id delegate;

@ end.

* Calling the delegate method from some other object :-

@ interface NEW_CLASS : BASE CLASS <DelegateName>

{

}

@ end.

for eg:-

Class A

```
@ protocol ADelegate<NSObject>
-(void)abc;
@ end
@ interface class A : NSObject
{
    id<ADelegate> delegate;
}
@ property (nonatomic, strong) id delegate;
```

Process start.

Class B

```
#import "classA.h"
@ interface class B : NSObject
<ADelegate>
{
    [self delegate]abc];
}
@ end.
```

Process end.

* Memory Management in Objective-C :-

→ In objective-c we having two types of memory management i.e. static memory allocation & dynamic memory allocation.

1) Static memory allocation :-

→ When we are creating the memory at the time of compilation then it is called compile time memory management or static memory allocation.

→ Generally in objective-C, doesn't require to go for static memory allocation.

2) Dynamic Memory Allocation :-

- When we are creating the memory at the time of execution then it is called Dynamic memory allocat?
- In objective-C "NSAutoreleasePool" class is responsible for dynamic memory allocation.
- In Objective-C dynamic memory allocation will handle by reference counting mechanism.

* Reference Count :-

- This is objective-C memory management technique which holds number of counts which is handled by object. i.e. one object is holding how many reference are count value
- When we are allocating the memory for any object then automatically reference count is incremented by 1, when we are decreasing the memory then automatically reference count value decrease by 1.
- whenever one obj reference count value is 0 (zero) then it is called object is free.
- When we are using the "release" method, then it will deallocate memory for the object & reference count also decrease by 1.

- when we are calling "retain" method then memory will deallocated for the object but reference count will not decrease.
- when we are working with autoreleasepool obj then always recommended to use "drain" method only. because "drain" method will call garbage collection system.
- Garbage collector is a automated process which will destroyed the appl" related memory when the appl" is terminated.
- In place of creating object of type NSAutoreleasePool then always recommended to go for "@autoreleasepool" block.

```
@autoreleasepool
{
    // Code benefitting from a local autoreleasepool.
}
```

- In place of holding the reference count manually & deallocating by using "release" or "retain", it is recommended to go for ARC (Auto Reference Counting).
- According to autorelease method , whenever reference count value is zero, then automatically object will be destroyed.
- If we require to make enable ARC, then go for Select Project file, choose build settings & go to "APPLE LLVM ~~Compiler~~ Compiler 4.2 - language" & choose Objective-C Auto Reference Count.(ARC) Yes or No

FRAMEWORKS

16/10/2013

- A framework is a collection of classes, methods & some protocols.
- In MAC OS, we having more than 90 frameworks.
- In implementation when we are designing any kind of applications then depending on the requirement, that specific framework required to import in the appl'.

* Procedure to import a framework in Applications :-

- Select "Project template" file, then choose "Build" phase.
- Under "Build" phase, go to "Linker Binary with Libraries" & click on "Add" button.

* Common using frameworks in iOS Applications :-

1) UIKit framework (UIKit.framework) :-

- This framework is available for all iOS appl' for cocoa events.
- UIKit framework - will provide all user Interface related components.

2) Map Kit framework (MapKit.framework) :-

By using this framework, Map related events can be handle, including Google map services also.

3) Message UI framework (MessageUI.framework) :-

→ By using this framework, we can provide message services.

4) Address Book UI framework :- (AddressUI.framework) :-

→ When we required to replicate the functionality of Address Book, then go for Address Book UI framework.

5) Game Kit framework (GameKit.framework) :-

→ Gaming appl^ related services are available in this framework.

6) iAd framework (iAd.framework) :-

→ By using this framework we can provide advertisements in iOS appl^.

7) Core Video framework (CoreVideo.framework) :-

→ By using this framework, we can provide media player services.

8) Core Text framework (CoreText.framework) :-

→ By using this framework, we can provide graphics into text.

9) Image I/O framework (ImageIO.framework) :-

→ By using this framework, we can manipulate the images.

10) Assets Library framework (`AssetsLibrary.framework`) :-

→ By using this framework, we can interact with device's specific operations.

11) Core Graphics framework (`CoreGraphics.framework`) :-

→ By using this framework, we can provide 2D graphics API functionalities into the applications.

12) Quartz Core framework (`QuartzCore.framework`) :-

→ This framework will provide animation related APIs.

13) OpenGL ES framework (`OpenGL ES.framework`) :-

→ This framework will provide graphics components in Gaming appl'.

14) Core Audio framework (`CoreAudio.framework`

`AudioToolbox.framework`

`AudioUnit.framework`) :-

→ By using this framework, we can load audio services related functionalities.

15) Media Player framework (`MediaPlayer.framework`) :-

→ This framework can play different types of file format

16) CF Network framework (`CFNetwork.framework`) :-

→ By using this framework, we can handle network services like requesting URL & getting response from URL.

17) Core Data Framework (CoreData.framework) :-

- By using this framework, we can handle database through core data.

18) Core Foundation Framework (CoreFoundation.framework) :-

- This framework will provide 'C' based functionalities like datatypes, string manipulations, loops, socket programming & some other services which are provided by objective C.

19) Core Telephony f/w :- (CoreTelephony.framework) :-

- This f/w will provide call services related information i.e. call register option.

20) EventKit f/w :- (EventKit.framework) :-

- By using this f/w we can handle calendar & alarm related API services.

21) CoreLocation f/w : (CoreLocation.framework) :-

- By using this f/w we can find geographical location. (GPS).

22) SQLite Library :-

- By using this f/w we can work with light weighted database in 'C' app!
objective

* Foundation Framework :-

- This framework will provide 'C' type primitive datatypes along with Objective C related, object type data.
- When we are working with primitive datatypes of C then, we can't work on message passing system.
i.e. it doesn't allow to pass message from one variable to another variable.
- When we are working with Objective C appl', then every variable required to communicate with other variables, so we need object type data. like NSNumber, NSString, NSArray, NSData etc.

* Mutable :-

Whenever the object is allowed to modify the data, then it is called Mutable, if doesn't allow to modify then it is called immutable.

* Number Objects (NSNumber) :-

- It is a predefined class which is available in foundation fw.
- This class related object will allow to hold any kind of 'C' primitive data type.

* ex → Create Commandline tool appl' with name NSNumberEx.

```

#import <Foundation/Foundation.h>
int main (—)
{
    @autoreleasepool
    {
        NSNumber * charNum, * shortInt, * unsignedIntNum;
        charNum = [NSNumber numberWithChar: 'A'];
        NSLog(@"char value: %c, [charNum charValue]");
        shortInt = [NSNumber numberWithInt: 135];
        NSLog(@"Short Value: %hi, [shortInt shortValue]");

        unsignedIntNum = [NSNumber numberWithUnsignedInt: 65355];
        NSLog(@"unsigned value: %u", [unsignedIntNum unsignedIntValue]);

        NSNumber * intValue;
        intValue = [NSNumber numberWithInt: 10];
        if ([intValue compare: shortInt] == NSOrderedAscending)
        {
            NSLog(@"%i < %hi", [intValue intValue], [shortInt
                                                shortValue]);
        }
        else if ([intValue compare: shortInt] == NSOrderedDescending)
        {
            NSLog(@"%i > %hi", [intValue intValue], [shortInt shortValue]);
        }
        else if ([intValue compare: shortInt] == NSOrderedSame)
        {
            NSLog(@"%i == %hi", [intValue intValue], [shortInt shortValue]);
        }

        NSNumber * floatNum, * doubleNum;
        floatNum = [NSNumber numberWithFloat: 123.45];
        doubleNum = [NSNumber numberWithDouble: 123456.12e+34];
        NSLog(@"float value: %f", [floatNum floatValue]);
        NSLog(@"double value: %e", [doubleNum doubleValue]);
    }
}

```

NSNumber Reference

17/10/2013

* description :-

- when we are calling description method on NSNumber object, then it prints description of data in the form of string.

* stringValue :-

- when we call stringValue method on NSNumber object then it returns string type data.

* Compare :-

- By using this method, we can compare two NSNumber object data.
- when we are working with 'compare' method, then it returns NSComparisonResult value.
- NSComparisonResult value can be NSOrderedDescending or NSOrderedAscending or NSOrderedSame.

* isEqualToNumber :-

- By using this method we can compare two NSNumber objects. but this method returns boolean data. i.e YES or NO.

* NSMutableString :-

- It is a predefined class which is available in Foundation framework.
- By using string class object we can manipulate string type data more efficiently.

Note: C type string will works with the help of ASCII code i.e. 1 byte data, NSString will works with the help of Unicode data i.e. 2 bytes value.

- When we are working with NSString object then it is not possible to change content because by default it is immutable.
- Whenever we require to change string content throughout the program depends on requirement, then go for NSMutableString object.

* Create a commandline tool appl' with the name NSStringEx.

⇒ Code in main.m:-

```
#import <Foundation/Foundation.h>
int main(—)
{
    @autoreleasepool
    {
        NSString *str1 = @"This is string A";
        NSString *str2;
        str2 = [NSString stringWithString: str1];
        NSLog(@"%@", str2);
        NSLog(@"%@", str2);

        short int l;
        l = [str2 length];
        NSLog(@"Length of String : %hi", l);

        NSString *str3 = @"This is string B";
        // str3 = [str3 stringByAppendingString: str1];
        str3 = [str3 stringByAppendingFormat: @" is appending %@", str1];
        NSLog(@"%@", str3);
    }
}
```

```

NSString *str4;
str4=[NSSString stringWithFormat:@"%@", @"SUM OF %d + %d = %d",
      10, 20, 10+20];
NSLog(@"%@", @"Test string: %@", str4);
NSLog(@"%@", @"Lower Case: %@", [str4 lowercaseString]);
NSLog(@"%@", @"Upper Case: %@", [str4 uppercaseString]);

char temp[100];
strcpy(temp, [str2 UTF8String]);
NSLog(@"%@", @"Temp String: %@", temp);

NSSString *tempStr;
tempStr=[NSSString stringWithUTF8String: temp];
NSLog(@"%@", @"Temp NSSString: %@", tempStr);
if ([str1 compare:str2]==NSOrderedSame)
{
    NSLog(@"%@", @"Both strings are same");
}
if ([str1 isEqualToString:str2]==YES)
{
    NSLog(@"%@", @"Both are same");
}
return 0;
}

```

* Creating two user defined methods for NSString class.
i.e. reverseString & changeCase.

- ⇒ Create a commandline tool appl" with the name NSString Class-Enhancement.
- ⇒ Create a new obj. C category class with the name ~~StringReverse~~ StringRevCng. category on NSString.

* Code in NSString + StringRevCng.h :-

```
#import <Foundation/Foundation.h>
@interface NSString (StringRevCng)
+(NSString*) StringReverse:(NSString*)temp;
-(NSString*) StringChangeCase;
@end.
```

* Code in NSString + StringRevCng.m :-

```
#import "NSString + StringRevCng.h"
@implementation NSString (StringRevCng)
+(NSString*) StringReverse :(NSString *)@temp
{
    char *pt2 = NULL;
    char ch;
    int i, l;
    l = (int)[temp length];
    pt2 = (char *) calloc(l, sizeof(char));
    strcpy(pt2,[temp UTF8String]);
    for(i=0; i<l/2; i++)
    {
        ch = pt2[i];
        pt2[i] = pt2[l-i-1];
        pt2[l-i-1] = ch;
    }
}
```

```
NSString *result = [NSString stringWithUTF8String : pt2];
```

```
free(pt2);
```

```
pt2 = NULL;
```

```
return result;
```

```
}
```

```
- (NSString *)StringChangeCase.
```

```
{
```

```
char *pt2 = NULL;
```

```
int i, l;
```

```
l = (int) [self length];
```

```
pt2 = (char *) malloc(l, sizeof(char));
```

```
strcpy(pt2, [self UTF8String]);
```

```
for (i = 0; i < l; i++)
```

```
{
```

```
if (pt2[i] >= 'A' && pt2[i] <= 'Z')
```

```
pt2[i] = pt2[i] + 32;
```

```
else if (pt2[i] >= 'a' && pt2[i] <= 'z')
```

```
pt2[i] = pt2[i] - 32;
```

```
}
```

```
self = [NSString stringWithUTF8String : pt2],
```

```
free(pt2);
```

```
pt2 = NULL;
```

```
return self;
```

```
}
```

```
}
```

```
@ end.
```

* Code in main.m :-

```
#import <Foundation/Foundation.h>
#import "NSStrint+String Reverse"
int main ( )
{
    @autoreleasepool
    {
        NSStrint *obj1 = @"welCome";
        NSStrint *obj2 = [NSStrint String Reverse: obj1];
        NSLog(@"%@", [obj2 reverseString]);
        NSStrint *obj3 = @"Hello Test";
        obj3 = [obj3 String ChangeCase];
        NSLog(@"%@", [obj3 capitalizedString]);
    }
    return 0;
}
```

18/10/2013

* Ex. with NSMutableString :-

```
#import <Foundation/Foundation.h>
int main ( )
{
    @autoreleasepool
    {
        NSStrint *str1 = @"This is string A";
        NSStrint *search, *replace;
        NSMutableString *mstr;
        NSRange substr;
    }
}
```

```
mta = [NSMutableString stringWithString : str];
NSLog(@"%@", mta);
NSString * str2 = @"Find string B";
[mta insertString @"mutable" atIndex : 7];
NSLog(@"%@", mta);

NSString * str2 = @"and string B";
[mta insertString : str2 atIndex : [mta length]];
NSLog(@"%@", mta);

[mta appendString : @" & string C"];
NSLog(@"%@", mta);

[mta deleteCharacterInRange : NSMakeRange (16, 13)];
NSLog(@"%@", mta);

substa = [mta rangeOfString : @" string B and "];
NSLog(@"%@", substa.location, substa.length);
if (substa.location != NSNotFound)
{
    [mta deleteCharactersInRange : substa];
    NSLog(@"%@", mta);
}

[mta replaceCharactersInRange : NSMakeRange (8, 8) withString :
 @" a mutable string"];
NSLog(@"%@", mta);

Search = @"This Is";
replace = @" An example of";
```

```

C Substr = [mstr rangeOfString : search];
if ( substr.location != NSNotFound )
{
    [mstr replaceCharactersInRange : substr.withString : replace];
    NSLog(@"%@", mstr);
}
search = @"a";
replace = @"x";
substr = [mstr rangeOfString : search];
while (substr.location != NSNotFound)
{
    [mstr replaceCharactersInRange substr withString : replace];
    substr = [mstr rangeOfString : search];
}
NSLog(@"%@", mstr);
}
return 0;
}

```

* NSStringEx 2 :

```

#import <Foundation/Foundation.h>
int main(—)
{
    @autoreleasepool
    {
        NSString * str1 = @"This is string A";
        NSString * res;
        NSRange subRange;
        res = [str1 substringToIndex : 4];
        NSLog(@"%@", @"Chars from index 5 of str1 : %@", res);
    }
}

```

```

a8 = [str.substringFromIndex:8]sub
NSLog(@"%@", "char from index 5 of str: %@", a8);

a8 = [str.substringFromIndex:8]substringToIndex:6];
NSLog(@"%@", "chars from index 8 through 13: %@", a8);
⇒ An easy way to do the same :-  

a8 = [str substringWithRange: NSMakeRange (8, 6)];
NSLog(@"%@", "char from index 8 through 13: %@", a8);

SubRange = [str rangeOfString: @"String A"];
NSLog(@"%@", "index %lu, length is %lu", subRange.location,
         subRange.length);

SubRange = [str rangeOfString: @"String B"];
if (SubRange.location == NSNotFound)
    NSLog(@"%@", "String Not Found");
else
    NSLog(@"%@", "index %lu, length %lu", subRange.location,
         subRange.length);
}
return 0;
}

```

* NSRange :-

- It is a predefined structure which is available in foundation framework.
- By using this structure, we can find range of a substring.

- Range is a combination of two NSUInteger values
i.e. location and length.
- When we are finding the substring from given string
if it is not occur then location value is "NSNotFound."
- NSNotFound equivalent value is "NSIntegerMax" value

* Common methods of NSString class :-

1) StringWithString :-

→ By using this method, we can create a string object with another string content. (class method).

2) length :-

→ By using this method, we can find length of a string.

3) StringByAppendingString :-

→ By using this method, we can append a string to another string.

→ String appending always takes place at the end of the current string only.

4) StringWithFormat :-

→ It is a class method which is used to create a string obj in customized format. (user defined format string data).

5) String By Appending Format :-

→ By using this method, we can append - customized string format data to existing string.

6) lowercaseString :-

→ This method will convert current string content in lower case format

7) uppercaseString :-

→ This method will convert current string data in uppercase.

8) stringWithUTF8String :-

→ It is a class method which is used to create NSString obj from 'C' type string.

9) → This method will convert, ASCII coded string to Unicoded string. (2byte to 2bytes).

9) UTF8String :-

→ This method is used to convert unicode string to C type string. (NSString to char*/2b to 2b).

10) isEqualToString :-

→ This method is used to compare both strings are same or not.

→ This method returns boolean value i.e. YES or NO.

11) Compare :-

- By using this method, we can compare 2 strings.
- Compare method returns, NSComparisonResult value
- NSComparisonResult value can be NSOrderedAscending or NSOrderedSame or NSOrderedDescending.

12) substringToIndex :-

- By using this method, we can extract the substring upto to given location.

13) substringFromIndex :-

- By using this method, we can extract the substring from given location to end of the string.

14) substringWithRange :-

- By using this method, we can extract the substring in specific range.
- When we are using this method, we require to pass NSRange value i.e. location, length.

15) rangeOfString :-

- This method will return substring range in the form of NSRange data i.e. location & length.

* Common methods on NSMutableString :-

1) stringWithString :-

→ By using this class method we can create NSMutableString object.

2) insertString :-

→ By using this method, we can insert a string in mutableString data.

→ When we are using this method, we required to pass index value, if we are inserting at ending posⁿ then index must be length of the string.

3) appendString :-

→ By using this method, we can append the string at end of the string data.

4) deleteCharactersInRange :-

→ By using this method, we can delete mutable string content in specific range.

→ When we are using this method, we require to pass NSRange value.

→ NSRange value can be passed by using NSMakeRange function.

19/10/2013

5) replaceCharacterInChange:-

- By using this method, we can replace the character in specific range.
- When we are using this method we require to pass NSRange value i.e. location & length.

* NSArray:-

- It is a predefined class which is available in foundation framework, by using this class object we can store any kind of list of objects.
- NSArray is a dynamic data structure which can manage number of elements at runtime.

* Ex: Create a commandline appl" with name NSArrayEx:-

* Code in main.m:-

```
#import <Foundation/Foundation.h>
int main (—)
{
    @autoreleasepool
    {
        NSArray *appleObjects = [[NSArray alloc] initWithObjects:
            @{@"Apple", @"Mac", @"iPhone", @"iPad", @"iPod",
            @"iOS"; nil}];
        for(int i=0; i<[appleObjects count]; i++)
        {
            NSLog(@"index: %i, object: %@", [appleObjects objectAtIndex:i]);
        }
        [appleObjects release];
    }
    return 0;
}
```

- When we are working with NSArray related class object we can't update elements because it is a immutable.
- At the time of execution of appl? if we required to update the data then go for NSMutableArray class.

* Common Methods of NSArray class :-

1) initWithObjects :-

→ This method will allows to create NSArray from any other list of Array elements.

2) initWithArray :-

→ This method will allow to create new array object from existing array.

3) initWithContentsOfFile :-

→ By using this method, we can load array data from file.

4) initWithContentsOfURL :-

→ By using this method, we can create list of array element from NSURL.

5) arrayWithContentsOfFile :-

→ It is a class method which allow to create the array elements from file.

6) arrayWithContentsOfURL :-

→ By using this class method we can load array elements from NSURL.

7) arrayWithObjects :-

→ By using this class method, we can create list of array elements.

8) count :-

→ It returns number of elements count which is available in current array object.

9) objectAtIndex :-

→ By using this method, we can extract the elements with the help of index.

→ When we are using this method, index value must be within the range only, out of range if we are given then appl' will crash.

* NSMutableArray :-

```
#import <Foundation/Foundation.h>
int main(→)
{
    @autoreleasepool
}
NSMutableArray *appleObjects = [[NSMutableArray alloc]
    initWithObjects: @"Apple", @"Mac", @"iPhone",
    @"iPad", @"iPod", nil];
for(int i=0; i < [appleObject count]; i++)
{
    NSLog(@"%@", [appleObject
        objectAtIndex:i]);
}
```

```
[appleObjects removeAllObjects];
NSLog(@"%@", [appleObjects count]);
[appleObject addObject:@"Apple"];
[appleObject addObject:@"Mac OSX"];
[appleObject addObject:@"Mac Mini"];
NSLog(@"Array Objects: %@", appleObjects);
[appleObjects removeLastObject];
NSLog(@"Array Objects: %@", appleObjects);
[appleObjects insertObject:@"iOS" atIndex:0];
[appleObjects insertObject:@"iPhone" atIndex:[appleObjects
                                         count]];
NSLog(@"List: %@", appleObjects);
[appleObjects removeObjectAtIndex:0];
NSLog(@"list2 %@", appleObjects);
}
[appleObject release];
return 0;
}
```

20/10/2013

Cocoa

- Cocoa is an applⁿ environment for both OS X & iOS.
- Cocoa is a set of object oriented frameworks, libraries that provides a runtime environment for applⁿ which is running in OS X & iOS.
- Cocoa is a permanent applⁿ development environment for Mac OS X & iOS.
- Cocoa is a combination Foundation, AppKit, UIKit frameworks
- In OS X, cocoa is used with the help of Foundation & AppKit classes.
- In iOS, cocoa is used with the help of Foundation & UIKit classes.
- In both type of operating system, cocoa will provide Graphical User Interface (GUI).
- We cannot develop any cocoa applⁿ without using AppKit classes in OS X.
- We cannot develop any cocoa applⁿ without using UIKit classes in iOS.
- Cocoa will provide event driven architecture for OS X applⁿ
- Cocoa will provide MVC architecture for iOS applⁿ.

Cocoa Architecture for OS X

User Experience

Aqua

Dashboard

Spotlight

Accessibility

Application frameworks.

Carbon

Java

AppKit

core frameworks

Graphics & Media (Application Services)

Core Animation

Core Image

Core Video

Quick Time

OpenGL

Quartz

Core Audio

core Services

carbon core

Launch Services

Core Foundation

Foundation

core framework

DarWin

- In OSX, Cocoa having two main parts called Application services & Core Services.
- Appl' services will provide UI, App framework, GUI & media services for application.
- Core services will provide system related process.

* AppKit :-

- It provides object & appl' display with UI & event handling process also.

* Foundation :-

- It is a core services layer which provides object for primitive datatypes, collections & operating system services.

* Core Foundation :-

- Many classes Foundation framework will work on core foundation layer.

* Carbon Core :-

- This part provides any kind of system services like, process, file systems or any other system level programming.

* Core Graphics :-

- The Cocoa drawing & image related classes will works with the help of core graphics which provides animations 2D related graphics & windows services

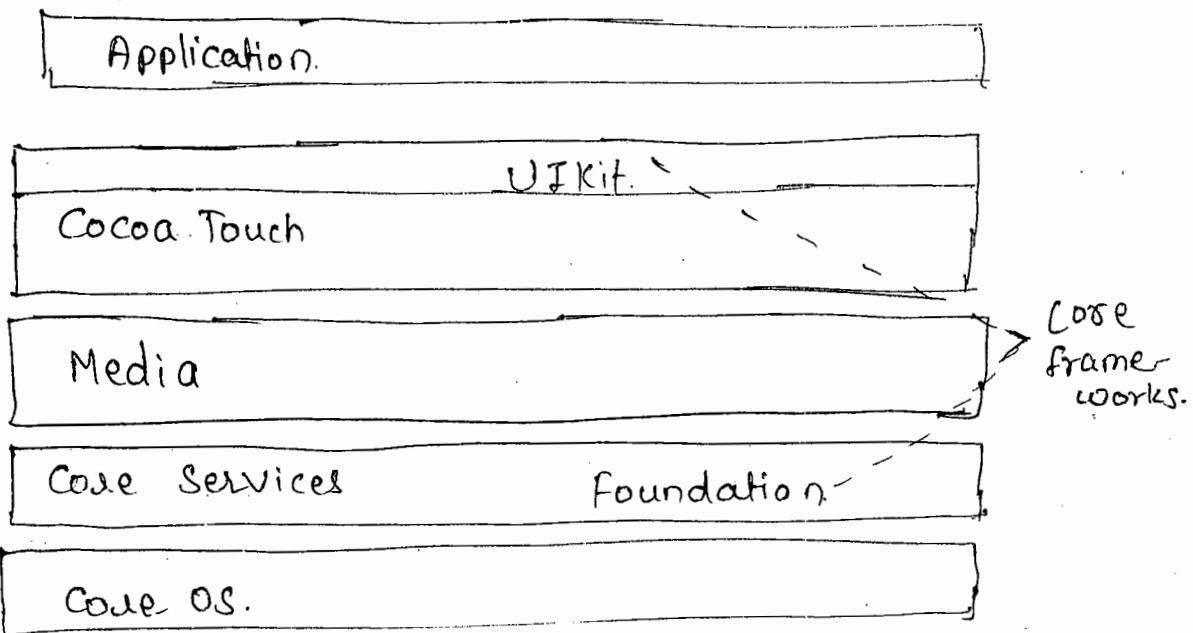
* Launch Services :-

- To get the icons & applⁿ related documents to the applⁿ, we need to use launch services
- Cocoa is commonly referred for combination, Appkit frameworks & it is used to develop Mac & OSX applⁿ.

Cocoa Touch

- Cocoa touch is a user Interface framework which is provided by Apple for designing native applⁿ of iPhone, iPad, iPod touch.
- It is developed in Objective C language, based on Mac OS.
- Cocoa touch will follow MVC architecture.
- Cocoa touch is a combination of Foundation & UIKIT frameworks which is used to develop iOS applⁿ.

Cocoa Touch Architecture for iOS



* Core OS :-

→ This layer contains the kernel, file system, networking infrastructure, security, power management & a number of device drivers.

* Core Services :-

→ This framework will provide core services like string manipulation, collections, networking, URL utilities, contacts management & preference.

* Media :-

→ This layer provides graphical & multimedia services to cocoa touch including Core graphics, Core text, OpenGL, Core animations, AV Foundation, Core audio, & Video player services.

* Code Touch :-

→ The framework in this layer, directly supports appl' based iOS features including GameKit, MapKit.
e iAd

* UIKit :-

→ This framework will provides the objects & appl' display along with UI & event handling process.

* Foundation :-

→ This framework will provide objects, primitive data types, collections , OS services & it is a part of core Foundation framework.

→ AppKit related some of the functionalities may not be available in UIKit & UIKit related functionalities may not be available in AppKit.

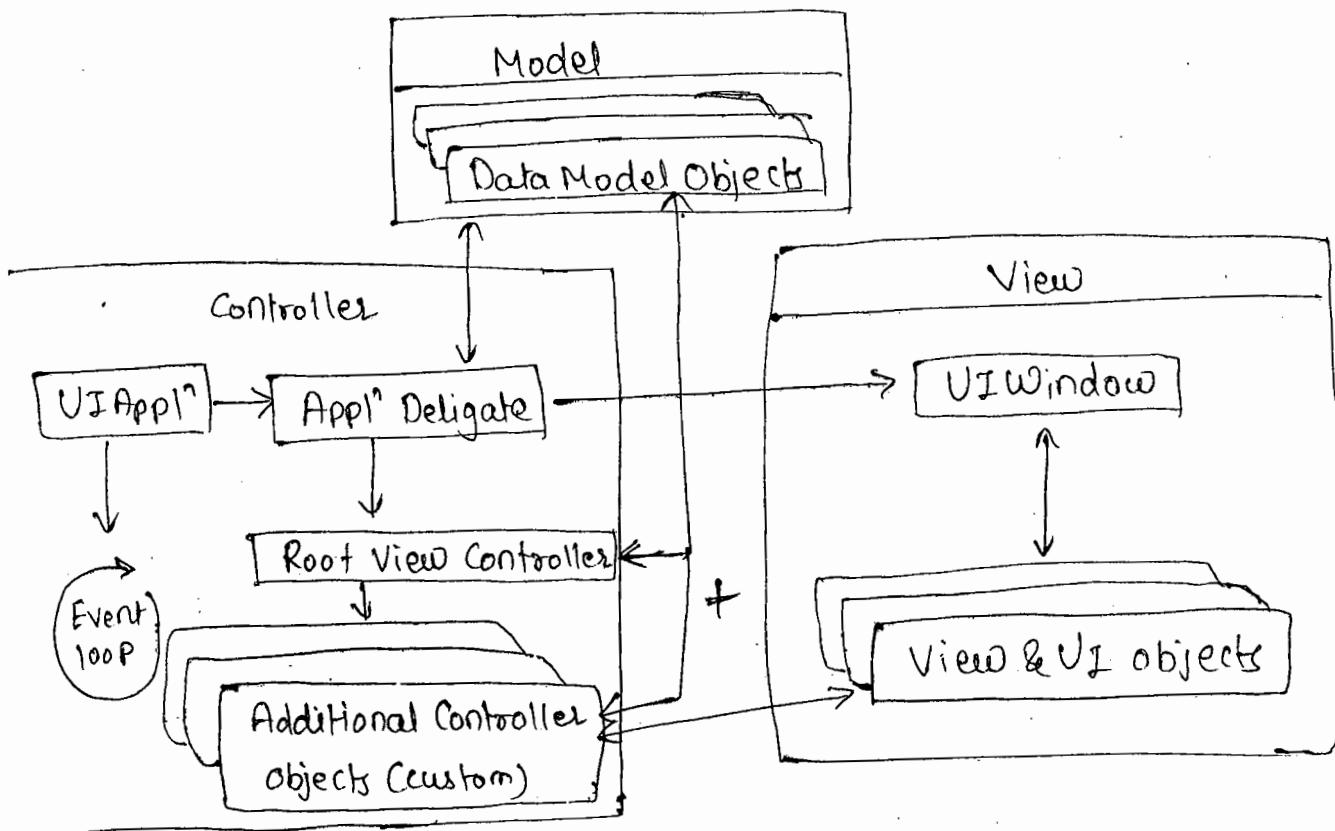
Model View Controller (MVC)

→ MVC is a design pattern which assigns three roles for any kind of appl'. i.e.

- ① Model
- ② View
- ③ Controller.

- This pattern defines not only the rules, it defines the way of communicating the objects with another objects.
- Each of three types of objects is separated from others by abstracting & communicating with objects across the appl'?
- The basic advantage of MVC design is appl' extensibility with other appl'.
- Many of technologies including cocoa frameworks & architectures are designed based on MVC only.
- As a programmer it is our responsibility to design custom objects with the help of MVC architecture.

MVC Architecture



* Model Objects :-

- Model objects encapsulate the data specific to an app' and provides set of logic for processing the data.
- A model object should have no explicit connection to the view object that presents its data & allows to user edit the data.

* View Objects :-

- This object only visible to end user.
- For any kind of user action view objects only responds.
- The basic purpose of view object is display the data which is available in model object & allow to edit the information by user if it is required.

* Controller Objects :-

- Controller object will perform, How the model is presented to the view.
- Communicating
 - ① A controller object is intermediate b/w multiple view objects & model objects.
- The controller only responsible for changing the model object data or changing the view object data along with corresponding model & view objects.
- When the view is updated then controller will takes the action to update the model object, when model object is changed then controller will take the action to ~~view~~ view to update

21/10/2013

* Window based Applications in iOS or Single view Applications.

- Create a new iOS app" of type single view with the name windowApp1.
- Deselect autoreference count & story board options.

* Code in Viewcontroller.h :-

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

```
@interface ViewController : UIViewController
```

```
{
```

```
    IBOutlet *viewable;
```

```
    UIButton *viewButton;
```

```
}
```

```
@property (nonatomic, retain) IBOutlet *viewable;
```

```
@property (nonatomic, retain) UIButton *viewButton;
```

```
-(void)ButtonClick;
```

```
@end
```

* Code in Viewcontroller.m :-

```
@implementation ViewController
```

```
@synthesize viewable, viewButton;
```

* Code in viewDidLoad method :-

```
[super viewDidLoad];
```

```
viewable = [[[UILabel alloc] initWithFrame:CGRectMake(20, 60, 200, 60)] autorelease];
```

```
[viewLabel setText:@"label Text here"];
[viewLabel setBackgroundColor:[UIColor clearColor]];
[self.view addSubview:viewLabel];

viewButton=[[UIButton buttonWithType:UIButtonTypeRoundedRect]
            autorelease];
[viewButton setFrame:CGRectMake(40, 120, 80, 45)];
[viewButton setTitle:@"Click me" forState:UIControlStateNormal];
[viewButton addTarget:self action:@selector(buttonClick)
            forControlEvents:UIControlEventTouchUpInside];
[self.view addSubview:viewButton];
}

-(void)buttonClick
{
    if ([viewLabel.text isEqualToString:@"Button Clicked"]){
        viewLabel.text=@"label Text here";
    }
    else
        [viewLabel setText:@"Button Clicked"];
}
```

22/10/2013

* @ class :-

- By using this preprocessor directive we can provide forward declaration of any other classes without using inheritance.
- In implementation when we required to create a object of a class in another class without using inheritance then recommended to go for '@ class'.
- UIResponder class will provide event handling process by using user action.
- UIApplicationDelegate is a protocol which allows to implement app life cycle related methods.
- AppDelegate is a current application delegate class which is inherited from UIResponder & UIApplication.
- A view is a visual objects that creates user interface of an iOS appl'.
- Views are defines within the specific rectangle area of a screen.
- All views are subclasses of UIKit, UIView classes.
- UIWindow class provides the surface on screen which allow to add view components to display.
- In any kind of ios appl' only one UIWindow object is available.
- We cannot place any kind of UIViewController directly on UIWindow.

- UIWindow must be fit entire screen of the device.
- UIView class provides model UI for displaying the content view.
- View-Controller class is a subclass of UIViewController which allows to add any type of UIControllers.
- AppDelegate is a subclass of UIResponder which allows to handle surface and App life cycle event.

* iOS App LifeCycle :-

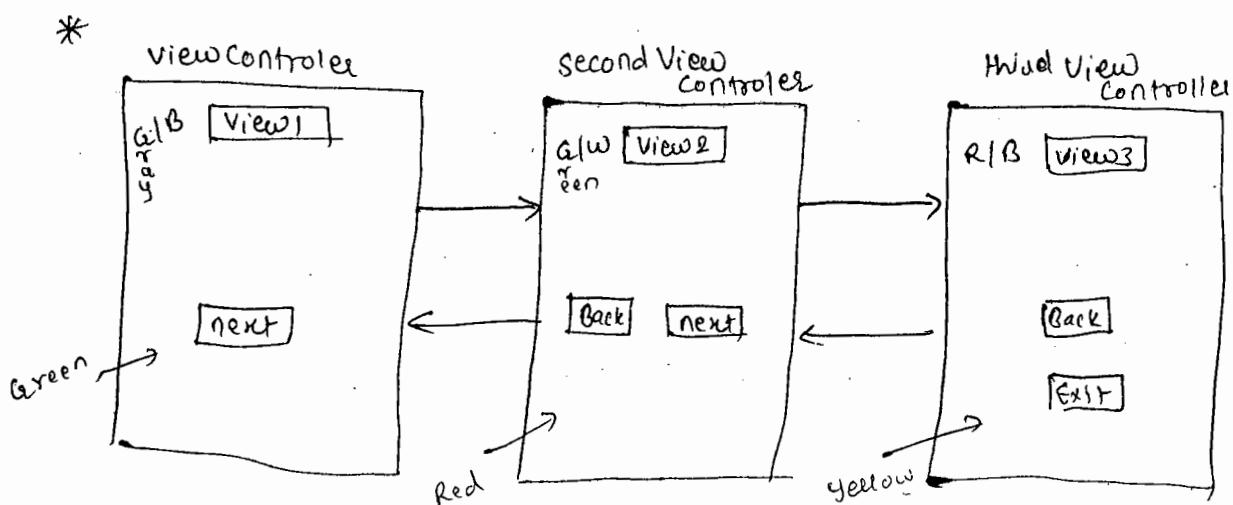
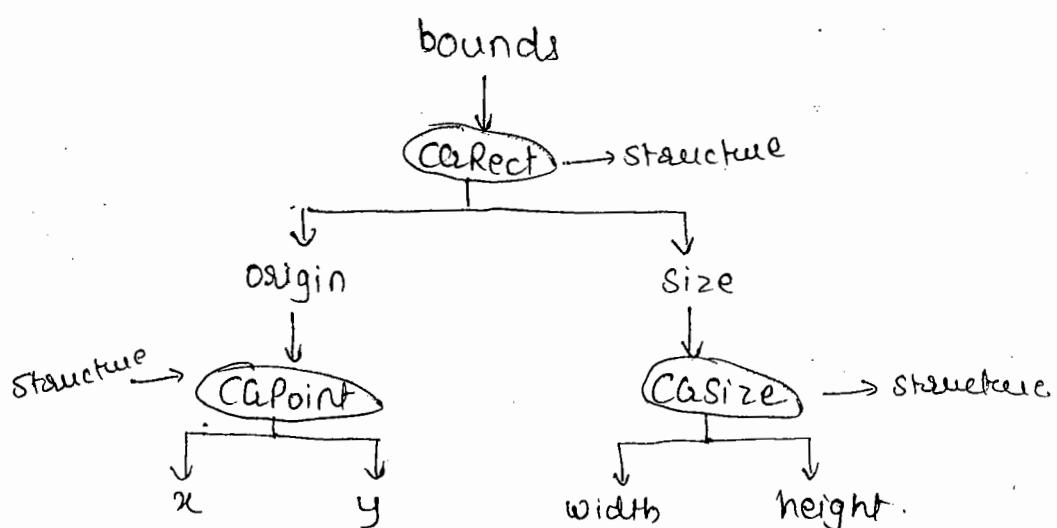
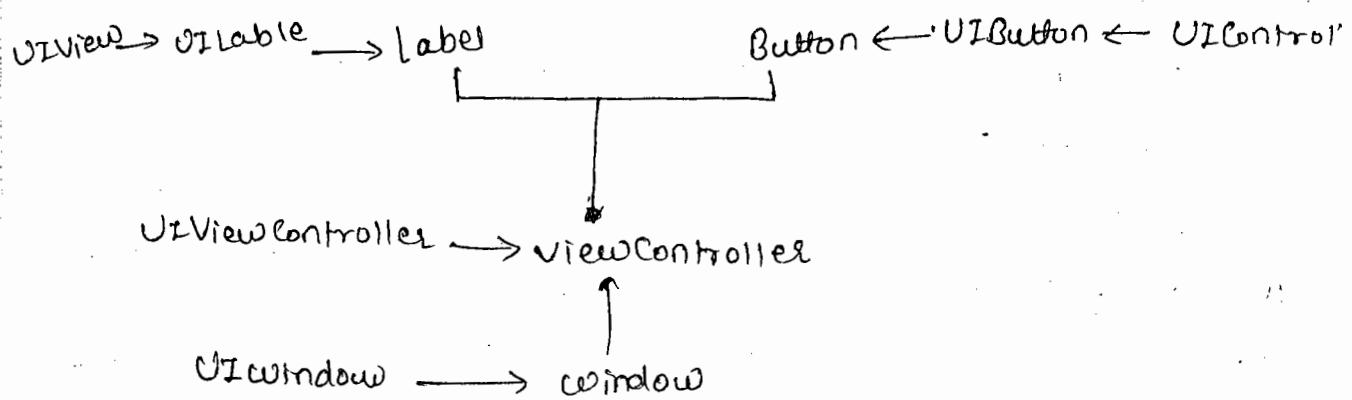
- App lifecycle events are available in AppDelegate.m file.
- When we are launching the app's first time, then control will goes "application: didFinishLaunchingWithOptions" method.
- This method will called only once when we starts the app's first time.
- All app's related initialization must be take place within this method only.
- The next method is called during the app's launch i.e. "ApplicationDidBecomeActive" (if URL is not present).
- This method may be called multiple times.
- If the URL is available then control will enter into OpenURL or handleOpenURL method, from here control will goes to "ApplicationDidBecome Active" method.

↑ When the user hits Home button or if interruption occurs then it goes to "ApplicationWill"

- After ApplicationDidBecomeActive mode control will goes to ApplicationWillEnterForeground method for handling UI events.
- When user hits Home button or any interruption occurs using phone calls or messages then control will goes to ApplicationWillResignActive method.
- In this method only we required to pause appl' related tasks if anything is in running status.
- From Resign Active Method control will goes to ApplicationDidEnterBackground method, this method can hold only for 5 secs., so if we required more than this time, then we required to handle "beginBackgroundTaskWithExpirationHandler" method.
- When the appl' enters into background if the memory is not available then appl' can be terminated by OS.
- When we double tap the Home button then appl' will be relaunched, when we are relaunching the appl' the control will enter into foreground method, from there openURL or applicationDidBecomeActive method will call.
- Whenever the appl' is terminating then, applicationWillTerminate method will call.
- whenever app having memory related problem then applicationDidReceiveMemoryWarning method will called.

→ In implementation when we are working with push notification services then we require to handle following methods.

- * application: didReceiveRemoteNotification
- * application: didReceiveLocalNotification



* Create a ^{empty} single view App under iOS category with the name windowApp for device iPhone.

→ When we are working with empty app only AppDelegate class will be constructed.

→ Add a new class file with the name FirstViewController with .xib file.

→ When we create a new class then automatically three files are created i.e. FirstViewController.h, FirstViewController.m, FirstViewController.xib.

* Code in AppDelegate.h :-

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

```
@class FirstViewController;
```

```
@interface AppDelegate : UIResponder <UIApplicationDelegate>
```

```
@property (strong, nonatomic) UIWindow *window;
```

```
@property (strong, nonatomic) FirstViewController *viewController;
```

```
@end.
```

* Code in AppDelegate.m :-

```
#import "FirstViewController.h"
```

```
-(void) dealloc
```

```
{
```

```
[_viewController release];
```

```
}
```

```
didFinishLaunchingWithOptions
```

```
{
```

```
self.viewController = [[FirstViewController alloc] initWithNibName:  
Name:@"FirstViewController" bundle:nil];
```

```
self.window.rootViewController = self.viewController;
```

}

* Code in FirstViewController.h :-

23/10/2013

```
@ interface FirstViewController : UIViewController  
{  
}  
  
@property (nonatomic, retain) UITableView *table;  
@property (nonatomic, retain) UIButton *nextButton;  
-(void) nextButtonClicked;  
@ end.
```

=> create a new viewController to the project with the name SecondViewController without .xib file.

* Code in FirstViewController.m :-

```
#import "FirstViewController.h"  
#import "SecondViewController.h"  
@ interface FirstViewController()  
@ end.  
@ implementation FirstViewController  
@synthesize table, nextButton;
```

* Code in viewDidLoad method :-

```
{  
    [super viewDidLoad];  
    [self.view setBackgroundColor:[UIColor greenColor]];
```

```
label=[[UILabel alloc] initWithFrame:CGRectMake(40, 30, 200, 80)];
[ label setText: @"View 1"];
[ label setBackgroundColor:[UIColor grayColor]];
[ label setTextColor:[UIColor blueColor]];
[ label setTextAlignment:NSTextAlignmentCenter];
[ self.view addSubview:label];

nextButton=[UIButton buttonWithType:UIButtonTypeRoundedRect];
[nextButton setFrame:CGRectMake(40, 200, 60, 20)];
[nextButton setTitle:@"Next" forState:UIControlStateNormal];
[nextButton setTitleColor:[UIColor redColor] forState:
    UIControlStateHighlighted];
[nextButton addTarget:self action:@selector(nextButtonClicked)
    forControlEvents:UIControlEventTouchUpInside];
[ self.view addSubview:nextButton];
}

-(void)nextButtonClicked
{
secondViewController * nextView=[[SecondViewController alloc]
    initWithNibName:nil bundle:nil];
[ self.view addSubview:nextView.view];
}

-(void)dealloc
{
    [label release];
    [nextButton release];
    [super dealloc];
}
```

* Code in SecondViewController.h :-

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

@interface SecondViewController : UIViewController
{
}

@property (nonatomic, retain) UILabel *label;
UIButton *nextButton;
UIButton *backButton;

-(void)nextButtonClicked;
-(void)backButtonClicked;

@end.
```

☞ Add a new ViewController to the project with the name thirdViewController without .xib file.

* Code in SecondViewController.m :-

```
#import "SecondViewController.h"
#import "ThirdViewController.h"

@implementation SecondViewController

@synthesize label, backButton, nextButton;

- (void)viewDidLoad {
    [super viewDidLoad];
    self.view.backgroundColor = [UIColor redColor];
    label = [[UILabel alloc] initWithFrame:CGRectMake(40, 30, 200, 50)];
    [label setText:@"View 2"];
    [label setBackgroundColor:[UIColor greenColor]];
}
```

```
[label setTextColor : [UIColor whiteColor]];
```

```
[label setTextAlignment : NSTextAlignmentCenter];
```

```
[self.view addSubview : label];
```

```
backButton = [UIButton buttonWithType:UIButtonTypeRoundedRect];
```

```
[backButton setFrame : CGRectMake(40, 200, 60, 40)];
```

```
[backButton setTitle : @"Back" forState : UIControlStateNormal];
```

```
[backButton setTitleColor : [UIColor redColor] forState :  
UIControlStateHighlighted];
```

```
[backButton addTarget : self action : @selector(backButtonClicked)  
forControlEvents : UIControlEventTouchUpInside];
```

```
[self.view addSubview : backButton];
```

```
nextButton = [UIButton buttonWithType:UIButtonTypeRoundedRect];
```

```
[nextButton setFrame : CGRectMake(150, 200, 60, 40)];
```

```
[nextButton setTitle : @"Next" forState : UIControlStateNormal];
```

```
[nextButton setTitleColor : [UIColor redColor] forState :  
UIControlStateHighlighted];
```

```
[nextButton addTarget : self action : @selector(nextButtonClicked)  
forControlEvents : UIControlEventTouchUpInside];
```

```
[self.view addSubview : nextButton];
```

```
- (void)nextButtonClicked
```

```
{
```

```
ThirdViewController *nextView = [[ThirdViewController alloc]  
initWithNibName : nil bundle : nil];
```

```
[self.view addSubview : nextView.view];
```

```
- (void) backButtonClicked  
{  
    [self.view removeFromSuperview];  
}  
  
- (void) dealloc  
{  
    [label release];  
    [nextButton release];  
    [backButton release];  
    [super dealloc]; ————— deallocated last always.  
                           super class objects.  
}  
@end
```

* Code in ThirdViewController.h :-

```
#import <UIKit/UIKit.h>  
@interface ThirdViewController : UIViewController  
{  
}  
  
@property (nonatomic, retain) UILabel *label;  
—————/  
—————/  
@property (nonatomic, retain) UIButton *exitButton;  
@property (nonatomic, retain) UIButton *nextButton;  
                           back  
-(void) exitButtonClicked;  
-(void) backButtonClicked;  
@end.
```

* Code in ThirdViewController.m :-

```
@synthesize label, exitButton, backButton.
```

* Code in viewDidLoad method :-

```
[self.view setBackgroundColor:[UIColor yellowColor]];
label=[[UILabel alloc] initWithFrame:CGRectMake(40, 30, 200, 50)],
[label setText:@"View3"];
[label setBackgroundColor:[UIColor redColor]];
[label setTextColor:[UIColor blackColor]];
[label setTextAlignment:NSTextAlignmentCenter];
[self.view addSubview:label];
backButton=[UIButton buttonWithType:UIButtonTypeRoundedRect];
```

* Same as previous backbutton code :

```
exitButton=[UIButton buttonWithType:UIButtonTypeRoundedRect];
[exitButton setFrame:CGRectMake(150, 200, 60, 40)];
[exitButton setTitle:@"Exit" forState:UIControlStateNormal];
[exitButton setTitleColor:[UIColor redColor] forState:
 UIControlEventStateHighlighted];
[exitButton addTarget:self action:@selector(exitButtonClicked)
 forControlEvents:UIControlEventTouchUpInside];
[exitButton addTarget:self action:@selector(exitButtonClicked)
 forControlEvents:UIControlEventTouchUpInside];
[self.view addSubview:exitButton];
}

-(void)backButtonClicked
{
    [self.view removeFromSuperview];
}
```

```

-(void)exitButtonClicked
{
    exit(0);
}

```

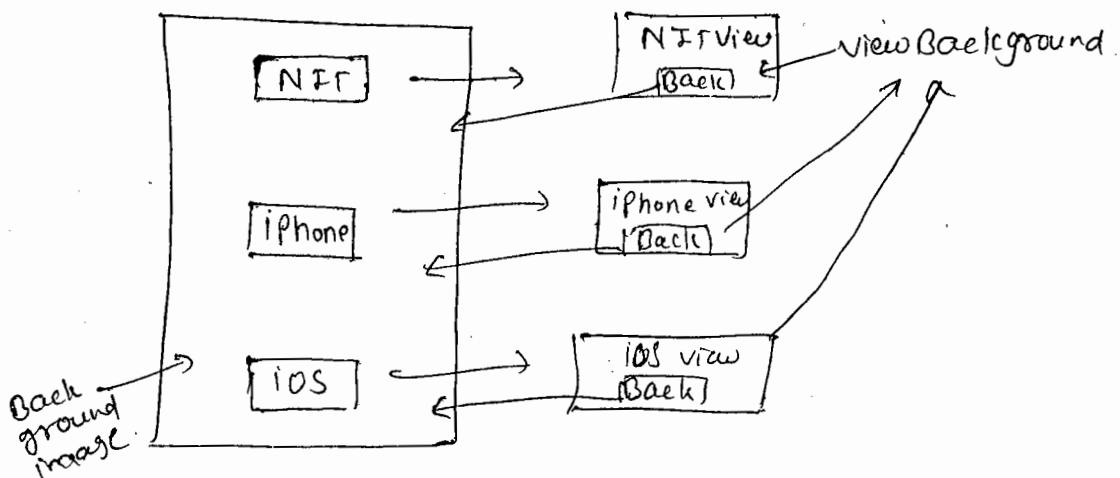
```

-(void)dealloc
{
    [label release];
    [exitButton release];
    [backButton release];
    [super dealloc];
}

```

@end.

— XOX —



View size

iPhone	}	320 x 480
iphone3		
3GS		
4		
4S		

5	}	320 x 568
5C		
5S		

* Create a iOS appl' with the name WindowApp3 of empty appl?

24/10/2013

- ⇒ Create 4 objective C classes with the name HomeViewController, NITViewController, iOSViewController & iPhoneViewController.
- ⇒ Create 4 png images with 320x460 size. with the name iPhone.png, NIT.png, Home.png, iOS.png.
- ⇒ Drag these all 4 png files into project template or add to the project.
- ⇒ Create another 3 button images with 100x100 sizes with the name button1.png, button2.png, button3.png. and drag or add to the current project.

* Code in AppDelegate.h :-

```
#import <UIKit/UIKit.h>
@interface AppDelegate : UIResponder <UIApplicationDelegate>
@property (strong, nonatomic) UIWindow *window;

```

----- " ----- HomeViewController * viewController;

* code in AppDelegate.m :-

```
#import "AppDelegate.h"
#import "HomeViewController"
-(void)dealloc
{
    [_viewController release];
    [super dealloc];
}
```

didFinishLaunchingWithApplication.

{
=====

self.viewController = [[HomeViewController alloc] initWithNibName:
nil bundle: nil];

self.window.rootViewController = self.viewController;

=====

* Code in HomeViewController.h :-

#import <UIKit/UIKit.h>

@interface HomeViewController : UIViewController

{

}

-(void)buttonClicked :(id) sender;

@end.

* Code in HomeViewController.m :-

#import "HomeViewController.h"

--- "NITViewController.h"

--- "iOSViewController.h"

--- "iPhoneViewController.h"

@end.

@implementation HomeViewController

{

=====

}

* Code in viewDidLoad method :-

```
- (void) viewDidLoad
{
    [super viewDidLoad];
    [self.view setFrame:CGRectMake(0, 0, 320, 460)];
    UIImageView *backgroundImage = [[UIImageView alloc] initWithFrame:-
        Image:[UIImage imageNamed:@"home.png"]];
    [backgroundImage setFrame: self.view.bounds];
    [self.view addSubview: backgroundImage];

    UIButton *button1 = [UIButton buttonWithType:UIButtonTypeCustom];
    [button1 setFrame:CGRectMake(100, 50, 80, 50)];
    [button1 setBackgroundImage:[UIImage imageNamed:@"button1.png"]
        forState: UIControlStateNormal];
    [button1 setTag:1];
    [self.view addSubview: button1];
    [button1 addTarget: self action:@selector(buttonClicked:)
        forControlEvents:UIControlEventTouchUpInside];
}

UIButton *button2 = _____, _____ as above
[button2 setFrame:CGRectMake(100, 150, 80, 50)];
_____ as above : @ "button2.png"];
[button2 setTag:2];
[self.view addSubview: button2];
[button2 addTarget: self action:@selector(buttonClicked:)
    forControlEvents:UIControlEventTouchUpInside];
```

```
UIButton *button3 = [UIButton buttonWithType:UIButtonTypeCustom];
                     "                                     as above (100,250,80,50)];
                     "                                     @"button3.png"];
[button3 setTag:3];
[self.view addSubview: button3];
[button3 addTarget:self action:@selector(buttonClicked:) forControlEvents:UIControlEventTouchUpInside];
}

-(void)buttonClicked:(id)sender;
{
    UIButton *bt = (UIButton *)sender;
    if (bt.tag == 1) // if ([bt.currentTitle isEqualToString:@""])
    {
        NSLog(@"button1");
        NITViewController *nitView = [[NITViewController alloc]
                                       initWithNibName:nil bundle:nil];
        [self.view addSubview:nitView.view];
    }
    else if (bt.tag == 2)
    {
        NSLog(@"button2");
        iPhoneViewController *nitView = [[iPhoneViewController alloc]
                                         initWithNibName:nil bundle:nil];
        [self.view addSubview:nitView.view];
    }
    else if (bt.tag == 3)
    {
        NSLog(@"button3");
        iOSViewController *nitView = [[iOSViewController alloc] initWith-
                                         NibName:nil bundle:nil];
        [self.view addSubview:nitView.view];
    }
}
```

* Code in NITViewController.m :-

```
- (void) viewDidLoad {
{
    [super viewDidLoad];
    [self.view setFrame:CGRectMake(0, 0, 320, 460)];
    UIImageView * imageView = [[UIImageView alloc] initWithFrame:
        [UIImage imageNamed:@"NIT.png"]];
    [imageView setFrame: self.view.frame];
    [self.view addSubview: imageView];
    UIButton * button = [UIButton buttonWithType:UIButtonTypeRoundedRect];
    [button setFrame: CGRectMake(40, 50, 80, 40)];
    [self.view addSubview: button];
    [button addTarget: self action: @selector(backButtonClicked)
        forControlEvents: UIControlEventTouchUpInside];
}
-(void) backButtonClicked
{
    [self.view removeFromSuperview];
}
@end.
```

* code in iPhoneViewController.m :-

```
- (void) viewDidLoad {
{
    [self.view setFrame:CGRectMake(0, 0, 320, 460)];
    UIImageView * imageView = [[UIImageView alloc] initWithFrame:
        [UIImage imageNamed:@"iPhone.png"]];
```

```
[imageView setFrame: self.view.frame];
[self.view addSubview: imageView];

UIButton *back = [UIButton buttonWithType: UIButtonRoundedRect];
[back setFrame: CGRectMake(40, 50, 80, 40)];
[self.view addSubview: back];
[back addTarget: self action: @selector(backButtonClicked)
    forControlEvents: UIControlEventTouchUpInside];
}

-(void) backButtonClicked
{
    [self.view removeFromSuperview];
}
@end.
```

* code in iOSViewController.m :-

```
-(void) viewDidLoad
{
    [self.view setFrame: CGRectMake(0, 0, 320, 460)];
    UIImageView *imageView = [[UIImageView alloc] initWithFrame:
        [UIImage imageNamed @"image.png"];
    [imageView setFrame: self.view.frame];
    [self.view addSubview: imageView];

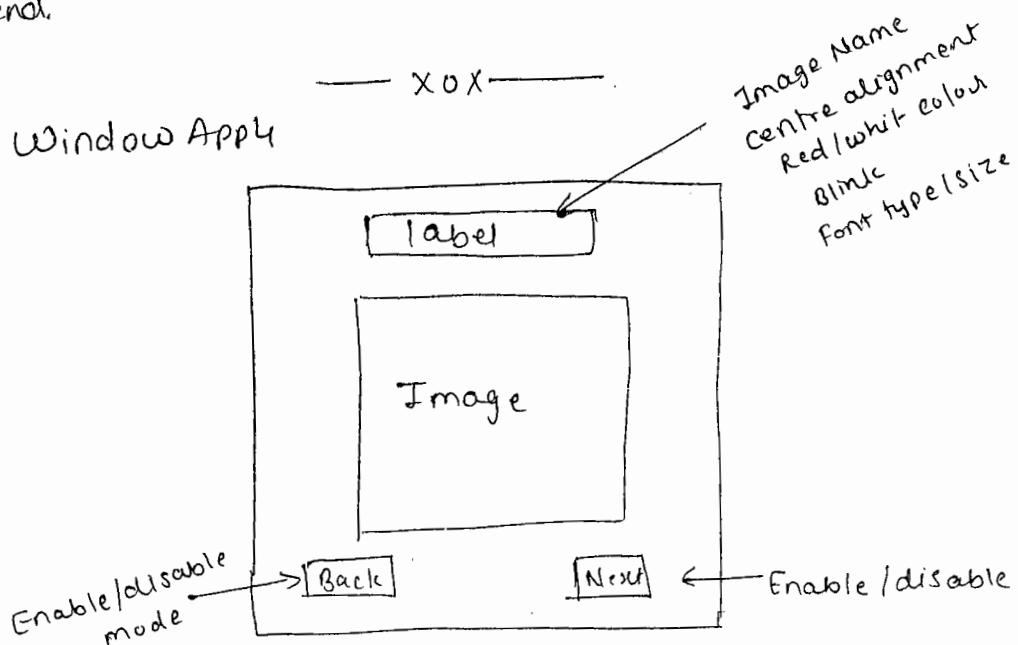
    UIButton *back = [UIButton buttonWithType: UIButtonRoundedRect];
    [back setFrame: CGRectMake(40, 50, 80, 40)];
    [self.view addSubview: back];
    [back addTarget: self action: @selector(backButtonClicked)
        forControlEvents: UIControlEventTouchUpInside];
}
```

```

-(void) backButtonClicked
{
    [self.view removeFromSuperview];
}
@end

```

* Window App4



⇒ Create a empty project with the name windowApp4 for iPad device.

⇒ Drag few images or add some images to the project.

⇒ Create a new objective C class file with the name HomeViewController.

* Code in AppDelegate.h :-

```

#import <UIKit/UIKit.h>
@class HomeViewController;
@interface AppDelegate : UIResponder <UIApplicationDelegate>
@property (strong, nonatomic) UIWindow *window;
@property (strong, nonatomic) HomeViewController *viewController;
@end

```

* Code in AppDelegate.m :-

```
#import "AppDelegate.h"
#import "HomeViewController.h"

@implementation AppDelegate

- (void)dealloc {
    [viewController release];
    [super dealloc];
}

didFinishLaunchingWithOptions:(NSDictionary *)launchOptions {
    self.window = [[[UIWindow alloc] initWithFrame:[UIScreen mainScreen].bounds] autorelease];
    self.viewController = [[HomeViewController alloc] initWithNibName:nil
        bundle:nil];
    self.window.rootViewController = self.viewController;
}
```

* Code in HomeViewController.h :-

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

@interface HomeViewController : UIViewController
{
    NSMutableArray *images;
    UILabel *imageName;
    UIButton *backButton;
    UIButton *nextButton;
    UIImageView *imageView;
}
```

```
@property (nonatomic, retain) NSArray *images;
UITable *imageName;
UIButton *backButton;
UIButton *nextButton;
UIImageView *imageView;

-(void)blink;
-(void)buttonClick:(id)sender;
@end.
```

* Code in HomeViewController.m:-

```
@synthesize images, imageName, imageView, nextButton,
backButton;

short int currentIndex = 0;
bool blinking = NO;

- (id)initWithNibName:(NSString *) nibNameOrNil bundle:
(CNSBundle *) nibBundleOrNil {
    images = [[NSArray alloc] initWithObjects: @"Beach.jpg",
              @"Elephant.jpg", @"Stones.jpg", @"Tahoe.jpg", @"Lion.jpg",
              nil];
}

return self;
}

- (void)viewDidLoad {
    [super viewDidLoad];
    [self.view setBackgroundColor:[UIColor grayColor]];
    imageName = [[UILabel alloc] initWithFrame:CGRectMake(250, 100, 300, 50)];
    [imageName setText:[images objectAtIndex:currentIndex]];
    [imageName setTextAlignment:NSTextAlignmentCenter];
    [imageName setBackgroundColor:[UIColor redColor]];
}
```

```

[imageName setTextColor : [UIColor whiteColor]];
[imageName setFont : [UIFont fontWith Name: @"Courier" size: 25.0]];
[imageName setUserInteractionEnabled: NO];
[self.view addSubview: imageName];
NSTimer *timer = [NSTimer scheduledTimerWithTimeInterval:
(NSTimeInterval) 0.25] target: self
selector: @selector (blink)
userInfo: nil
repeats: TRUE];
imageView = [[UIImageView alloc] initWithFrame: [UIImage
 imageNamed: [images objectAtIndex: currentIndex]]];
[imageView setFrame: CGRectMake(100, 200, 600, 600)];
[imageView setUserInteractionEnabled: FALSE];
[self.view addSubview: imageView];
backButton = [UIButton buttonWithType: UIButtonTypeRoundedRect];
[backButton setFrame: CGRectMake (100, 820, 100, 50)];
[backButton setTitle: @"Back" forState: UIControlStateNormal];
[backButton setTitleColor: [UIColor blackColor] forState:
UIControlStateNormal];
[backButton setTitleColor: [UIColor redColor] forState:
UIControlStateHighlighted];
[backButton.setEnabled: FALSE];
// [backButton setAlpha: 0];
[backButton addTarget: self action: @selector (buttonClick)
forControlEvents: UIControlEventTouchUpInside];
[self.view addSubview: backButton];

```

```
nextButton = [UIButton buttonWithType: UIButtonTypeRoundedRect];
[nextButton setFrame: CGRectMake(600, 820, 100, 50)];
[nextButton setTitle: @"Next" forState: UIControlStateNormal];
[nextButton setTitleColor: [UIColor blackColor] forState:
    UIControlStateNormal];
[nextButton setTitleColor: [UIColor redColor] forState:
    UIControlStateHighlighted];
[nextButton.setEnabled: TRUE];
// [nextButton setAlpha: 0];
[nextButton addTarget: self action: @selector(buttonClick:)
    forControlEvents: UIControlEventTouchUpInside];
[self.view addSubview: nextButton];
}

-(void)buttonClick: (id)sender
{
    UIButton *bt = (UIButton *)sender;
    if ([bt.currentTitle isEqualToString: @"Next"])
    {
        currentIndex++;
        [imageView setImage: [UIImage imageNamed: [images
            objectAtIndex: currentIndex]]];
        [imageName setText: [image objectAtIndex: currentIndex]];
        if (backButton.isEnabled == FALSE)
        if ([backButton isEnabled] == FALSE)
            [backButton.setEnabled: TRUE];
    }
}
```

```
if (currentIndex == [images count] - 1)
    [nextButton setEnabled: FALSE];
}

else if ([bt currentTitle] isEqualToString: @"Back")]
{
    --currentIndex;
    [imageView setImage: [UIImage imageNamed: [images
        objectAtIndex: currentIndex]]];
    [imageName setText: [images objectAtIndex: currentIndex]];
    if ([nextButton isEnabled] == FALSE)
        [nextButton setEnabled: TRUE];
    if (currentIndex == 0)
        [backButton setEnabled: FALSE];
}
}

-(void)blink
{
    if (blinking == NO)
    {
        // imageName alpha = 0;
        // [imageView setAlpha: 0];
        [imageName setBackgroundColor: [UIColor whiteColor]];
        [imageName setTextColor: [UIColor redColor]];
        blinking = YES;
    }
    else
    {
        // imageName. alpha = 1;
        // [imageView setAlpha: 1];
    }
}
```

```

    [imageName setBackgroundColor:[UIColor whiteColorled]];
    [imageName setTextColor:[UIColor whitecolor]];
    blinking = NO;
}
- (void) dealloc
{
    [images release];
    [imageName release];
    [imageView release];
    [nextButton release];
    [backButton release];
    [super dealloc];
}
end
}

```

—XOX—

28/10/2013

- * Create a single view iOS applⁿ with the name WindowApp5.
- ⇒ open ViewController.xib & place following objects.
 - Label object drag to nib file & change the attributes.
 - Make the alignment as center & font size as 20.
 - Drag the RoundRect button object to the nib file.
 - set the tag as 1 & make type as custom & remove button title.
 - Drag a image to the project with the name left.png & make the size as 23x20 dimension.
 - Select the button object & change the image as left.png.
 - Drag another roundRect button & change the attributes
 - type as custom
 - tag is 2.
 - and select the image right.png.

- Drag sight.png to the project then only it is visible.
- Place the imageView object on nib file. and change alignment properly.
- Go to ViewController.h .

* Code in Viewcontroller.h :-

```
#import <UIKit/UIKit.h>
@interface ViewController : UIViewController
{
    NSMutableArray * imagelist ;
    NSTimer * timer1 ;
    NSTimer * timer2 ;
    @property (nonatomic,retain) NSMutableArray * imagelist ;
    NSTimer * timer2,* timer2 ;
    IBOutlet UILabel * label ;
    IBOutlet UIButton * back ;
    IBOutlet UIButton * next ;
    IBOutlet UIImageView * imageview ;
    -(IBAction) buttonClick : (id)sender ;
    -(void) blink ;
    -(void) hidebutton ;
}
@end
```

→ Open ViewController.nib & make the relation b/w
IBOutlets objects & UI Objects.

- Drag from files owner to label & select label outlet.
(we need to press right mouse button until making the relation).

- Drag file's owner to button & select back outlet.
 - Drag file's owner to another button object & select 'next' outlet.
 - Drag from file's owner to imageView & select imageView object.

* Actions :-

→ To make the actions for the button we need to choose following procedures.

- Drag buttonUI to First Responder & select button.Click method
 - Same procedure for second button also.

* code in viewcontroller.m :-

```
@synthesize label, back, next, imagelist, imageview, timer1,
timer2;

short int cindex;
bool isBlink = NO;
bool isHide = NO;

-(void) viewDidLoad
{
    [super viewDidLoad];
    [self.view setBackgroundColor:[UIColor blackColor]];
    imagelist = [[[NSMutableArray alloc] init] autorelease];
    for (int i=1; i<=10; i++) {
        [imagelist addObject:[NSString stringWithFormat:@"%@image%@\.%@", @"image", i, @"jpg", i]];
    }
}
```

```

[label setBackgroundColor:[UIColor whiteColor]];
[label setTextColor:[UIColor redColor]];
[label setText:[imageList objectAtIndex:cindex]];
NSTimer *timer1=[NSTimer scheduledTimerWithTimeInterval:(NSTimeInterval)
(0.25) target:self selector:@selector(blink) userInfo:nil
repeats:TRUE];
}

-(IBAction)buttonClick:(id)sender
{
UIButton *bt=(UIButton*)sender;
if(bt.tag==1)
{
--cindex;
[imageView setImage:[UIImage imageNamed:[imageList
objectAtIndex:cindex]]];
[label setText:[imageList objectAtIndex:cindex]];
[label setText:[imageList objectAtIndex:cindex]];
if([next isEnabled]==FALSE)
{
[next setEnabled:YES];
[next setAlpha:1];
}
if(cindex==0)
{
[back setEnabled:FALSE];
[back setAlpha:0];
}
}
}

```

```
        }
    }
    +cinder;
    [imageView setImage: [UIImage imageNamed : [imagedList
        objectAtIndexIndex : cindex]]];
    [label setText: [imagedList objectAtIndexIndex : cindex]];
    if ([back isEnabled == FALSE])
    {
        [back setEnabled: TRUE];
        [back setAlpha: 1];
    }
    if ([cindex == [imagedList count] - 1])
    {
        [next setEnabled: FALSE];
        [next setAlpha: 0];
    }
}
}

-(void) blink
{
    if (isBlink == NO)
    {
        [label setBackgroundColor: [UIColor redColor]];
        [label setTextColor: [UIColor redColor]];
        isBlink = YES;
    }
    else
    {
        [label setBackgroundColor: [UIColor whiteColor]];
        [label setTextColor: [UIColor redColor]];
        isBlink = NO;
    }
}
```

```
- (void) hideButtons  
{  
    if (isHide == NO)  
    {  
        [back setAlpha: 0];  
        [next setAlpha: 0];  
        isHide = YES;  
        [timer2 invalidate]; // stop timer but doesn't kill the object.  
    }  
}
```

```
- (void) touchesBegin: (NSSet *) touches withEvent: (UIEvent *) event  
{  
    NSTimer *time = [NSTimer scheduledTimerWithTimeInterval:  
        (NSTimeInterval) 0.3 target: self selector: @selector  
        (hideButtons) userInfo: nil repeats: TRUE];  
    timer2 = time;  
    isHide = NO;  
    if ([back isEnabled])  
        [back setAlpha: 1];  
    if ([next isEnabled])  
        [next setAlpha: 1];  
}
```

```
- (void) dealloc  
{  
    [imagerlist release];  
    [timer1 release];  
    [timer2 release];  
    [super dealloc];  
}
```

29/10/2013

* Create a single view iOS app with the name Login App.

• Open ViewController.xib & place following controllers.

1) Label :- Text → user size → 15

Text Alignment → left.

2) Label :- Text → Password

Text Alignment → left., size → 15.

3) UITextField :- Placeholder → Username.

4) UITextField :- Placeholder → Password , secure enable

5) UIButton :- Title → login.

6) Activity Indicator View.

* Code in ViewController.h :-

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

```
@interface ViewController: UIViewController <UITextFieldDelegate>
{
}
```

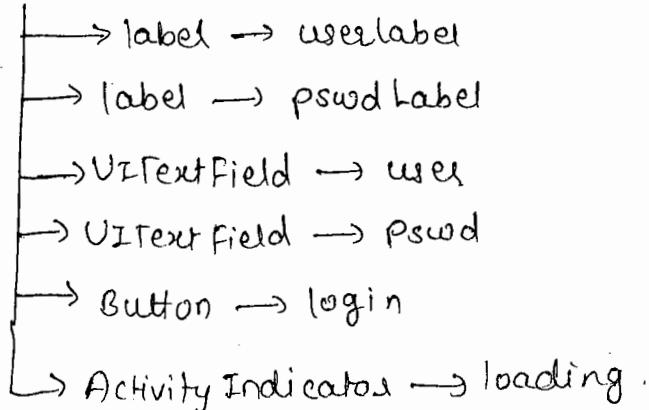
```
@property (nonatomic, retain) IBOutlet UILabel *userLabel;
@IBOutlet UILabel *pswdLabel;
@IBOutlet UITextField *user;
@IBOutlet UITextField *pswd;
@IBOutlet UIButton *login;
@IBOutlet UIActivityIndicatorView *loading;
```

```
- (IBAction) loginButtonClicked: (id) sender
```

```
@end
```

• Make the relation betn IBOutlets & File's owner.

file's Owner



- Make the buttonAction to login button & select loginButtonClicked.

Button → firstResponder → loginButtonClicked.

* Code in Viewcontroller.m :-

```
@synthesize userLabel, pswdLabel;  
—————— user, pswd;  
—————— login, loading;
```

pragma -mark UITextFielddelegate method.

```
- (BOOL) textFieldShouldReturn : (UITextField *) textField  
{  
    [textField resignFirstResponder];  
    return YES;  
}  
  
- (BOOL) textFieldShouldBeginEditing : (UITextField *) textField  
{  
    if (textField.tag == 1)  
        [userLabel setTextColor: [UIColor blackColor]];  
    else if (textField.tag == 2)  
        [pswdLabel setTextColor: [UIColor blackColor]];
```

```
[textField setReturnKeyType: UIReturnKeyDone];
return YES;
}

# pragma -mark ViewControllerMethods
- (void) viewDidLoad
{
    [super viewDidLoad];
    [user setDelegate: self];
    [user setTag: 1];
    [pswd setTag: 2];
    [pswd setDelegate: self];
    [pswd setTag: 2];
}

- (IBAction) loginButtonClicked:(id)sender
{
    NSString * uname = @ "Balu";
    NSString * pname = @ "1234";
    UIAlertView * alert = nil;
    [loading startAnimating];
    if ([user text] isEqualToString: @ "" )
    {
        alert = [[UIAlertView alloc] initWithTitle: @ "Login Alert"
                                           message: @ "User is Empty" delegate: nil
                                           cancelButtonTitle: @ "OK" otherButtonTitles: nil, nil];
        [alert show];
        [userLabel setTextColor: [UIColor redColor]];
        [loading stopAnimating];
    }
}
```

```
else if ([Pswd Text] isEqualToString: @"J")
{
    alert = [[UIAlertView alloc] initWithTitle: @"Login Alert"
                                         message: @"Password is Empty" delegate: nil
                                         cancelButtonTitle: @"OK" otherButtonTitles: nil, nil];
    [alert show];
    [PswdLabel setTextColor: [UIColor redColor]];
    [loading stopAnimating];
}

else if ([User Text] caseInsensitiveCompare: uname] == NSOrderedSame)
{
    alert = [[UIAlertView alloc] initWithTitle: @"Login Alert"
                                         message: @"invalid user" delegate: nil cancelButtonTitle: @"OK" otherButtonTitles: nil, nil];
    [alert show];
    [UserLabel setTextColor: [UIColor redColor]];
    [User setText: @""];
    [loading stopAnimating];
}

else if ([Pswd Text] isEqualToString: pname] == FALSE)
{
    alert = [[UIAlertView alloc] initWithTitle: @"Login Alert"
                                         message: @"Invalid Password" delegate: nil
                                         cancelButtonTitle: @"OK" otherButtonTitles: nil, nil];
    [alert show];
}
```

```
[pswdLabel setTextColor: [UIColor redColor]];
[pswd setText: @""];
[loading stopAnimating];
}
```

else

```
{
```

```
//Sleep(3);
[user setAlpha: 0];
[pswd setAlpha: 0];
[userLabel setAlpha: 0];
[pswdLabel setAlpha: 0];
[loading setAlpha: 0];
[login setAlpha: 0];
```

```
UILabel *label = [[UILabel alloc] initWithFrame:(CGRectMake  
(20, 50, 100, 44))];
```

```
[label setText: @"welcome"];
[self.view addSubview: label];
```

```
}
```

```
[label release];
```

@end.

30/10/2013

No written code

UIAlertView :-

- By using this controller we can pass alertviews in device.
- When we are working with UIAlertView, it cannot be customized. i.e. default view only visible.
- If we required to handle UIAlertView button actions then we required to implement UIAlertViewDelegate methods.
- Generally these all methods are optional i.e. if we required to perform the task then only recommended to go for delegate methods.

UISwitch :-

- By using this control we can place switch.
- UISwitch having two options i.e. on or OFF.
- When we are working with UISwitch control then we need to handle UIControlEventValueChanged action.
(normally for buttons UIControlEventTouchUpInside).

UIStepper :-

- By using this controller we can implement drop down list behaviour in iOS.
- Generally this controller is recommended to use when we required to place some quantity value.

- UIStepper having three major properties i.e. minimum value, maximum value & step value.
- Minimum value will provide minimum stepper data, maximum value will provide maximum stepper data & Step value provides difference b/w previous value & new value.
- When we are working UIStepper if we required to access stepper current value then go for value property.
- On UIStepper we required to handle UIControlEvent-Valuechanged action.

UISlider :-

- By using this controller we can make volume control view
- When we are working with UISlider then we need to set minimum, maximum values.
- When we are working with UISlider, we required to handle UIControlEvent ValueChange Action.

* Create a singleview appl' with the name windowApp6.

* Code in ViewController.h :-

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

```
@interface ViewController : UIViewController <UIAlertViewDelegate>
{
    UIAlertView *alert;
    UIButton *button;
    UISwitch *switchView;
    UIStepper *stepperView;
    UILabel *label;
}
```

```
- (void) buttonClicked;  
-(void) switchClicked;  
-(void) stepperClicked;  
-(void) sliderClicked : (UISlider*) slider;  
    //for local objects need to pass arguments.  
@end.
```

* Code ViewController.m:-

```
- (void) viewDidLoad {  
    [super viewDidLoad];  
    button = [UIButton buttonWithType: UIButtonRoundedRect];  
    [button setFrame: CGRectMake(30, 50, 150, 80)];  
    [button addTarget: self action: @selector(buttonClicked)  
        forControlEvents: UIControlEventTouchUpInside];  
    [button setTitle: @"click" forState: UIControlStateNormal];  
    [self.view addSubview: button];  
  
    switchView = [[UISwitch alloc] initWithFrame: CGRectMake  
        (30, 150, 150, 50)];  
    [switchView setEnabled: YES];  
    [switchView setOn: YES];  
    [switchView addTarget: self action: @selector(switchClicked)  
        forControlEvents: UIControlEventValueChanged];  
    [self.view addSubview: switchView];
```

```
StepperView = [[UIStepper alloc] initWithFrame: CGRectMake(30, 200, 100, 50)];  
[StepperView setMinimumValue: 0];  
[StepperView setMaximumValue: 10];  
[StepperView setValue: 2];  
[StepperView addTarget: self action: @selector(stepperClicked)  
forControlEvents: UIControlEventValueChanged];  
[self.view addSubview: stepperView];  
  
label = [[UILabel alloc] initWithFrame: CGRectMake(150, 190, 200, 60)];  
[self.view addSubview: label];  
[label setText: @" "];  
  
.UISlider *slider = [[UISlider alloc] initWithFrame:  
CGRectMake(30, 300, 100, 50)];  
CGAffineTransform transform = CGAffineTransformMakeRotation  
(M_PI * 0.5);  
slider.transform = transform;  
[slider setMinimumValue: 0];  
[slider setMaximumValue: 100];  
[slider addTarget: self action: @selector(sliderClicked:  
forControlEvents: UIControlEventValueChanged)];  
[self.view addSubview: slider];  
}
```

```
- (void) sliderClicked: (UISlider*) slider
{
    NSLog(@"%@", [slider value]);
    [label setText: [NSString stringWithFormat: @"Slider value: %i",
        [slider value]]];
}

-(void) stepperClicked
{
    NSLog(@"%@", [stepperView value]);
    [label setText: [NSString stringWithFormat: @"stepper value:
        %i", [stepperView value]]];
}

-(void) switchClicked
{
    NSLog(@"Inside Function");
    if ([switchView isOn] == FALSE)
    {
        [button setHidden: YES];
        [self.view setBackgroundColor: [UIColor redColor]];
    }
    else
    {
        [button setHidden: NO];
        [self.view setBackgroundColor: [UIColor whiteColor]];
    }
}

-(void) buttonClicked
{
    alert = [[UIAlertView alloc] initWithTitle: @"Alert View"
        message: @"Select Option" delegate: self
        cancelButtonTitle: @"Cancel" otherButtonTitles: @"OK",
        @"Continue", nil];
}
```

```
[alert show];
}

-(void)alertView : (UIAlertView *)alertView clickedButtonAt-
Index (NSInteger)buttonIndex
{
    if (buttonIndex == 0)
    {
        NSLog(@"it is cancel button");
        [button setTitle: @"cancel" forState: UIControlStateNormal];
    }
    else if (buttonIndex == 1)
    {
        NSLog(@"it is ok button");
        [button setTitle: @"OK" forState: UIControlStateNormal];
    }
    else (buttonIndex != 2)
    {
        NSLog(@"it is Continue button");
        [button setTitle: @"continue" forState: UIControlState-
Normal];
    }
}

-(void)dealloc
{
    [alert release];
    [button release];
    [switchView release];
    [StepperView release];
    // [SliderView release];
    [label release];
    [super dealloc];
}

@end.
```

02/11/2013

* UITextField :-

- By using this view we can place single line text content.
- When we are working UITextField we require to implement UITextFieldDelegate methods.
- When we are editing UITextField data then automatically keyboard will appear.
- Generally UITextField having following properties.
 - 1) BorderStyle
 - 2) AutoCorrectionType.
 - 3) KeyboardType.
 - 4) KeyboardAppearance
 - 5) ReturnKeyType.
 - 6) Delegate.
 - 7) Enabled.
 - 8) Secure / SecurityTextEntry.
- textFieldShouldReturn method will call when we click on return button of keyboard.
- textFieldShouldBeginEditing method will call when editing is started.
- textFieldShouldEndEditing is called when editing is finished.
- To height the keyboard on any textField then we need to call resignFirstResponder method.
- When we require to focus on any one of the textField then recommended to go to becomeFirstResponder method.
- When we required to create non editable text field we doesn't having any properties so, we required to hide the keyboard when user touches the specific textField.(in this case we can disable userInteraction also)

UITextView :-

- When we required to enter multiline content then recommended to use UITextView.
- UITextView doesn't having borderstyle property. So we require to make customized border.
- When we are working with UITextView then we require to handle UITextViewDelegate methods.
- When we are working with UITextView then return button doesn't having default action. So programmatically we required to handle return button action.

Example:-

* Create a single view application with name windowApp.

- import QuartzCore Framework to current project
(Goto Build phase → Link Binary With Libraries → '+')

* Code in ViewController.h :-

```
#import <UIKit/UIKit.h>
@interface ViewController : UIViewController <UITextFieldDelegate>
<UITextViewDelegate>
{
    UITextField *one;
    UITextField *two;
    UITextField *three;
    UITextView *text;
    CGSize viewSize;
}
@property (nonatomic, retain) UITextField *one;
@property (nonatomic, retain) UITextView *text;
@end
```

* code in ViewController.m :-

```
#import "ViewController.h"
#import <QuartzCore/QuartzCore.h>

@interface ViewController : UIViewController

@end

@implementation ViewController
@synthesize one,two;
-(void) viewDidLoad
{
    one=[[UITextField alloc] init];
    two=[[UITextField alloc] init];
    three=[[UITextField alloc] init];
    text=[[UITextView alloc] initWithFrame:CGRectMake(20,
    190, 270, 150)];
}

-(void) viewDidAppear:(BOOL)animated
{
    [super viewDidAppear:animated];
    [one setFrame:CGRectMake(20, 20, 180, 35)];
    [one setBorderStyle:UITextBorderStyleRoundedRect];
    [one setAutoCorrectionType:UITextAutoCorrectionTypeNo];
    [one setKeyboardType:UITextKeyboardTypeAlphabet];
    [one setKeyboardAppearance:UITextKeyboardAppearanceDefault];
    [one setReturnKeyType:UITextReturnKeyTypeDone];
    [one setDelegate:self];
    [one setEnabled:YES];
}
```

```
C // [one becomeFirstResponder];
C // [one setSecureTextEntry:YES]; password.
    [self.view addSubview:one];

    [two setFrame:CGRectMake(20, 75, 150, 35)];
    [two setBorderStyle:UITextBorderStyleLine];
    [two setAutocorrectionType:UITextAutocorrectionTypeYES];
    [two setKeyboardType:UIKeyboardTypeNamePhonePad];
    [two setKeyboardAppearance:UIKeyboardAppearanceAlert];
    [two setReturnKeyType:UIReturnKeyGo];
    [two setDelegate:self];
    [two setEnabled:YES];
    // [two setAllowEditingTextAttributes:FALSE];
    // [two setSecureTextEntry:YES];
    [self.view addSubview:two];

    [three setFrame:CGRectMake(20, 130, 150, 35)];
    [three setBackgroundColor:[UIColor redColor]];
    [three setBorderStyle:UITextBorderStyleRoundedRect];
    [three setDelegate:self];
    [three setText:@"Welcome"];
    // [three setUserInteractionEnabled:FALSE];
    [self.view addSubview:three];

    [text setShowVerticalScrollIndicator:YES];
    [text setTextColor:[UIColor redColor]];
    [text setAutoCapitalizationType:UITextAutoCapitalizationTypeWord];
```

```
[text setDelegate: self];
[text setKeyboardType : UIKeyboardTypeDefault];
// [text setReturnKeyType : UIReturnKeyDone];
[self.view addSubview: text];

CALayer* borderLayer = [CALayer layer];
CGRect borderFrame2 = CGRectMake(0,0, text.frame.size.width,
                                text.frame.size.height);

[borderLayer setBackgroundColor:[[UIColor clearColor] CGColor]];
[borderLayer setFrame: borderFrame2];
[borderLayer setMasksToBounds: YES];
[borderLayer setNeedsDisplayOnBoundsChange: YES];
[borderLayer setCornerRadius: 8.0];
[borderLayer setBorderWidth: 2];
[borderLayer setBorderColor: [[UIColor redColor] CGColor]];
[text.layer addSublayer: borderLayer];
}

-(void) touchesBegan :(NSSet *) touches withEvent :(UIEvent *) event
{
    [self.view endEditing: YES];
}

-(void) dealloc
{
    [one release];
    [two release];
    [super dealloc]; [three release]; [text release];
}
```

pragma - mark UITextField Delegates.

- (BOOL) textFieldShouldReturn : (UITextField *) textField

{

[textField resignFirstResponder];
return TRUE;

}

- (BOOL) textFieldShouldBeginEditing : (UITextField *) textField

{

if (textField == three)

{

[three resignFirstResponder];
return FALSE;

}

return TRUE;

}

- (BOOL) textFieldShouldEndEditing : (UITextField *) textField

{

if (textField == ^{End}three)

{

[three resignFirstResponder]; [one setText: [one text]];
return FALSE; [one resignFirstResponder];

}

return TRUE;

}

return TRUE;

- (BOOL) textField

pragma - mark UITextView Delegates.

- (BOOL) textViewShouldBeginEditing : (UITextView *) textView

{

viewSize = self.view.frame;
(self.view setFrame: CGRectMake (viewSize.origin.x,
viewSize.origin.y - 150, viewSize.size.width,
viewSize.size.height));
return TRUE;

```

-(BOOL) textViewShouldEndEditing:(UITextView *)textView
{
    [self.view setFrame:viewSize];
    return TRUE;
}

-(BOOL) textView:(UITextView *)textView shouldChangeTextInRange:(NSRange)range replacementText:(NSString *)text
{
    if ([text isEqualToString:@"\n"])
    {
        [textView resignFirstResponder];
        return NO;
    }
    return YES;
}

```

②

04/11/2013

* UITabBarController :-

- It is a subclass of UIViewController.
- By using UITabBarController we can create Tab bar appl'.
- In UITabBarController all views are placed on UITabBar.
- For every view UITabBar contains a button called TabBarItem.
- When we click on any TabBarItem then corresponding view will appear on screen.
- When we are working with UITabController then recommended to place 3 or 4 TabBarItems only.

- When we are working with UITabBarItem then, that item image size should be 30x30 pixels only; for retina image we can place 60x60 pixels.

Example:-

* Create an empty app^ with the name UITabBar App.

* Code in AppDelegate.h:-

```
#import <UIKit/UIKit.h>
@interface AppDelegate : UIResponder <UIApplicationDelegate>
<UITabBarControllerDelegate>
@property (strong, nonatomic) UIWindow *window;
@ _____" _____ UITabBarController * tabBarView -
Controller;
@end
```

* ~~code~~ + Create three viewController classes with the name ClockViewController, Timer ViewController, STwatchViewController.

* Code in AppDelegate.m:-

```
#import "AppDelegate.h"
#import "ClockViewController.h"
#import "TimerViewController.h"
#import "STwatchViewController.h".
```

@ implementation AppDelegate

```
- (void)dealloc
{
    [_tabBarView Controller release];
}
```

* Code in didFinishLaunchingWithOptions:-

{

self.window.backgroundColor = [UIColor whiteColor];

NSMutableArray * viewList = [[[NSMutableArray alloc] init
autorelease];

ClockViewController * viewController1 = [[ClockViewController alloc]
initWithNibName: nil bundle: nil];

[viewList addObject: viewController1];

TimerViewController * viewController2 = [[TimerViewController
alloc] initWithNibName: nil bundle: nil];

[viewList addObject: viewController2];

STWatchViewController * viewController3 = [[STWatchViewController
alloc] initWithNibName: nil bundle: nil];

[viewList addObject: viewController3];

self.tabBarController = [[UITabBarController alloc] init];

self.tabBarController.viewControllers = viewList;

self.window.rootViewController = self.tabBarController;

[self.window makeKeyAndVisible];

return YES;

}

Place three png files into the project with the name

clock.png, Timer.png, Stop.png. (For retina images
'clock@2x', 'Timer@2x', 'Stop@2x').

* Code in ClockViewController.h :-

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

@interface ClockViewController : UIViewController
{
    UILabel * timelabel;
}

@property (nonatomic, retain) UILabel * timelabel;

-(void) updateTime;

@end.
```

* Code in ClockViewController.m :-

```
@synthesize timelabel;

* code in initWithNibName:-
{
    self.title = NSLocalizedString(@"Clock", @"Clock");
    self.tabBarItem.image = [[UIImage imageNamed:@"clock"]];

    [self.view setBackgroundColor:[UIColor redColor]];
}

return self;
}

-(void) viewDidLoad
{
    timelabel = [[UILabel alloc] initWithFrame:CGRectMake(30, 100, 200, 50)];
    [timelabel setBackgroundColor:[UIColor blackColor]];
    [timelabel setTextColor:[UIColor whiteColor]];
    [timelabel setTextAlignment:NSTextAlignmentCenter];
    [self.view addSubview:timelabel];
    [self updateTime];
}
```

```
- (void) updateTime {
    NSDateFormatter *dateFormatter = ([NSDateFormatter alloc] init);
    [dateFormatter setDateFormatter: @"dd/MMM/yyyy hh:mm:ss"];
    // [dateFormatter setDateFormatter: @"eee/MMM/yy hh:mm:ss"];
    [timeLabel setText: [dateFormatter stringWithFormat: [NSDate date]
                                                fromDate: nil]];
    [self performSelector:@selector(updateTime) withObject:self
        afterDelay: 1.0];
}
```

* Code in STWatchViewController.h :-

```
@ interface STWatchViewController : UIViewController
```

```
{
```

```
    UILabel *label;
```

```
    UIButton *start;
```

```
    UIButton *pause;
```

```
    UIButton *resume;
```

```
    NSTimer *time;
```

```
    NSDate *pauseStart, *previousDate;
```

```
}
```

```
- (void) buttonClicked: (id)sender;
```

```
- (void) countUp;
```

```
@ end.
```

* code STWatchViewcontroller.m:-

code in init with NibName:-

```
{  
    self.title = NSLocalizedString(@"Stop", @"Stop");  
    self.tabBarItem.image = [[UIImage imageNamed:@"stop.png"]  
    [self.view setBackgroundColor:[UIColor blueColor]];  
}  
return self;  
};
```

- (void) viewDidLoad.

```
{  
    label = [[UILabel alloc] initWithFrame:CGRectMake(30, 80, 200,  
                                                    60)];  
    [label setBackgroundColor:[UIColor whiteColor]];  
    [label setTextColor:[UIColor blackColor]];  
    [label setText:[NSString stringWithFormat:@"00:00:00:00"]];  
    [label set.TextAlignment:center];  
  
    [self.view addSubview:label];  
  
    start = [UIButton buttonWithType:UIButtonTypeRoundedRect];  
    [start setFrame:CGRectMake(80, 150, 100, 50)];  
    [start setTitle:@"Start" forState:UIControlStateNormal];  
    [start addTarget:self action:@selector(buttonClicked:)  
        forControlEvents:UIControlEventTouchUpInside];  
  
    [self.view addSubview:start];
```

```

pause = [UIButton buttonWithType: UIButtonTypeRoundedRect];
[pause setFrame: CGRectMake(80, 220, 100, 50)];
[pause setTitle:@"Pause" forState: UIControlStateNormal];
[pause addTarget: self action: @selector(buttonClicked:)];
forControlEvents: UIControlEventTouchUpInside];
[pause setEnabled: NO];
[self.view addSubview: pause];

resume = [ _____, as above ];
[resume setFrame: CGRectMake(80, 290, 100, 50)];
[resume setTitle:@"Resume" forState: UIControlStateNormal];
[resume addTarget: self action: @selector(buttonClicked:)];
forControlEvents: UIControlEventTouchUpInside];
[resume setEnabled: NO];
[self.view addSubview: resume];
}

short int hh, mm, ss, ms;
-(void) buttonClicked : (id) sender
{
    UIButton * bt = (UIButton *) sender;
    if ([[bt currentTitle] isEqualToString: @"Start"])
    {
        [start setEnabled: FALSE];
        [pause setEnabled: YES];
        if (timer == nil)
            timer = [NSTimer scheduledTimerWithTimeInterval: 0.01
                target: self selector: @selector(countup) userInfo: nil
                repeats: YES];
    }
}

```

```
else if ([[bt.currentTitle] isEqualToString:@"Pause"])
{
```

```
[Pause setEnabled: FALSE]
```

```
[Start setTitle: @"Stop" forState: UIControlStateNormal];
```

```
[Start setEnabled: TRUE];
```

```
[Resume setEnabled: TRUE];
```

```
pauseStart = [[NSDate dateWithTimeIntervalSinceNow: 0] retain];
```

```
previousFireDate = [[Time fireDate] retain];
```

```
[Timer setFireDate: [NSDate distantFuture]],
```

```
}
```

```
else if ([[bt.currentTitle] isEqualToString:@"Resume"]).
```

```
{
```

```
[resume setEnabled: YES];
```

```
[Pause setEnabled: YES];
```

```
float pauseTime = -1 * [pauseStart timeIntervalSinceNow];
```

```
[Timer setFireDate: [NSDate dateWithTimeInterval: pauseTime  
sinceDate: previousFireDate]];
```

```
[pauseStart release];
```

```
[previousFireDate release];
```

```
}
```

```
else if ([[bt.currentTitle] isEqualToString:@"Stop"]).
```

```
{ [Start setTitle: @"Start" forState: UIControlStateNormal];
```

```
[Pause setEnabled: FALSE];
```

```
[Timer invalidate];
```

```
time1 = nil;
```

```
hh = mm = ss = ms = 0;
```

```
[label setText:[NSString stringWithFormat:@"00:00:00,00"]];
```

```
}
```

```
}
```

- (Void) countup.

```
{  
    ms;  
    if (ms == 100)  
    {  
        ss;  
        ms = 0;  
    }  
    if (ss == 60)  
    {  
        mm;  
        ss = 0;  
    }  
    if (mm == 60)  
    {  
        hh;  
        mm = 0;  
    }  
}
```

[label setText:[NSString stringWithFormat:@"%02i : %02i : %02i : %02i", hh, mm, ss, ms]];

}

05/11/2013

* Code in TimerViewController.h :-

@ Interface TimerViewController : UIViewController < UIAlertViewDelegate>

```
{  
    UIStepper *hstepper;  
    UIStepper *mstepper;  
    UIStepper *sstepper;  
    UILabel *hlabel;  
    UILabel *mlabel;  
    UILabel *slabel;
```

```
UIButton * start;
UIButton * reset;
NSTimer * timer;
{
-(void)buttonClicked:(id)sender
-(void)countDown;
-(void)stepperValueUpdate:(id)sender;
@end.
```

* Code in TimerViewController.m:-

Code in initWithNibName:-

```
{
    self.title = NSLocalizedString(@"Timer", @"Timer");
    self.tabBarItem.image = [UIImage imageNamed:@"Timer.png"];
    [self.view setBackgroundColor:[UIColor grayColor]];
}
return self;
}

-(void)viewDidLoad{
{
    hStepper = [[UIStepper alloc] initWithFrame:CGRectMake(50, 100,
    50, 50)];
    [hStepper setMinimumValue:0];
    [hStepper setMaximumValue:23];
    [hStepper setStepValue:1];
    [hStepper addTarget:self action:@selector(stepperValueUpdate:)];
    forControlEvents:UIControlEventValueChanged];
}
```

```
hlabel = [[UILabel alloc] initWithFrame: CGRectMake(30, 0, 33, 28)];  
[hlabel setBackgroundColor:[UIColor clearColor]];  
[hlabel setText:@"00"];  
[hlabel setTextAlignment:NSTextAlignmentCenter];  
[hstepper addSubview:hlabel];
```

```
mstepper = [[UIStepper alloc] initWithFrame:CGRectMake(30, 180,  
50, 50)];  
[mstepper setMinimumValue:0];  
[mstepper setMaximumValue:59];  
[mstepper setStepValue:1];  
[mstepper addTarget:self action:@selector(stepperValueUpdate:)  
forControlEvents:UIControlEventValueChanged];
```

```
mlabel = [[UILabel alloc] initWithFrame:CGRectMake(30, 0, 33, 28)];  
[mlabel setBackgroundColor:[UIColor clearColor]];  
[mlabel setText:@"00"];  
[mlabel setTextAlignment:NSTextAlignmentCenter];  
[mstepper addSubview:mlabel];
```

```
sstepper = [[UIStepper alloc] initWithFrame:CGRectMake(10, 240, 50  
50)];  
[sstepper setMinimumValue:0];  
[sstepper setMaximumValue:59];  
[sstepper setStepValue:1];  
[sstepper addTarget:self action:@selector(stepperValueUpdate:)  
forControlEvents:UIControlEventValueChanged];
```

label = [[UILabel alloc] initWithFrame:CGRectMake(30, 0, 33, 28)];

```

[label setBackgroundColor: [UIColor clearColor]];
[label setText:@"00"];
[label setTextColor:[UIColor blackColor]];
[label setFont:[UIFont boldSystemFontOfSize:16]];
[label setTextColor:[UIColor blackColor]];
[label setFont:[UIFont boldSystemFontOfSize:16]];
[label setTextColor:[UIColor blackColor]];
[label setFont:[UIFont boldSystemFontOfSize:16]];

[self.view addSubview:label];
[self.view addSubview:hstepper];
[self.view addSubview:mstepper];
[self.view addSubview:sstepper];

start = [UIButton buttonWithType:UIButtonTypeRoundedRect];
[start setFrame:CGRectMake(160, 100, 80, 45)];
[start setTitle:@"Start" forState:UIControlStateNormal];
[start addTarget:self action:@selector(buttonClicked)
    forControlEvents:UIControlEventTouchUpInside];
[self.view addSubview:start];

reset = [UIButton buttonWithType:UIButtonTypeRoundedRect];
[reset setFrame:CGRectMake(160, 180, 80, 45)];
[reset setTitle:@"Reset" forState:UIControlStateNormal];
[reset addTarget:self action:@selector(buttonClicked)
    forControlEvents:UIControlEventTouchUpInside];
[reset setEnabled:NO];
[self.view addSubview:reset];
};

stop = [UIButton buttonWithType:UIButtonTypeRoundedRect];
short int hh, mm, ss;
-(void) stepperValueUpdate:(id)sender
{
    UIStepper *st = (UIStepper *)sender;
    if (st == hstepper)
        [label setText:[NSString stringWithFormat:@"%..2i", (int)
            [hstepper value]]];
}

```

```

else if (st == mstepper)
{
    [mlabel setText:[NSString stringWithFormat:@"%.2i",
        (int)[mstepper value]]];
}
else if (st == sstepper)
{
    [slabel setText:[NSString stringWithFormat:@"%.2i",
        (int)[sstepper value]]];
}

-(void) buttonClicked : (id)sender
{
    UIButton * bt = (UIButton *) sender;
    if ([[bt currentTitle] isEqualToString:@"Start"])
    {
        [hstepper setUserInteractionEnabled:FALSE];
        [mstepper _____];
        [sstepper _____];
        [start setTitle:@"Stop" forState:UIControlStateNormal];
        [aeret setEnabled:TRUE];
        hh = (short int) [hstepper value];
        mm = (short int) [mstepper value];
        ss = (short int) [sstepper value];
        if (Hmer == nil)
            timer = [NSTimer scheduledTimerWithTimeInterval:1
                target: self selector:@selector(countDown) userInfo:nil
                repeats:YES];
    }
}

```

```

else if ([bt currentTitle] isEqualToString:@"Reset"]).
{
    [reset setEnabled: FALSE];
    [start setEnabled: @"start" forState: UIControlStateNormal];
    if ([start isEnabled] == FALSE).
        [Start setEnabled: YES];
    [hstepper setUserInteractionEnabled: TRUE];
    [mstepper setUserInteractionEnabled: TRUE];
    [sstepper setUserInteractionEnabled: TRUE];
    [timer invalidate];
    timer = nil;
    hh = mm = ss = 0;
    [sstepper setValue: ss];
    [mstepper setValue: mm];
    [hstepper setValue: hh];
    [label setText:[NSString stringWithFormat:@"% .2i", int
        sstepperValue]];
    [mstepper setValue: mm];
    [mlabel setText:[NSString stringWithFormat:@"% .2i", (int)
        & mstepperValue]];
    [hstepper setValue: hh];
    [hlabel setText:[NSString stringWithFormat:@"% .2i", (int)
        & hstepperValue]]];
}

else if ([bt currentTitle] isEqualToString:@"Stop"])
{
    [reset setEnabled: TRUE];
    [start setTitle:@"Start" forState: UIControlStateNormal];
    [start setEnabled: FALSE];
    [timer invalidate];
    timer = nil;
}

```

```

-(void) countDown
{
    if (ss>0)
        --ss;
    else if (ss==0)
    {
        ss=59;
        if (mm>0)
            --mm;
        else if (hh>0)
        {
            mm=59;
            --hh;
        }
    }
    [sstepper setValue:ss];
    [slabel setText:[NSString stringWithFormat:@"%d", (int)[sstepper value]]];
    [mstepper setValue:mm];
    [mlabel setText:[NSString stringWithFormat:@"%d", (int)[mstepper value]]];
    [hstepper setValue:hh];
    [hlabel setText:[NSString stringWithFormat:@"%d", (int)[hstepper value]]];
    if (hh==0 && mm==0 && ss==0)
    {
        UIAlertView *alert=[[UIAlertView alloc] initWithTitle:@"Alert"
                                                    message:@"Time Up" delegate:self cancelButtonTitle:@"OK"
                                                    otherButtonTitles:nil, nil];
        [timer invalidate];
        timer=nil;
        [alert show];
    }
}

```

```

-(void)alertView:(UIAlertView *)alertView clickedButtonAt-
Index:(NSInteger)buttonIndex.
{
    [start setTitle:@"Start" forState:UIControlStateNormal];
    [reset setEnabled:FALSE];
}
@end.

```

06/11/2013

Navigation Controller

- Navigation related functionalities can be implemented by using `UINavigationController`.
- When we are working with `UINavigationController` if we reqd to add a new view then we reqd. to go for `PushViewController` option, it is always works from left to right.
- When we required to remove view then we need to go for `popViewController` option & it will handle automatically by the `NavigationController` & it works towards right to left.
- When we are working with Navigation related appl' by default `UINavigationController` will handles pop operation. (view going back).

Example:-

- * Create a empty app with the name NaviApp.
- By default when we are constructing empty app we will get only AppDelegate class.
- Create a new objective c file with the name RootViewController.
[Custom view]

* Code in AppDelegate.h :-

```
#import <UIKit/UIKit.h>
@interface AppDelegate : UIResponder<UIApplicationDelegate>
@property (strong, nonatomic) UIWindow *window;
@property (strong, nonatomic) UINavigationController *navigationController;
@property (strong, nonatomic) RootViewController *rootViewController;
@end.
```

* Code in AppDelegate.m :-

```
#import "AppDelegate.h"
#import "RootViewController.h"
@implementation AppDelegate
-(void) dealloc {
{
    [-navigationController release];
    [-rootViewController release];
}
didFinishLaunchingWithOptions {
    self.rootViewController = [[RootViewController alloc]
        initWithNibName:nil bundle:nil];
}
```

↳ self.navigationController = [[UINavigationController alloc] initWithRootViewController:self.rootViewController];

[self.navigationController.navigationBar setTintColor:[UIColor blackColor]];

self.window.rootViewController = self.navigationController;

* code in RootViewController.h :-

```
{  
    -(void)nextViewButton;  
@end.
```

* code in RootViewController.m :-

```
#import "RootViewController.h"  
#import "FirstViewController.h"  
@implementation RootViewController.
```

```
-(id)initWithNibName...  
{  
    [self setTitle:@"RootView"];  
    return self;  
}
```

-(void)viewDidLoad

```
{  
    UIButton *button = [UIButton buttonWithType:UIButtonTypeRoundedRect];
```

[button setFrame:CGRectMake(20, 50, 80, 50)];

[button setTitle:@"Next" forState:UIControlStateNormal];

[button addTarget:self action:@selector(nextViewButton)
forControlEvents:UIControlEventTouchUpInside];

[self.view addSubview:button];

}

```
- (void)nextViewButton  
{  
    FirstViewController *nextView = [[FirstViewController alloc]  
        initWithNibName:nil bundle:nil];  
    [self.navigationController pushViewController:nextView  
        animated:YES];  
}
```

* Code in FirstViewController.h :-

```
{  
- (void)nextViewButton;  
@end.
```

Note: Same code which is available in RootViewController class but require to change Title.

- (void)nextViewButton.

```
{  
    SecondViewController *nextView = [[SecondViewController alloc]  
        initWithNibName:nil bundle:nil];  
    [self.navigationController pushViewController:nextView  
        animated:NO];  
}
```

- Another two view class required to add to the project with the name FirstViewController, SecondViewController

Example : 2

* Navigation View Controller with TableView.

- UITableView is required whenever we are representing the data with the help of rows.
- Create a single view app with the name NaviApp2.

07/11/2013

* code in AppDelegate.h :-

@ class ViewController;

@ property (strong, nonatomic) UINavigationController *navigationController;

* code in AppDelegate.m :-

```
- (void) dealloc  
{  
    [_navigationController release];
```

didFinishLaunchingWithOptions :-

```
{  
    self.navigationController = [[UINavigationController alloc]  
        initWithRootViewController : self.viewController];  
  
    [self.navigationController.navigationBar setTintColor : [UIColor  
        blackColor]];  
  
    self.window.rootViewController = self.navigationController;  
}
```

}

Note:- Drag few images to the project with the name airbook, apple, ipad, iphone, macbook.

* Code in ViewController.h :-

```
@interface ViewController : UIViewController < UITableViewDelegate,  
UITableViewDataSource>
```

```
{  
    IBOutlet UITableView *myTable;  
    NSArray *tableItems;  
}
```

```
@ Property (nonatomic, retain) IBOutlet UITableView *myTable;  
@ property (nonatomic, retain) NSArray *tableItems;
```

@ end.

- Open viewController.xib file & make following changes.
 - Place a UITableView on xib file.
 - Make the relation betn file's owner to tableView & select myTable outlet.
 - Make the relation from tableView to file's owner & select DataSource outlet.
 - Make the relation from tableView to file's owner & select delegate outlet.

* Code in ViewController.m :-

```
# import "ImageViewController.h"
```

```
@ implementation ViewController
```

```
@ synthesize myTable, tableItems;
```

```
- (void) viewDidload
{
    self.title = @"App Store";
    tableItems = [[NSMutableArray alloc] initWithObjects:@"airbook",
                 @"ipad", @"iphone", @"macbook", @"apple", nil];
}

#pragma mark UITableView

-(NSInteger)numberOfSectionsInTableView:(UITableView *)tableView
{
    return [tableItems count];
}

-(NSInteger)tableView:(UITableView *)tableView numberOfRowsInSection:(NSInteger)section
{
    return 2;
}

-(CGFloat)tableView:(UITableView *)tableView
heightForRowAtIndexPath:(NSIndexPath *)indexPath
{
    static NSString *cellIdentifier = @"cell";
    NSString *imageName;
    UITableViewCell *cell = [tableView dequeueReusableCellWithIdentifier:cellIdentifier];
    if (cell == nil)
    {
        cell = [[[UITableViewCell alloc] initWithStyle:UITableViewCellStyleDefault reuseIdentifier:cellIdentifier] autorelease];
    }
    cell.accessoryType = UITableViewCellAccessoryDisclosureButton;
}
```

```
if (indexPath.section == 0)
{
    cell.textLabel.text = [tableItems objectAtIndex: 0];
    imageName = [tableItems objectAtIndex: 0];
    cell.imageView.image = [UIImage imageNamed: imageName];
}

else if (indexPath.section == 1)
{
    cell.textLabel.text = [tableItems objectAtIndex: 1];
    imageName = [tableItems objectAtIndex: 1];
    cell.imageView.image = [UIImage imageNamed: imageName];
}

else if (indexPath.section == 2)
{
    cell.textLabel.text = [tableItems objectAtIndex: 2];
    imageName = [tableItems objectAtIndex: 2];
    cell.imageView.image = [UIImage imageNamed: imageName];
}

else if (indexPath.section == 3)
{
    cell.textLabel.text = [tableItems objectAtIndex: 3];
    imageName = [tableItems objectAtIndex: 3];
    cell.imageView.image = [UIImage imageNamed: imageName];
}

else if (indexPath.section == 4)
{
    cell.textLabel.text = [tableItems objectAtIndex: 4];
    imageName = [tableItems objectAtIndex: 4];
    cell.imageView.image = [UIImage imageNamed: imageName];
}

else;
return cell;
}
```

```
- (void) tableView: (UITableView *) tableView didSelectRowAtIndexPath:  
    IndexPath: (NSIndexPath *) indexPath,  
{  
    ImageViewController * nextView = [[ImageViewController alloc]  
        initWithImageName: [tableViewItem objectAtIndex: indexPath.  
            section]];  
    [self.navigationController pushViewController: nextView animated:  
        YES];  
}  
-(void) dealloc:  
{  
    [tableViewItem release];  
    [myTable release];  
    [super dealloc];  
}  
@ end.
```

* add a new Obj-C file with the name = ImageViewController.

* Code in ImageViewController.h:

```
@interface ImageViewController: UIViewController  
{  
    NSString * imageName;  
}  
-(id) initWithImageName: (NSString *) data;  
@ end.
```

* ImageViewController.m :-

```
@implementation ImageViewController  
-(id) initWithImageName: (NSString*) data  
{  
    imageName = data;  
    return ([self initWithNibName:nil bundle:nil]);  
}  
-(id) initWith  
-(void) viewDidLoad  
{  
    [self setTitle: imageName];  
    UIImageView * imageView = [[UIImageView alloc] initWithFrame: CGRectMake(10, 30, 150, 150)];  
    imageView setImage: [UIImage imageNamed: imageName];  
    [self.view addSubview: imageView];  
}
```

@ end.

- when we are working with UITableViewcontroller then mandatory to handle some specific function.

* -(NSInteger) numberOfSectionsInTableView: (UITableView*) tableView :-
→ This method always decide how many sections are required in tableView.

* -(NSInteger) tableView: (UITableView*) tableView
numberOfRowsInSection :-

→ This method will decide how many rows are required for a section.

* - (CGFloat) tableView: (UITableView *) tableView

heightForRowAtIndexPath: (NSIndexPath *) indexPath :-

→ This method will specify height of a row according to section or according to row also.

* - (UITableViewCell *) tableView: (UITableView *) tableView

cellForRowAtIndexPath: (NSIndexPath *) indexPath :-

→ This method will specify designing path of a cell i.e. UI of a cell.

* - (void) tableView: (UITableView *) tableView didSelectedRow

AtIndexPath:-

→ This method will call whenever we selected specific row of a table.

* - (NSString *) tableView: (UITableView *) tableView

titleForHeaderInSection: (NSInteger) section. :-

→ This method will specify what should be the title of a section.

08/11/2013

* Programmatically creating UITableView :-

- Create a single view app" with the name NaviApp3.
- Place few Png images to the project with the name airbook, ipad, iphone, macbook, & greenapple, banana, pineapple, Apple, mango .
- Create a new Obj-C class with the name imageViewcontroller.

* Code in AppDelegate.h :-

```
@property(strong, nonatomic) UINavigationController *navigationController;
```

```
@end.
```

* Code in AppDelegate.m :-

```
#import "imageViewcontroller"  
@implementation  
-(void)dealloc  
{  
    [_window release];  
    [_viewController release];  
    [navigationController release];  
}
```

```
- (BOOL) didFinishLaunchingWithOptions
```

```
{  
    self.navigationController = [[UINavigationController alloc] initWithRoot-  
    ViewController: self.viewController];
```

```
[self.navigationController.navigationBar setTintColor: [UIColor  
blackColor]];
```

```
self.window.rootViewController = self.navigationController;
```

7

* code in viewcontroller.h :-

```
@interface ViewController : UIViewController <UITableViewDelegate,  
UITableViewDataSource>  
{  
    UITableView *myTable;  
    NSArray *list1, *list2;  
}  
@property (nonatomic, retain) UITableView *myTable;  
@ _____ " _____ *list1, *list2;  
@ end.
```

* code in Viewcontroller.m :-

```
#import "ViewController.h"  
@interface ViewController()  
@end  
@implementation ViewController  
@synthesize myTable, list1, list2;  
- (void)viewDidLoad  
{  
    [self setTitle:@"List View"];  
    myTable=[[UITableView alloc] initWithFrame:CGRectMake(0, 0,  
    320, 415) style:UITableViewStylePlain];  
    [myTable setBackgroundColor:[UIColor clearColor]];  
    UIImageView *image=[[UIImageView alloc] initWithImage:  
    [UIImage imageNamed:@"background.png"]];
```

```
[myTable setBackgroundView : image];
[myTable setDataSource : self];
[myTable setDelegate : self];
[myTable setScrollsToTop : YES];
[myTable setPagingEnabled : YES];
[self.view addSubview : myTable];
list1 = [[NSMutableArray alloc] initWithObjects : @ "airbook", @ "ipad",
@ "iphone", @ "apple", @ "macbook", nil];
list2 = [[NSMutableArray alloc] initWithObjects : @ "greenapple", @ "banana",
@ "pineapple", @ "apple", @ "mango", nil];
```

5.

```
# Pragma - mark UITableView
-(NSInteger)numberOfSectionsInTableView : (UITableView *) tableView
{
    return 2;
}
-(NSInteger)tableView : (UITableView *) tableView numberOfRowsInSection :
InSection : (NSInteger) section
{
    if (section == 0)
        return [list1 count];
    else
        return [list2 count];
}
-(CGFloat)tableView : (UITableView *) tableView
heightForRowAtIndexPath : (NSIndexPath *) indexPath
{
    if (indexPath.section == 0)
        return 75;
```

```
        else
    {
        if (indexPath.row == 0)
            return 25;
        else if (indexPath.row == 1)
            return 40;
        else
            return 50;
    }
}

-(NSString *)tableView:(UITableView *)tableView
titleForHeaderInSection:(NSInteger)section
{
    if (section == 0)
        return @"App Store";
    else
        return @"Fault Store";
}

-(UITableViewCell *)tableView:(UITableView *)tableView
cellForRowAtIndexPath:(NSIndexPath *)indexPath
{
    static NSString *cellIdentifier = @"cell";
    NSString *imageName,
    UITableViewCell *cell = [tableView dequeueReusableCellWithIdentifier:
    cellIdentifier];
    if (cell == nil)
    {
        cell = [[[UITableViewCell alloc] initWithStyle:
        UITableViewCellStyleDefault reuseIdentifier:
        cellIdentifier] autorelease];
    }
}
```

```
if(indexPath.section == 0)
{
    cell.accessoryType = UITableViewCellAccessoryCheckmark;
    cell.textLabel.text = [list1 objectAtIndex:indexPath.row];
    [cell.textLabel setTextColor:[UIColor whiteColor]];
    imageName = [list2 objectAtIndex:indexPath.row];
    cell.imageView.image = [UIImage imageNamed:imageName];
}

else if(indexPath.section == 1)
{
    cell.accessoryType = UITableViewCellAccessoryDisclosureIndicator;
    cell.textLabel.text = [list2 objectAtIndex:indexPath.row];
    [cell.textLabel setTextColor:[UIColor whiteColor]];
    imageName = [list2 objectAtIndex:indexPath.row];
    cell.imageView.image = [UIImage imageNamed:imageName];
}

else;
return cell;
}

-(void)tableView:(UITableView *)tableView didSelectRowAtIndexPath:(NSIndexPath *)indexPath
{
    if(indexPath.section == 0)
    {
        ImageViewController *nextView = [[ImageViewController alloc]
            initWithImage:[list1 objectAtIndex:indexPath.row]];
        [self.navigationController pushViewController:nextView
            animated:YES];
    }
    else
}
```

```

    {
        UIImageViewController *nextView = [[UIImageViewController alloc]
            initWithFrame: [list2 objectAtIndex:indexPath.row]];
        [self.navigationController pushViewController: nextView
            animated: YES];
    }
}

-(void) dealloc.
{
    [myTable release]; [list1 release]; [list2 release];
    [super dealloc];
}

```

* Code in ImageViewController.h :-

```

@interface ImageViewController : UITImageView
    UIViewController
{
    NSString *imageName;
}
-(id) initWithImage: (NSString *) image;
@end.

```

* Code in ImageViewController.m :-

@ implementation ImageViewController.

```

-(id) initWithImage: (NSString *) image
{
    imageName = image;
    return [self initWithNibName: NSString * nibName:
        nil bundle: nil];
}

```

```

-(void) ViewDidLoad.
{
    [self setTitle: imageName];
    UIImage * image = [UIImage imageNamed: imageName];
    UIImageView * imageView = [[UIImageView alloc] initWithImage:
        image];
    [imageView setFrame: CGRectMake(10, 50, image.size.width,
        image.size.height)];
    [self.view addSubview: imageView];
}
@end.

```

09/11/2013

- * Example on Editing the UITableView's content
 - Create a single view app with the name NaviEdit App.
- * Code in AppDelegate.h :-

- @ Property (strong, nonatomic) UINavigationController *navigationController;
- @ end.
- * Code in AppDelegate.m :-

```

#import "ViewController.h"
-(void) dealloc
{
    [_navigationController release];
}

```

didFinishLaunchingWithOptions

```
{  
    self.navigationController = [[UINavigationController alloc] initWithRootViewController : self.viewController];  
    self.window.rootViewController = self.navigationController;  
}
```

* Code in Viewcontroller.h :-

```
@ interface Viewcontroller : UIViewController < UITableViewDelegate,  
UITableViewDataSource, UIAlertViewDelegate, UITextFieldDelegate  
>;
```

```
{  
    IBOutlet UITableView * myTableView;  
    NSMutableArray * dataItems dataList;  
    UITextField * rowTitle; textData;  
}
```

```
@ property (nonatomic, retain) IBOutlet UITableView * myTableView;  
NSMutableArray * dataItems list;  
UITextField * rowTitle textData;
```

```
- (void) Edit : (id) sender;
```

```
@ end.
```

* Code in Viewcontroller.m :-

```
@synthesize myTableView, dataItems, rowTitle; textData;
```

```
- (void) ViewDidLoad.
```

```
{  
    self.title  
    self setTitle : @"App Store";  
    myTableView = [[UITableView alloc] initWithFrame : CGRectMake  
    (0, 0, 320, 415) style: UITableViewStylePlain];
```

```
[tableView setDataSource: self];
[tableView setDelegate: self];
[self.view addSubview: tableView];
datalist = [NSMutableArray alloc] initWithObjects: @[@"airbook",
@@"ipad", @@"iphone", @@"macbook", nil];
}

-(void) viewDidAppear: (BOOL) animated {
{
    [super viewDidAppear: animated];
    UIBarButtonItem *editButton = [[UIBarButtonItem alloc]
        initWithTitle: @"Edit" style: UIBarButtonItemStyleBordered
        target: self action: @selector(Edit:)];
    [self.navigationItem setRightBarButtonItem: editButton];
}
}

-(void) Edit : (id)sender {
{
    if (self.editing) {
        [super setEditing: NO animated: NO];
        [tableView setEditing: NO animated: NO];
        [tableView reloadData];
        [self.navigationItem.rightBarButtonItem setTitle: @"Edit"];
        [self.navigationItem.rightBarButtonItem setStyle:
        UIBarButtonItemStylePlain];
    }
    else {
        [super setEditing: YES animated: YES];
        [tableView setEditing: YES animated: YES];
    }
}
```

```
- [tableView reloadData];
  [self.navigationItem.rightBarButtonItem setTitle:@"Done"];
  [self.navigationItem.rightBarButtonItem setStyle:
    UIBarButtonStyleDone];
}
- (void)dealloc
{
  [tableView release];
  [dataArray release];
  [textData release];
  [super dealloc];
}
// pragma - make UITableView
-(NSInteger)numberOfSectionsInTableView:(UITableView*)
  tableView
{
  return 1;
}
-(NSInteger)tableView:(UITableView*)tableView
  numberOfRowsInSection:(NSInteger)section
{
  int rowCount = [dataArray count];
  if (self.editing)
    rowCount++;
  return rowCount;
}
-(UITableViewCell*)tableView:(UITableView*)tableView
  cellForRowAtIndexPath:(NSIndexPath*)indexPath
{
  static NSString * cellIdentifier = @"cell";

```

```
UITableViewController *cell = [tableView dequeueReusableCellWithIdentifier:  
    forIndexPath:CellIdentifier];  
if (cell == nil)  
{  
    cell = [[[UITableViewController alloc] initWithStyle:UITableViewStyleDefault  
        reuseIdentifier:CellIdentifier] autorelease];  
    cell.editingAccessoryType = UITableViewCellAccessoryNone; // Enable editing.  
}  
int count = 0;  
if (self.editing && indexPath.row != 0)  
    count = 1;  
if (indexPath.row == ([dataArray count]) && self.editing)  
{  
    cell.textLabel.text = @"Add Data";  
    return cell;  
}  
cell.textLabel.text = [dataArray objectAtIndex:indexPath.row];  
cell.accessoryType = UITableViewCellAccessoryDetailDisclosureButton;  
return cell;  
}  
-(UITableViewCellEditingStyle)tableView:(UITableView*)tableView  
    editingStyleForRowAtIndexPath:(NSIndexPath*)indexPath;  
{  
    if (self.editing == NO || !indexPath)  
        return UITableViewCellEditingStyleNone;  
    if (self.editing && indexPath.row == ([dataArray count]))  
    {  
        return UITableViewCellEditingStyleInsert;  
    }  
}
```

```

    else
    {
        return UITableViewCellStyleDelete;
    }
    return UITableViewCellStyleNone;
}

-(void) tableView:(UITableView*) aTableView commitEditingStyle:(UITableViewCellEditingStyle) editingStyle
forRowAtIndexPath:(NSIndexPath*) indexPath
{
    if (editingStyle == UITableViewCellStyleDelete)
    {
        [dataArray removeObjectAtIndex: indexPath.row];
        [tableView reloadData];
    }
    else if (editingStyle == UITableViewCellStyleInsert)
    {
        UIAlertView* alert = [[[UIAlertView alloc] initWithTitle:@"New Row"
                                                       message:@"\n\n\n"
                                                       delegate:self
                                                       cancelButtonTitle:@"Cancel"
                                                       otherButtonTitles:@"Save",
                                                       nil] autorelease];
        CGRect rect = {12, 60, 260, 25};
        textData = [[[UITextField alloc] initWithFrame:rect]
                    autorelease];
        [textData setReturnKeyType:UIReturnKeyDone];
        [textData setDelegate:self];
        textData.backgroundColor = [UIColor whiteColor];
        [textData becomeFirstResponder];
        [alert addSubview:textData];
        [alert show];
    }
}

```

```
# pragma -mark UIAlertView.  
-(void)alertView:(UIAlertView *)alertView  
clickedButtonAtIndex:(NSInteger)buttonIndex  
{  
    if (buttonIndex == 0)  
    {  
        // cancel clicked... no action.  
    }  
    else if (buttonIndex == 1 && [textData.text length] != 0)  
    {  
        // save clicked  
        [datalist insertObject:textData.text atIndex:[datalist  
            count]];  
        [tableView reloadData];  
    }  
}  
  
# pragma -mark UITextField  
-(BOOL)textFieldShouldReturn:(UITextField *)textField  
{  
    [textField resignFirstResponder];  
    return TRUE;  
}  
  
@end.
```

* Example on creating customized UITableView cell designing.

- Create a single view app! with the name NaviApp5.
- Create a customized cell view class with the name CellView which is subclass of type "UIView".

* code in cellview.h :-

```
@interface CellView : UIView
{
    UIImageView * imageView;
    UILabel * labelData;
}

@property (nonatomic, retain) UIImageView * imageView;
@property (nonatomic, retain) UILabel * labelData;

@end.
```

* code in CellView.m :-

```
@synthesize imageView, labelData.

-(id) initWithFrame : (CGRect) frame
{
    self = [super initWithFrame : frame];
    if (self)
    {
        imageView = [[[UIImageView alloc] initWithFrame :
        CGRectMake(80, 5, 120, 57)] autorelease];
        [self addSubview: imageView];

        labelData = [[[UILabel alloc] initWithFrame : CGRectMake(
        110, 60, 100, 20)] autorelease];
        [labelData setBackgroundColor : [UIColor clearColor]];
        [self addSubview: labelData];
    }
    return self;
}
```

* Code in AppDelegate.h:-

@ Property (strong, nonatomic) UINavigation Controller *
navigationController;

@ end.

* Code in AppDelegate.m:-

#import "cellView.h"
#import "imageviewController.h".

=====

@synthesize table, tableData;

-(void) viewDidLoad.

{

[self setTitle:@"App Store"];
table = [[UITableView alloc] initWithFrame:CGRectMake(0, 0, 320, 415) style:UITableViewStylePlain];

[table setBack

#import "viewController.h"

=====

-(void) dealloc.

{

[-NavigationController release];

}.

-(BOOL) application:(UIApplication *)

didFinishLaunching -

{

self.viewController = [[ViewController alloc]

self.navigationController = [[UINavigationController alloc] initWithRootViewController:self.viewController];

self.window.rootViewController = self.navigationController;

}

* Code in ViewController.h :-

- Add few images to the project & create a new class with the name ImageViewController.

```
@interface ViewController : UIViewController <UITableViewDataSource,  
UITableViewDelegate>
```

{

```
    UITableView *table;  
    NSArray *tableData;
```

}

```
@property (nonatomic, retain) UITableView *table;  
NSArray *tableData;
```

```
@end.
```

* Code in ViewController.m :-

```
#import "CellView.h"  
#import "ImageViewController.h"
```

```
@synthesize table, tableData;
```

```
- (void) viewDidLoad
```

{

```
[self setTitle:@"App Store"];
```

```
[_table = [[UITableView alloc] initWithFrame:[CGRectMake  
(0, 0, 320, 415) style:UITableViewStylePlain]];
```

```
[_table setBackgroundColor:[UIColor whiteColor]];
```

```
[_table setSeparatorStyle:UITableViewCellSeparatorStyleNone];
```

```
[_table setDataSource:self];
```

```
[_table setDelegate:self];
```

```
[_self.view addSubview:_table];
```

```
tableData = [[NSMutableArray alloc] initWithObjects: @“photo1.jpg”,  
@“Photo2.jpg”, @“Photo3.jpg”, @“Photo4.JPG”, @“Photo5.jpg”,  
@“photo6.png”, @“view1.png”, @“view2.png”, @“view3.png”,  
nil];  
}  
-(void) dealloc  
{  
    [tableData release];  
    [table release];  
    [super dealloc];  
}  
# pragma -mark UITableView  
-(NSInteger)numberOfSectionsInTableView: (UITableView*)  
tableView  
{  
    return ([tableData count]);  
}  
-(NSInteger)tableView: (UITableView*) tableView  
numberOfRowsInSection: (NSInteger) section  
{  
    return 1;  
}  
-(CGFloat)tableView: (UITableView*) tableView  
heightForRowAtIndexPath: (NSIndexPath*) indexPath  
{  
    return 100;  
}  
-(UITableViewCell*) tableView: (UITableView*) tableView  
cellForRowAtIndexPath: (NSIndexPath*) indexPath  
{  
    static NSString * cellIdentifier = @“Cell”;
```

```

UITableViewController * controller = [UITableViewController alloc] initWithStyle:UITableViewStylePlain];
controller.tableView.delegate = self;
controller.tableView.dataSource = self;
[controller.tableView reloadData];
}

- (UITableViewCell *)tableView:(UITableView *)tableView cellForRowAtIndexPath:(NSIndexPath *)indexPath {
    static NSString *CellIdentifier = @"Cell";
    UITableViewCell *cell = [tableView dequeueReusableCellWithIdentifier:CellIdentifier];
    if (cell == nil) {
        cell = [[[UITableViewCell alloc] initWithStyle:UITableViewCellStyleDefault reuseIdentifier:CellIdentifier] autorelease];
    }
    return cell;
}

- (void)tableView:(UITableView *)tableView didSelectRowAtIndexPath:(NSIndexPath *)indexPath {
    UITableViewCell *selectedCell = [tableView cellForRowAtIndexPath:indexPath];
    selectedCell.accessoryType = UITableViewCellAccessoryCheckmark;
    [selectedCell setSelectionBackgroundImage:[UIImage imageNamed:@"checkmark.png"] forState:UIControlStateSelected];
    [selectedCell setSelectionBackgroundImage:[UIImage imageNamed:@"checkmark.png"] forState:UIControlStateHighlighted];
    [selectedCell setSelectionBackgroundImage:[UIImage imageNamed:@"checkmark.png"] forState:UIControlStateNormal];
}

```

Or

```
UITImage *image = [UITImage imageNamed: [TableData  
objectAtIndex: indexPath.section] objectAtIndex:  
indexPath.row];  
NSString *title = [TableData objectAtIndex:indexPath.row];  
imageViewController *nextView = [[imageViewController alloc]  
initWithImage: image imageName: title];  
[self.navigationController pushViewController: nextView  
animated: YES];  
}
```

* Code in ImageViewController.h:-

```
@interface imageViewController : UIViewController  
{  
    UITImage *image;  
    NSString *title;  
}  
-(id)initWithImage:(UITImage *)image imageName:(NSString *)  
name;  
@end.
```

* code in ImageViewController.m:-

```
-(id)initWithImage:(UITImage *)image imageName:(NSString *)  
name.  
{  
    image = image;  
    title = name;  
    return ([self initWithNibName: nil bundle: nil]);  
}
```

```

- (void) viewDidLoad
{
    UIImageView * imageView = [[UIImageView alloc] initWithFrame:
        CGRectMake(10, 10, 150, 150)];
    [imageView setImage:image];
    [self.view addSubview:imageView];
    [self setTitle:title];
    [self.view setBackgroundColor:[UIColor whiteColor]];
}
@end

```

~~11/11/2013~~
~~12/11/2013~~

* Example on UISearchBar :-

- Create a single view app' with the name SearchBarApp.
- Create a new class with the name DetailViewController.
- Add few images to the project with some specific counties.

* Code in AppDelegate.h :-

```
@property (strong, nonatomic) UINavigationController * navigationController;
```

* Code in AppDelegate.m :-

```
self.navigationController = [[UINavigationController alloc] initWithRootViewController:self.viewController];
```

```
self.window.rootViewController = self.navigationController;
```

* Code in Viewcontroller.h :-

@ interface ViewController : UIViewController < UITableViewDataSource,
UITableviewDelegate, UISearchBarDelegate>.

```
{  
    UITableView * mytable ;  
    UISearchBar * searchbar ;  
    NSMutableArray * items ;  
    NSMutableArray * copylistofItems ;  
    BOOL searching ;  
    BOOL letUserSelectRow ;  
}
```

@ Property (nonatomic, retain) UITableView * mytable ;
@ Property (nonatomic, retain) UISearchBar * searchbar ;
@ Property (nonatomic, retain) NSMutableArray * items ;
@ Property (nonatomic, retain) NSMutableArray * copylistofItems ;

@ end.

* Code in Viewcontroller.m :-

@ Synthesize mytable, searchbar, items, copylistofItems ;

```
- (void) viewDidLoad  
{  
    self.navigationController.title = @ "Counters" ;  
    self.navigationController.navigationBar.tintColor = [UIColor  
        colorWithRed: 17 / 255.0f green: 4 / 255.0f blue: 150 / 255.0f  
        alpha: 1] ;
```

```
myTable = [[UITableView alloc] initWithFrame: CGRectMake(0, 0, 320, 400)]
style:UITableViewStylePlain];
[myTable setDelegate:self];
[myTable setDataSource:self];
[self.view addSubview:myTable];

Searchbar = [[UISearchBar alloc] initWithFrame: CGRectMake(0, 0, 320,
50)];
[Searchbar setDelegate:self];
[searchbar setAutoresizingType: UITableViewAutoresizingFlexibleHeight];
self.myTable.tableHeaderView = searchbar;
// [self.view addSubview:searchbar];

Item = [[NSMutableArray alloc] init];
// Add item

NSArray * countriesToLiveInArray = [NSArray arrayWithObjects:
@ "Pakistan", @ "China", @ "Sri Lanka", @ "Switzerland",
@ "New Zealand", @ "Greenland", @ "Africa", @ "Zimbabwe",
@ "Canada", @ "Japan", @ "Afghanistan", nil];
NSDictionary * countriesToLiveInDict = [NSDictionary dictionaryWithObjects:
forKeys:@ "Country", @ "Country"];
NSArray * countriesLivedInArray = [NSArray arrayWithObjects:
@ "India", @ "USA", @ "Nepal", nil];
NSDictionary * countriesLivedInDict = [NSDictionary dictionaryWithObjects:
forKeys:@ "Country", @ "Country"];
```

```
[Item addObject:countriesToLiveInDict]; //obj1
```

```
[Item addObject:countriesLivedInDict]; //obj2
```

```
CopyListOfItem = [[NSMutableArray alloc] init];
```

```
}
```

```
-(void) viewDidLoadAppeared : (BOOL) animated.
```

```
{ [super viewDidLoadAppeared : animated],
```

```
searching = NO;
```

```
letUserSelectRow = YES;
```

```
searchbar.text = @"";
```

```
[searchbar resignFirstResponder];
```

```
[myTable reloadData];
```

```
}
```

```
# pragma mark TableView Data Source.
```

```
- (NSInteger) numberOfSectionsInTableView : (UITableView *) tableView
```

```
{
```

```
if (searching)
```

```
return 1;
```

```
else
```

```
return [Item count]; // 2 sections
```

```
}
```

```
- (NSInteger) tableView : (UITableView *) tableView
```

```
numberOfRowsInSection : (NSInteger) section
```

```
{
```

```
if (searching)
```

```
return [CopyListOfItem count];
```

```
{ else
```

```
NSDictionary * dictionary = [item objectForKey: @"section"];
NSArray * array = [dictionary objectForKey: @"countries"];
return [array count];
}

-(NSString *)tableView:(UITableView *)tableView
titleForHeaderInSection:(NSInteger)section
{
    if (searching)
        return @"Search Result";
    if (section == 0)
        return @"Countries to visit";
    else
        return @"Countries visited";
}

-(NSInteger)tableView:(UITableView *)tableView
sectionForSectionIndexTitle:(NSString *)titleAtIndex:(NSInteger) index
{
    if (searching)
        return -1;
    return index / 2;
}

-(UITableViewCell *)tableView:(UITableView *)tableView
cellForRowAtIndexPath:(NSIndexPath *)indexPath
{
    static NSString * cellIdentifier = @"cell";
    UITableViewCell * cell = [tableView dequeueReusableCellWithIdentifier: cellIdentifier];
}
```

```
if (cell == nil)
{
    cell = [[[UITableViewCell alloc] initWithStyle:
        UITableViewCellStyleDefault reuseIdentifier:
            reuseIdentifier] autorelease];
    cell.accessoryType = UITableViewCellAccessoryCheckmark];
}

if (searching)
    cell.textLabel.text = [copyListofItems objectAtIndex:
        indexPath.row];
else
{
    // First get the dictionary obj
    NSDictionary *dictionary = [Items objectAtIndex:indexPath.section];
    NSArray *array = [dictionary objectForKey:@"countries"];
    NSString *cellText = [array objectAtIndex:indexPath.row];
    cell.textLabel.text = cellText;
}
return cell;
}

-(void) tableView: (UITableView *) tableView didSelectRowAtIndexPath:
   IndexPath: (NSIndexPath *) indexPath
{
    NSString *selectedCountry = nil;
    if (searching)
        selectedCountry = [copyListofItems objectAtIndex:
            indexPath.row];
}
```

```
    else
    {
        NSDictionary * dictionary = [item objectForKey:@"indexPath-
section];
        NSArray * array = [dictionary objectForKey:@"countries"];
        selectedCountry = [array objectAtIndex:indexPath.row];
    }
    detailViewController * dvController = [[detailViewController alloc]
        initWithTitle:selectedCountry];
    [self.navigationController pushViewController:dvController
        animated:YES];
    [dvController release];
    dvController = nil;
}

-(NSIndexPath *)tableView:(UITableView *)theTableView
willSelectRowAtIndexPath:(NSIndexPath *)indexPath
{
    if (!letUserSelectRow)
        return indexPath;
    else
        return nil;
}

#pragma -mark UISearchBar

-(void)searchBarTextDidBeginEditing:(UISearchBar *)theSearchBar
{
    if (searching)
        return;
}
```

```
    Searching = YES;
    letUserSelectRow = NO;
    self.myTable.scrollEnabled = NO;
}

-(void) searchBar:(UISearchBar *) theSearchBar textDidChange:
  (NSString *) searchText
{
    // Remove all objects first.
    [copyListOfItems removeAllObjects];
    if ([searchText length] > 0)
    {
        Searching = YES;
        letUserSelectRow = YES;
        self.myTable.scrollEnabled = YES;
        [self.searchTableView];
    }
    else
    {
        Searching = NO;
        letUserSelectRow = NO;
        self.myTable.scrollEnabled = NO;
    }
    [self.myTable reloadData];
}

-(void) searchBarCancelButtonClicked:(UISearchBar *) theSearchBar
{
    [Searchbar resignFirstResponder];
    [self.searchTableView];
}
```

```
- (void) searchTableView
{
    NSString * searchText = searchBar.text;
    NSMutableArray * searchArray = [[NSMutableArray alloc] init];
    for (NSDictionary * dictionary in items)
    {
        NSArray * array = [dictionary objectForKey:@"counters"];
        [searchArray addObjectFromArray:array];
    }
    for (NSString * sTemp in searchArray)
    {
        NSRange titleResultRange = [sTemp rangeOfString:searchText
            options:NSCaseInsensitiveSearch];
        if (titleResultRange.length > 0)
            [copyListOfItems addObject:sTemp];
    }
    [searchArray release];
    searchArray = nil;
}
```

```
- (void) doneSearching_Clicked : (id)sender
{
    searchBar.text = @"";
    [searchBar resignFirstResponder];
    letUserSelectRow = YES;
    searching = NO;
    self.tableView.scrollEnabled = YES;
    [self.tableView reloadData];
}
```

```
- (void) dealloc  
{  
    [copyListofItems release];  
    [searchBar release];  
    [myTable release];  
    [Items release];  
    [super dealloc];  
}
```

* Code in detailViewController.h :-

```
@interface detailViewController : UIViewController  
{  
    NSString * countryName;  
    UIImageView * imageView;  
}  
-(id) initWithTitle: (NSString *) name;  
@end.
```

* Code in detailViewController.m :-

```
-(id) initWithTitle: (NSString *) name  
{  
    countryName = name;  
    return ([self initWithFrame: nil bundle: nil]);  
}
```

- (void) viewDidLoad.

```
{  
    imageView = [[UIImageView alloc] initWithFrame:  
        CGRectMake(0, 50, 200, 200)];
```

```
[imageView setImage:[UIImage imageNamed:countryName]];
[self.view addSubview:imageView];
}
```

* UIScrollView :-

- Generally whenever the content is more than device's height & width the recommended to go for scrollview.
- When we are working with scrollview always recommended to change the property called content size.

Example:

- * Create a single view appl' with the name ScrollViewApp
- * Code in viewController.h :-

```
@interface viewController : UIViewController <UITextFieldDelegate>
{
```

```
UIScrollView * scrollview;
```

```
UITextField * textField1;
```

```
UITextField * textField2;
```

```
UITextField * field;
```

```
Bool displayKeyboard;
```

```
CGPoint offset;
```

```
}
```

```
@property (nonatomic, retain) UIScrollView * scrollview;
```

```
-----"-----
```

```
-----"-----
```

```
UITextField * textField1;
```

```
UITextField * textField2;
```

```
@ end.
```

* code in ViewController.m :-

13/11/2013

```
@synthesize textField1, textField2, scrollView;

-(void)viewDidLoad{
{
    ScrollView = [[UIScrollView alloc] initWithFrame: CGRectMake
(0, 0, 320.0f, 460.0f)];
    [self.view addSubview scrollView];

    textField1 = [[UITextField alloc] initWithFrame: CGRectMake
(50, 230, 200, 40)];
    [textField1 setBorderStyle: UITextBorderStyleRoundedRect];
    [textField1 setReturnKeyType: UIReturnKeyDone];
    [textField1 setDelegate: self];
    [scrollView addSubview: textField1];

    textField2 = [[UITextField alloc] initWithFrame: CGRectMake
(50, 360, 200, 40)];
    [textField2 setDelegate: self];
    [textField2 setReturnKeyType: UIReturnKeyDone];
    [textField2 setBorderStyle: UITextBorderStyleRoundedRect];
    [scrollView addSubview: textField2];
}

-(void)viewWillAppear:(BOOL)animated{
{
    [[NSNotificationCenter defaultCenter] addObserver: self
    selector:@selector(keyboardDidShow:) name:

```

```
UIKeyboardDidShowNotification object: nil];
    [NSNotificationCenter defaultCenter] addObserver: self selector:
        @selector(keyboardDidHide:) name;
UIKeyboardDidHideNotification object: nil];
scrollView.contentSize = CGSizeMake(320.0f, 460.0f);
displayKeyboard = NO;
}

-(void) viewWillDisappear: (BOOL)animated {
    [[NSNotificationCenter defaultCenter] removeObserver: self];
}

-(void) keyboardDidShow: (NSNotification*) notif {
    if (displayKeyboard) {
        return;
    }
    NSDictionary* info = [notif userInfo]; // holds multiple information
    // in the form of dictionary so we need to extract using
    // object key value.
    // NSValue* aValue = [info objectForKey: UIKeyboardBoundsUserInfoKey];
    // not supported in iOS 6.0.
    NSValue*aValue = [info objectForKey: UIKeyboardFrameBeginUserInfoKey];
    CGSize keyboardSize = [aValue CGRectValue].size;
    offset = scrollView.contentOffset;
    // holding old frame of scrollView
```

```
CRect viewFrame = scrollView.frame;
// assigning current frame to viewFrame obj.

viewFrame.size.height -= keyboardSize.height;
// decrement frame height by keyboard height.

scrollView.frame = viewFrame; // update new frame to scrollView

CRect textFieldRect = [field frame];
// find textField position.

textFieldRect.origin.y += 10;
// increase y value of text field by 10.

[scrollView scrollRectToVisible: textFieldRect animated: YES];
displayKeyboard = YES;

{
    - (void) keyboardDidHide: (NSNotification *) notif
    {
        if (!displayKeyboard)
            return;

        scrollView.frame = CGRectMake(0, 0, 320.0f, 460.0f);
        scrollView.contentOffset = offset;
        displayKeyboard = NO;
    }

    - (BOOL) textFieldShouldBeginEditing: (UITextField *) textField
    {
        field = textField;
        return YES;
    }
}
```

```
- (BOOL) textFieldShouldReturn: (UITextField *) textField  
{  
    [textField resignFirstResponder];  
    return YES;  
}
```

—————xox—————

* UIPageController :-

- By using this controller we can place multiple view controllers in a single page.
- When we are working UIPageController we required to set two properties i.e. number of pages & currentPageIndex.

Ex:- Create a singleView app with the name PageControllerApp

- Create 3 obj-C classes with the name Page1ViewController, Page2ViewController, and Page3ViewController.

* code in View Controller.h :-

```
@interface View Controller : UIViewController < UIScrollViewDelegate
```

```
{  
    UIScrollView * scrollView;  
    UIPageControl * pageControl;  
    NSMutableArray * viewControllers;  
    BOOL pageControlUsed;  
}
```

```
@property (nonatomic, retain) UIScrollView * scrollView;
```

```
—————"————— UIPageControl * pageControl;
```

```
—————"————— NSMutableArray * ViewControllers;
```

```
- (void) changePage : (id) sender;
```

```
@end
```

* Code in ViewController.m :-

```
#import "PageViewController.h"
____ 2 _____
____ " 3 _____"

static NSUInteger kNumberOfPages;

@interface ViewController (Private Methods)
- (void)loadScrollViewWithPage:(int)page;
- (void)scrollViewDidScroll:(UIScrollView *)sender;

@end

@implementation ViewController;

@synthesize scrollView, pageControl, viewControllers;
- (void)viewDidLoad
{
    [self.view setBackgroundColor:[UIColor blackColor]];
    PageViewController *page1=[[PageViewController alloc] init];
    ____ 2 _____" 2 = _____"; 
    ____ 3 _____" 3 = _____";
    viewControllers=[[NSMutableArray alloc] init];
    [viewControllers addObject:page1];
    [_____ " _____ Page2];
    [_____ " _____ Page3];
}

// Add all objects to the view.
```

```
    knumberOfPages = [viewController count];
    // count total no. of views

    PageControl = [[UIPageControl alloc] initWithFrame: CGRectMake
        (130.0f, 380.0f, 60.0f, 40.0f)];
    [PageControl setNumberOfPages : knumberOfPages];
    [PageControl setCurrentPage:0];
    [PageControl addTarget: self action:@selector(changePage:)
        forControlEvents:UIControlEventValueChanged];
    [self.view addSubview : PageControl];
    [self.view bringSubviewToFront : PageControl];

    ScrollView = [[UIScrollView alloc] initWithFrame: CGRectMake
        (0, 30, 320, 320)];
    [self.view addSubview: ScrollView];
    // [ScrollView addSubview : PageControl];

    ScrollView.contentSize = CGSizeMake (ScrollView.frame.size.width * knumberOfPages, 1);

    ScrollView.showsHorizontalScrollIndicator = NO;
    ScrollView.showsVerticalScrollIndicator = NO;
    ScrollView.scrollToTop = NO;
    ScrollView.pagingEnabled = YES;
    // if it is NO then stop view when user is stop
    // scrolling.

    ScrollView.delegate = self;
    [self loadScrollViewWithPage:0];
    [_____ : 1];
}
```

```
# pragma -mark protocol methods
-(void)loadScrollViewWithPage:(int)page{
{
    if (page<0)
        return;
    if (page>=kNumberOfPages) //when last page on view.
        return;
    if (page==0)
    {
        Page1ViewController *test1=[[Page1ViewController alloc]
            initWithNibName:nil bundle:nil];
        [viewController replaceObjectAtIndex:page withObject:test1];
        if (test1.view.superview==nil)
        {
            CGRect frame=scrollView.frame;
            frame.origin.x=frame.size.width*page;
            frame.origin.y=0;
            test1.view.frame=frame;
            [scrollView addSubview:test1.view];
        }
        if (page==1)
    {
        Page2ViewController *test2=[[Page2ViewController alloc]
            initWithNibName:nil bundle:nil];
        [viewController replaceObjectAtIndex:page withObject:test2];
        if (test2.view.superview==nil)
        {
            CGRect frame=scrollView.frame;
```

```

frame.origin.x = frame.size.width * page;
frame.origin.y = 0;
test2.view.frame = frame;
[scrollView addSubview: test2.view];
}

}

if (page == 2)
{
    Page3ViewController * test2 = [[Page3ViewController alloc]
        initWithNibName:nil bundle:nil];
    [viewControllers replaceObjectAtIndex: page withObject: test2];
    if (test2.view.superview == nil)
    {
        CGRect frame = scrollView.frame;
        frame.origin.x = frame.size.width * page;
        frame.origin.y = 0;
        test2.view.frame = frame;
        [scrollView addSubview: test2.view];
    }
}
}

#pragma mark UIScrollViewDelegate
-(void) scrollViewDidScroll: (UIScrollView *) sender
{
    if (pageControlUsed)
    {
        // do nothing - the scroll was initiated from the page control, do not let the user dragging
        return;
    }
}

```

control button

// switch the indicator when more than 50% of the previous/next page is visible.

[Cfloat pageWidth = scrollView.frame.size.width,

int page = floor((scrollView.contentOffset.x - pageWidth / 2) / pageWidth) + 1;

pageControl.currentPage = page;

// load the visible page and the page on either side of it (to avoid flashes when the user starts scrolling).

[self loadScrollViewWithPage: page - 1];

[————— : page];

[————— : page + 1];

}

// At the beginning of scroll dragging, reset the boolean used when scroll originate from the UIPageControl.

- (void) scrollViewWillBeginDragging : (UIScrollView *) scrollView

{

pageControlUsed = NO;

}

// At the end of scroll animation, reset the boolean used when scroll originate from the UIPageControl

- (void) scrollViewDidEndDecelerating : (UIScrollView *) scrollView

{

pageControlUsed = NO;

}

```
# pragma -mark UIPageControlButtonAction

-(void)changePage:(id)sender
{
    //UIPageControl *temp = (UIPageControl *)sender;
    //int page = temp.currentPage;
    int page = pagecontrol.currentPage;

    [self loadScrollViewWithPage:Page-1];
    [_____ " _____ : Page];
    [_____ " _____ : Page+1];

    // update the scroll view to the appropriate page.

    CGRect frame = scrollView.frame;
    frame.origin.x = frame.size.width * page;
    frame.origin.y = 0;
    [scrollView scrollRectToVisible:frame animated:YES];

    // set the boolean used when scrolls originate from
    // the UIPageControl. See *scrollViewDidScroll; above.
    PageControlUsed = YES;
}

-(void)dealloc
{
    [scrollView release];
    [viewController release];
    [pagecontrol release];
    [super dealloc];
}
```

14/11/2013

* Picker Controls:-

- By using picker controls we can display multiple list of items for selection.
- PickerControls can be with one columns or multiple columns also.
- On selection of the data through delegate methods we can extract specific content from pickercontrols.

* Create a new iOS appl" of tabbed application template with the name PickerApp.

- add another view to the current template with the name ThirdViewController & place two Png files with name Third.png & Third@2x.png (30x30, 60x60 pixels).

* Code in FirstViewController.h:-

```
@interface FirstViewController : UIViewController  
{  
    IBOutlet UIDatePicker *datePicker;  
}  
  
@property (nonatomic, retain) UIDatePicker *datePicker;  
  
-(IBAction)buttonClicked;  
  
@end.
```

- Open FirstViewController.xib & place
- Place a DatePicker on view & set the mode as Date & Time
- Make the relation b/w file's owner to DatePicker & select DatePicker object.
- Place a roundedRect button & change the title as "select".
- Make the relation from button to first responder. & select the method buttonClicked.

* code in FirstViewController.m:-

```

@synthesize datePicker;
-(IBAction)buttonClicked:(id)sender
{
    NSDate * selected = [datePicker date];
    //NSDate * selected = [[NSDate date] J]; default date.
    NSString * message = [[NSString alloc] initWithFormat:
        @"The date & time is %@" , selected];
    UIAlertView * alert = [[UIAlertView alloc] initWithTitle:
        @"Date & Time selected" message: message delegate: nil
        cancelButtonTitle : @ "YES" otherButtonTitles: nil];
    [alert show];
    [alert release];
}

-(void) viewDidLoad
{
    NSTimer * start = [NSTimer scheduledTimerWithTimeInterval:
        target: self selector: @selected (updateTime)
        userInfo: nil repeats: TRUE];
}

```

```
- (void) updateTime  
{  
    NSDate * now = [[[NSDate alloc] init] autorelease];  
    [datePicker setDate:now animated:YES];  
}
```

* code in SecondViewController.h :-

```
@ interface SecondViewController : UIViewController,  
< UIPickerViewDataSource, UIPickerViewDelegate>
```

{

```
    IBOutlet UIPickerView * singlePicker;  
    NSArray * pickerData;
```

}

```
@ Property (nonatomic, retain) UIPickerView * singlePicker;  
@ _____ NSArray * pickerData;
```

```
- (void) buttonClicked;
```

- // • Open SecondViewController.xib & place pickerView.
- // • Make the relation from file's owner to picker & select singlePicker outlet.
- // • Make the relation picker to file's owner & select datasource & delegate.

* code in SecondViewController.m :-

```
@synthesize pickerData, singlePicker;
```

```
- (void) viewDidLoad {
{
    NSArray * array = [[NSArray alloc] initWithObjects: @"iPhone",
    @"iPad", @"iMac", @"Mac", @"iBook", @"Safari", nil];
    self.pickerData = array;
    [array release];

    self.singlePicker = [[UIPickerView alloc] initWithFrame:
    CGRectMake(0, 0, 320, 200)];
    [self.singlePicker setDataSource: self];
    [self.singlePicker setDelegate: self];
    [self.singlePicker setShowsSelectionIndicator: YES];
    [self.view addSubview: self.singlePicker];

    UIButton * button = [UIButton buttonWithType:UIButtonTypeRoundedRect];
    [button setFrame: CGRectMake(120, 200, 100, 50)];
    [button setTitle: @"Select" forState: UIControlStateNormal];
    [button addTarget: self action: @selector(buttonClicked)
    forControlEvents: UIControlEventTouchUpInside];
    [self.view addSubview: button];
}

-(void) buttonClicked {
    NSInteger row = [self.singlePicker selectedRowInComponent: 0];
```

```

NSString * selected = [PickerData objectAtIndex:row];
NSString * title = [[NSString alloc] initWithFormat:
    @"You selected %@", selected];
UIAlertView * alert = [[UIAlertView alloc] initWithTitle:title
    message:@"Thank you for choosing." delegate:nil
    cancelButtonTitle:@"Welcome" otherButtonTitles:nil];
[alert show];
[alert release];
}

#pragma mark UIPickerViewDelegate
-(NSInteger)numberOfComponentsInPickerView:(UIPickerView *)pickerView
{
    return 1;
}
-(NSInteger)pickerView:(UIPickerView *)pickerView
    numberOfRowsInComponent:(NSInteger)component
{
    return [PickerData count];
}
-(NSString *)pickerView:(UIPickerView *)pickerView
    titleForRow:(NSInteger)row forComponent:(NSInteger)component
{
    return [PickerData objectAtIndex:row];
}
-(void)dealloc
{
    [self单一Picker release];
    [self PickerData release];
    [super dealloc];
}

```

15/11/2013

* code in ThirdViewController.h :-

```
@interface ThirdViewController : UIViewController  
<UIPickerViewDelegate, UIPickerViewDataSource>  
{  
    UIPickerView * doublePicker;  
    NSArray * fillingTypes;  
    NSArray * breadTypes;  
}  
  
@property (nonatomic, retain) UIPickerView * doublepicker;  
_____  
_____  
_____  
-(void) buttonClicked ;  
@end.
```

* code in ThirdViewController.m :-

```
@synthesize doublePicker, fillingTypes, breadTypes;  
  
-(void) viewDidLoad  
{  
    NSArray * breadArray = [[NSArray alloc] initWithObjects:  
        @ "white", @ "whole", @ "wheat", @ "Rye", @ "Sourdough",  
        @ "Seven", @ "Grain", nil];  
  
    self.fillingType = self.breadTypes = breadArray;  
  
    [breadArray release];  
  
    NSArray * fillingArray = [[NSArray alloc] initWithObjects:  
        @ "chicken", @ "Butter", @ "Mutton", @ "Salad",  
        @ "Beef", @ "Vegetable", nil];
```

```
self.fillingTypes = fillingArray;
[fillingArray release];
self.doublePicker = [[UIPickerView alloc] initWithFrame:
CGRectMake(0, 0, 320, 250)];
[Self.doublePicker setDataSource: self];
[Self.doublePicker setDelegate: self];
[Self.doublePicker setShowSelectionIndicator: YES];
[Self.view addSubview: self.doublePicker];

UIButton *button = [UIButton buttonWithType:UIButtonTypeRoundedRect];
[button setFrame: CGRectMake(120, 300, 100, 80)];
[button setTitle:@"Select" forState: UIControlStateNormal];
[button addTarget: self action: @selector(buttonClicked)
forControlEvents: UIControlEventTouchUpInside];
[Self.view addSubview: button];
}

-(void) dealloc {
{
[doublePicker release];
[breadTypes release];
[fillingTypes release];
[super dealloc];
}
```

```
- (void) buttonClicked {
    NSInteger breadRow = [doublePicker selectedRowInComponent:0];
    NSInteger fillingRow = [doublePicker selectedRowInComponent:1];
    NSString * bread = [breadTypes objectAtIndex:breadRow];
    NSString * filling = [fillingTypes objectAtIndex:fillingRow];
    NSString * message = [[NSString alloc] initWithFormat:
        @"Your %@ on %@ bread will be right up", filling, bread];
    UIAlertView * alert = [[UIAlertView alloc] initWithTitle:@""
        Thank you for your order" message delegate:nil
        cancelButtonTitle:@"Great!" otherButtonTitles:nil];
    [alert show];
    [alert release];
    [message release];
}
```

```
# pragma mark UIPicker View Delegate.
- (NSInteger)numberOfComponentsInPickerView:(UIPickerView*)
    pickerView {
    return 2;
}
- (NSInteger)pickerView:(UIPickerView*)pickerView
    titleForRow:(NSInteger)rowForComponent:(NSInteger)component
{
    if (component == 0)
        return [[self.breadTypes objectAtIndex:rowForComponent]];
```

```

    else
        return [self.fillingType objectAtIndex:row];
}

-(NSString *)pickerView:(UIPickerView *)pickerView titleForRow:
    (NSInteger) row forComponent:(NSInteger) component.

{
    if (component == 0)
        return [self.breadTypes objectAtIndex:row];
    else
        return [self.fillingTypes objectAtIndex:row];
}
@end.

```

16/11/2013

- * Example on UIPickerView with multiple components or two components.
- * Create a single view app with the name MobilePickerApp.
- * Code in View Controller.h :-
- @ Interface ViewController : UIViewController < UIPickerViewDelegate,
UIPickerViewDataSource>.
- {
 UIPickerView *doublePicker;
 ~~UIPickerView *~~ mobileType;
 NSArray *mobileBrand;
 NSMutableArray *mobileType;
 NSMutableDictionary *mobileDictionary;
}

```
@property (nonatomic, retain) UIPickerView *doublePicker;
" "
" "
" "
-(void) ButtonClicked,
```

```
NSMutableArray *mobileType;
NSMutableDictionary *mobileDictionary;
NSArray *mobileBrand;
```

```
- (void) ButtonClicked,
```

```
- @end.
```

* Code in ViewController.m :-

```
@synthesize doublePicker, mobileDictionary, mobileType,
mobileBrand;
```

```
- (void) viewDidLoad,
```

```
{
```

```
mobileBrand = [[NSArray alloc] initWithObjects: @"Apple",
@"Samsung", @"LG", nil];
```

```
mobileType = [[NSMutableArray alloc] init];
```

```
mobileDictionary = [[NSMutableDictionary alloc] init];
```

```
NSArray *appleType = [[NSArray alloc] initWithObjects:
```

```
@"iPhone 4", @"iPhone 4S", @"iPhone 5", @"iPhone 5C",
@"iPhone 5S", nil];
```

```
[mobileDictionary setValue: [NSArray arrayWithArray: appleType];
forKey: @"Apple"];
```

```
NSArray *samsungType = [[NSArray alloc] initWithObjects:
```

```
@"sami", @"sam2", @"Sam3", @"Sam4", nil];
```

```
[mobileDictionary setValue: [NSArray arrayWithArray:
samsungType forKey: @"Samsung"]];
```

```
NSMutableArray * lgType=[[NSMutableArray alloc] initWithObject:@“LG1”,
@“LG22”, @“LG33”, nil];
[ mobileDictionary setValue:[NSMutableArray arrayWithArray:lgType]
forKey:@“LG”];
mobileType=[mobileDictionary objectForKey:@“Apple”];
self.doublePicker=[[UIPickerView alloc] initWithFrame:
CGRectMake(0, 0, 320, 180)];
[self.doublePicker setDataSource:self];
[self.doublePicker setDelegate:self];
[self.doublePicker setShowsSelectionIndicator:YES];
[self.ο view addSubview:self.doublePicker];
UIButton * button=[[UIButton buttonWithType:UIButtonTypeRoundedRect];
button setFrame:CGRectMake(120, 300, 100, 50)];
[button setTitle:@“Select” forState:UIControlStateNormal];
[button addTarget:self action:@selector(buttonClicked)
forControlEvents:UIControlEventTouchUpInside];
[self.view addSubview:button];
}
```

```

-(void)buttonClicked {
    NSUInteger selectedBrandRow = [self.doublePicker selectedRow-
        InComponent:0];
    NSUInteger selectedTypeRow = [self.doublePicker selectedRowIn
        Component:1];

    NSString * message = [[NSString alloc] initWithFormat :
        @"Your order %@ and mode %@", brand, type];
    UIAlertView * alert = [[UIAlertView alloc] initWithTitle:@"Than-
        k you for your order" message:message delegate:nil
        cancelButtonTitle:@"Great" otherButtonTitles:nil];
    [alert show];
    [alert release];
    [alert message release];
}

-(void)dealloc {
    [mobileBrand release];
    [mobileType release];
    [doublePicker release]; [mobileDictionary release];
    [super dealloc];
}

# pragma -mark UIPickerView Delegate

-(NSInteger)numberOfComponentsInPickerView:(UIPickerView *)pickerView {
    return 2;
}

```

```

-(NSInteger) pickerView:(UIPickerView *) pickerView numberOfRowsInSection component:
    NSInteger component
{
    if (component == 0)
        return ([mobileBrand count]);
    else
        return ([mobileType count]);
}

-(NSString *) pickerView:(UIPickerView *) pickerView titleForRow:
    NSInteger row forComponent: NSInteger component
{
    if (component == 0)
        return ([mobileBrand objectAtIndex:row]);
    else
        return ([mobileType objectAtIndex:row]);
}

-(void) pickerView:(UIPickerView *) pickerView didSelectRow:
    NSInteger row inComponent: NSInteger component
{
    if (component == 0)
    {
        NSString * selectedBrand = [mobileBrand objectAtIndex: row];
        mobileType = [mobileDictionary objectForKey: selectedBrand];
        [pickerView selectRow: 0 inComponent: 1 animated: YES];
        [pickerView reloadComponent: 1];
    }
}
@end

```

* UITouch :-

- whenever the user required to handle touch events of end user on device then following methods will be called. i.e.
 - ① touchesBegan
 - ② touchesEnded
 - ③ touchesCancelled.
 - ④ touchesMoved.
- When initially touches are started on device then Began function will called, when it is end then Ended function will be called.

* Create a singleView application with the name UITouchApp & place an image to the project.

* Code in ViewController.h :-

@ interface ViewController : UIViewController.

```
{  
    UIImageView *imageview;  
    UILabel *textlabel;  
}
```

@ property (nonatomic, retain) UIImageView *imageview;

@ property (nonatomic, retain) UILabel *label;

@ end

* code in ViewController.m :-

```
@synthesize imageView,.textLabel;  
-(void) viewDidLoad  
{  
    textLabel = [[UILabel alloc] initWithFrame:CGRectMake(0, 20,  
            320, 20)];  
    textLabel.text = "View Touch";  
    textLabel.tag = 1;  
    textLabel.userInteractionEnabled = TRUE;  
    [textLabel setBackgroundColor:[UIColor clearColor]];  
    [textLabel setTextAlignment:NSTextAlignmentCenter];  
    [self.view addSubview:textLabel];  
  
    imageView = [[UIImageView alloc] initWithImage:[UIImage  
        imageNamed:@"1.png"]];  
    [imageView setFrame:CGRectMake(50, 50, 75, 75)];  
    [imageView setUserInteractionEnabled:TRUE];  
    [imageView setTag:2];  
    [self.view addSubview:imageView];  
}  
-(void) touchesBegan:(NSSet*) touches withEvent:(UIEvent*) event  
{  
    NSLog(@"Touches Began");  
    UITouch * touch = [event allTouches] anyObject];
```



```
- (void) touchesMoved : (NSSet *) touches withEvent :  
    (UIEvent *) event  
{  
    NSLog (@"Touches Moved");  
    UITouch * touch = [event allTouches] anyObject];  
    CGPoint touchLocation = [touch locationInView: self.view];  
    [textLabel setText: [NSString stringWithFormat: @"%@\n" location  
        x = y = 2f, y = y = 2f", touchLocation.x, touchLocation.y]];  
    if ( touchLocation.x > 240)  
        touchLocation.x = 240;  
    if ( touchLocation.y > 375)  
        touchLocation.y = 375;  
    [imageView setFrame: CGRectMake (touchLocation.x,  
        touchLocation.y, imageView.frame.size.width,  
        imageView.frame.size.height)];  
}  
- (void) touchesCancelled : (NSSet *) touches withEvent :  
    (UIEvent *) event  
{  
    NSLog (@"Touches Cancelled");  
}  
@end
```

18/11/2013

UI Gesture & Recognizer :-

- Generally this class will provide specific event for user touch options i.e. tapping or rotation or sweep swipe related events.
- * Create a single view app with the name UIGestureApp for iPad device.
- Place an image to the project with the name 1.png, 2.png, 3.png.
- Create a new class with the name FingerRotation subclass of type UIGestureRecognizer.

* code in View Controller.h :-

```
@interface ViewController : UIViewController < UIGestureRecognizerDelegate>
{
    UIImageView *image1;
    NSMutableArray *images;
    UITapGestureRecognizer *tap;
    UISwipeGestureRecognizer *swipL;
    UISwipeGestureRecognizer *swipR;
    UIRotateGestureRecognizer *rotate;
}

@property (nonatomic, retain) UIImageView *image1;
____ " _____ NSMutableArray *images;
____ " _____ UITapGestureRecognizer *tap;
____ " _____ UISwipeGestureRecognizer *swipL;
____ " _____ *swipR;
____ " _____ UIRotateGestureRecognizer *rotate;

@end
```

* Code in ViewController.m :-

```
@synthesize images, imageView, tap, swipeL, swipeR, rotate;
short int imageIndex;

-(void)viewDidLoad
{
    images = [[NSMutableArray alloc] initWithObjects: @"1.png",
              @"2.png", @"3.png", nil];
    imageView = [UIImageView alloc] initWithFrame: CGRectMake
        (200, 250, 300, 300)];
    [imageView setImage: [UIImage imageNamed: [images objectAtIndex
        imageIndex = 0]]];
    [self.view addSubview: imageView];
    [imageView setUserInteractionEnabled: YES];
    tap = [[UITapGestureRecognizer alloc] initWithTarget: self action:
        @selector(tapped:)];
    [tap setNumberOfTapsRequired: 2];
    [imageView addGestureRecognizer: tap];
    // [tap setDelegate: self];
    swipeL = [[UISwipeGestureRecognizer alloc] initWithTarget: self
        action: @selector(swipedLeft:)];
    [swipeL setDirection: UISwipeGestureRecognizerDirectionLeft];
    [imageView addGestureRecognizer: swipeL];
    // [swipeL setDelegate: self];
}
```

```
swipeR = [[UZSwipeGestureRecognizer alloc] initWithTarget: self  
action:@selector(swiped:)];  
[swipeR setDirection: UZSwipeGestureRecognizeDirectionRight];  
[imageView addGestureRecognizer: swipeR];  
  
rotate = [[UIRotationGestureRecognizer alloc] initWithTarget: self  
action:@selector(rotating:)];  
[imageView addGestureRecognizer: rotate];  
  
}  
  
# pragma - make UITapGestureRecognizer  
-(void) tapped:(UITapGestureRecognizer *)recognizer  
{  
    static BOOL status = NO;  
    NSLog(@"Tapgesture Recognized");  
    UIView * view = [recognizer view];  
    if (status == NO)  
    {  
        [view setTransform: CGAffineTransformMakeRotation(35)];  
        status = YES;  
    }  
    else  
    {  
        [view setTransform: CGAffineTransformMakeRotation(0)];  
        status = NO;  
    }  
}
```

```
# pragma -mark UIISwipeGesture.  
-(void)swipedUp : (UIISwipeGestureRecognize*) recognizer  
{  
    NSLog(@"Inside swiped");  
    UIView * view = [recognizer view];  
    [view setTransform: CGAffineTransformMakeRotation(0)];  
    if (imageIndex > 0)  
    {  
        --imageIndex;  
        [imageView setImage: [images objectAtIndex: imageIndex]];  
    }  
}  
-(void)swipedLeft : (UIISwipeGestureRecognize*) recognizer  
{  
    NSLog(@"Inside swiped");  
    UIView * view = [recognizer view];  
    [view setTransform: CGAffineTransformMakeRotation(0)];  
    if (imageIndex < 0)  
    {  
        ++imageIndex;  
        [imageView setImage: [images objectAtIndex: imageIndex]];  
    }  
}  
# pragma -mark.  
-(void)rotating : (UITapGestureRecognizer*) recognizer
```

```

#pragma -mark UI Rotation Gesture

-(void)rotating:(UIRotationGestureRecognizer*)recognizer
{
    UIImageView* view = [recognizer view];
    [imageView setTransform: CGAffineTransformRotate
    ([view transform], 0.8)];
}

-(void)dealloc
{
    [imageView release]; [tap release];
    [images release]; [swipe release];
    [super dealloc]; [swipeR release]; [rotate release];
}
@end.

```

* code in FingerRotation.h:-

```

@interface FingerRotation: UIGestureRecognizer
{
}

@property (nonatomic, assign) CGFloat rotation;
@end.

```

* code in FingerRotation.m:-

```

#import <UIKit/UIGestureRecognizerSubclass.h>
@synthesize rotation = rotation;

```

```
- (void) touchesBegan: (NSSet*) touches withEvent: (UIEvent*) event
{
    if ([event touchesFor GestureRecognizer : self] count) > 1
    {
        self. setstate : UIGestureRecognizerStateFailed];
    }
}

-(void) touchesMoved : (NSSet*) toucheswithEvent: (UIEvent*) event
{
    if ([self state] == UIGestureRecognizerState Possible)
    {
        [self setState : UIGestureRecognizerStateBegan];
    }
    else
    {
        [self setState : UIGestureRecognizerStateChanged];
    }

    UITouch * touch = [touches anyObject];
    UIView * view = [self, view];

    CGPoint center = CGPointMake(CGRectGetMidX([view bounds]),
                                 CGRectGetMidY([view bounds]));
    CGPoint currentTouchPoint = [touch locationInView : view];
    CGPoint previousTouch Point = [touch previousLocationInView : view];
}
```

```
CGFloat angleInRadians = atan2f(currentTouchPoint.y - center.y,
currentTouchPoint.x - center.x) - atan2f(previousTouchPoint.y
- center.y, previousTouchPoint.x - center.x);

[self setRotation:angleInRadians + .02];

}

-(void)touchesEnded:(NSSet*)touches withEvent:(UIEvent*)event
{
    if ([self state] == UIGestureRecognizerStateChanged)
    {
        [self setState:UIGestureRecognizerStateFailed];
    }
    else
    {
        [self setState:UIGestureRecognizerStateFailed];
    }
}

-(void)touchesCancelled:(NSSet*)touches withEvent:(UIEvent*)event
{
    [self setState:UIGestureRecognizerStateFailed];
}

@end.
```

19/11/2013

* UIImagePickerController :-

- This UIViewController will provide, to access device camera and image gallery.
- When we are working with this UIViewController, it works with the help of navigation view.
- UIImagePickerController having the delegate methods to handle the action.
- UIImagePickerController having a property called source type which allows to take proper source type according to requirement.
- Source type can be photo library or camera or saved photo albums.
- UIImagePickerController having image quality type that can be high, medium or low.
- From iPhone 4 onwards we are having one more quality i.e. quality type 640×480 .
- From iPhone 5 onwards we are having another two quality types i.e. QualityTypeIFrame 1280×720 , QualityTypeIFrame 960×540 .
- UIImagePickerController camera capture mode can be two types i.e ModePhoto or ModeVideo.

- UIImagePickerController camera device can be two types i.e. Rear or Front.
 - UIImagePickerController camera flash mode can be three type → off, on, auto mode.
- xox —

* Create a view based app? with the name PhotoApp.

* Code in Viewcontroller.h :-

@ interface ViewController : UIViewController < UIImagePickerControllerDelegate, UINavigationControllerDelegate>

{

 UIImageView * imageView;
 UIButton * chooseBtn; choosePhotoBtn;
 UIButton * takePhotoBtn;

}

@ property (nonatomic, retain) UIImageView * imageView;
_____, _____ UIButton * choosePhotoBtn;
_____, _____ UIButton * takePhotoBtn;

@ -(void) getPhoto:(id)sender

@ end.

* Code in Viewcontroller.m :-

@ synthesize imageView, choosePhotoBtn, takePhotoBtn;

- (void) viewDidLoad

{

 [self.view setBackgroundColor: [UIColor greenColor]];

```
imageView = [[UIImageView alloc] initWithFrame: CGRectMake(20, 20, 150, 150)];
```

```
[self.view addSubview: imageView];
```

```
choosePhotoBtn = [UIButton buttonWithType:UIButtonTypeCustom  
RoundedRect];
```

```
[choosePhotoBtn setFrame: CGRectMake(20, 270, 100, 40)];
```

```
[choosePhotoBtn setTitle: @"Choose Photo"  
Normal];
```

```
[self.view addSubview: choosePhotoBtn];
```

```
[choosePhotoBtn addTarget: self action: @selector(getPhoto:  
forControlEvents: UIControlEventTouchUpInside];
```



```
takePhotoBtn = [UIButton buttonWithType:UIButtonTypeCustom  
RoundedRect];
```

```
[takePhotoBtn setFrame: CGRectMake(160, 270, 100, 40)];
```

```
[takePhotoBtn setTitle: @"Take Photo"  
Normal];
```

```
[takePhotoBtn addTarget: self action: @selector(getPhoto:  
forControlEvents: UIControlEventTouchUpInside];
```

```
[takeP self.view addSubview: takePhotoBtn];
```

```
}
```

```
-(void) get Photo: (id) sender
```

```
{
```

```
UIButton * bt = (UIButton *) sender
```

```
UIImagePickerController * picker = [[UIImagePickerController alloc] init];
```

```

picker.delegate = self;
if (bt == choosePhotoBtn)
{
    NSLog(@"choosePhoto");
    picker.sourceType = UIImagePickerControllerSourceTypeSavedPhotoAlbum;
}
else if (!UIImagePickerController.isSourceTypeAvailable(
    UIImagePickerControllerSourceTypeCamera))
{
    UIAlertView * alert=[[UIAlertView alloc] initWithTitle:
    "Alert" message:@ "No camera" delegate: self
    cancelButtonTitle:@ "OK" otherButtonTitles:nil];
    [alert show];
    NSLog(@"No camera");
    picker.sourceType = UIImagePickerControllerSourceTypePhotoLibrary;
}
else
{
    picker.sourceType = UIImagePickerControllerSourceTypeCamera;
}
[ self presentViewController:animated:YES completion:nil];
}

#pragma mark UIImagePickerController
-(void)imagePickerController:(UIImagePickerController *)picker didFinishPickingMediaWithInfo:(NSDictionary *)info
{
    [[picker presentingViewController] dismissViewControllerAnimated:YES completion:nil];
}

```

```
imageView.image = [info objectForKey:@"UIImagePickerControllerOriginalImage"];
```

}.

@end.

* Camera action related sample link (only images):

developer.apple.com/library/ios/samplecode/PhotoPicker/Introduction/intro.html.

* AVCam example which is used for recording the audio & video which is available from iOS 6.0 or later. (it works on Xcode 5.0).

developer.apple.com/library/ios/samplecode/AVCam/Introduction/intro.html # apple_ref/doc/uid/DT84001012

More samples we can get from following link.

developer.apple.com/library/ios/navigation/#section=Flexibility&topic=Resource%20Types&Topic=Sample%20Code.

(MWPhotoBrowser)

* MWPhotoBrowser-

20/11/2013

- It is an open source third party iOS appl' which allows to work with images.
- By using this appl' we can implement photoBrowser appl' which having the functionality of zooming, pinching & swipping & more functionality on images.
- When we are working with this appl' which is allow to modify or update any source code related files.
- Download MWPhotoBrowser from Github server.
(multiple versions are available with tag values)
- When we extract the MWPhotoBrowser zip file, then it will provide MWPhotoBrowser master directory with demo & library files.
- All library files of MWPhotoBrowser are available in MWPhotoBrowser directory.

* Procedure to add MWPhotoBrowser framework to current appl'.

- * Create a new iOS appl' with Master-Detail appl' template & give the product name as PhotoGallery.

- Create a new group in app' with the name MwPhotoBrowser
 - Drag MwPhotoBrowser.bundle into MwPhotoBrowser group - or add this bundle file to current project.
 - Copy the classes related all files of MwPhotoBrowser into the current project.
 - Copy the library related all files into current app?
 - Verify that all MwPhotoBrowser related files need to be compiled.
 - Go to project template → select Build Phase & check in compile sources.
 - Add the new frameworks to the current project i.e ImageIO framework & MessageUI framework.
 - Select project template → Go to Build Phase → link binary with libraries & add required fw.
- * Create a new group with the name called photos and place some images into photos group.

* Code in MasterViewController.h :-

```
#import "MwPhotoBrowser.h"
@interface MasterViewController : UITableViewController
< MwPhotoBrowserDelegate >
@property (strong, nonatomic) MwPhotoBrowser *detailViewController;
@property (nonatomic, retain) NSMutableArray *photoList;
@end
```

* Code in Master View Controller.m :-

```
@synthesize photoList;

-(id)initWithNibName:(NSString *) nibNameOrNil
{
    self.title = NSLocalizedString(@"Gallery", @Master);
}

-(void)dealloc
{
    [-detailviewController release];
    [-Objects release];
    [photoList release];
    [super dealloc];
}

-(void)viewDidLoad
{
    # warning remove edit & add button
    self.navigationController.navigationBar.tintColor = [UIColor
        blackColor];
    [self.tableView setBackgroundColor: [UIColor grayColor]];
    _Objects = [NSMutableArray alloc] initWithObjects: @"photo1.png",
    @"photo2.png", @"photo3.png", @"photo4.png",
    -- -- -- -- @"photo5.png", nil];
}
```

```
photolist = [[NSMutableArray alloc] init];
MWPhoto * photo = nil;
for (int i=0; i<[_objects count]; i++)
{
    UIImage * image = [UIImage imageNamed: [_objects
        objectAtIndex:i]];
    Photo = [[MWPhoto alloc] initWithImage:image];
    photo.caption = [_objects objectAtIndex:i];
    [photolist addObject:photo];
}
```

pragma mark TableView

```
- (CGFloat)tableView: (UITableView *)tableView heightForRowAtIndexPath: _____
{
    return 130.0f;
}

-(UITableViewCell *)tableView: (UITableView *)tableView
cellForRowAtIndexPath: (NSIndexPath *) indexPath
{
    same as previous cell designing code
}

UIView * cellView = [[UIView alloc] initWithFrame:
    CGRectMake(20, 20, 250, 150)];
```

```
UIImageView * imageView = [[UIImageView alloc] initWithFrame:CGRectMake(0, 0, 220, 80)];  
UIImage * image = [UIImage imageNamed:[_objects objectAtIndex:indexPath.row]];  
[imageView setImage:image];  
[cellView addSubview:imageView];  
  
UILabel * imageLabel = [[UILabel alloc] initWithFrame:CGRectMake(80, 85, 100, 20)];  
[imageLabel setText:[_objects objectAtIndex:indexPath.row]];  
[cellView addSubview:imageLabel];  
[cell addSubview:cellView];  
return cell;  
}  
  
-(void)tableView:(UITableView *)tableView didSelectRowAtIndexPath:  
IndexPath:—  
{  
if (!self.detailViewController)  
{  
self.detailViewController = [[MWPPhotoBrowser alloc]  
initWithDelegate:self] autorelease;  
}  
[self.detailViewController setInitialPageIndex:indexPath.row];  
[self.navigationController pushViewController:self.  
detailViewController animated:YES];
```

```

#pragma mark - MWPhotoBrowserDelegate

-(NSUInteger)numberOfPhotosInPhotoBrowser:(CMWPhotoBrowser
*)photoBrowser
{
    return photoList.count;
}

-(MWPhoto *)photoAtIndex:(NSUInteger)index
{
    if (index < photoList.count)
        return [photoList objectAtIndex:index];
    return nil;
}

@end.

```

21/11/2013

* Working with document directory & file Systems:-

- When we are installing any app in iOS devices then automatically it creates a unique directory name which is local to that specific app.
- Generally when we are developing the app it will creates a unique directory in following location.

/user/Anurag/Library/Application Support/iphone Simulator/6.1/Applications

- 6.1 is indicating simulator version & within the appl' directory 'n' number of app related subdirectories are available.
- The structure of appl' directory is documents, library & temp.
- All app related files always we need to store in 'documents' directory only.

* Syntax to get documents directory location :-

```
NSMutableArray * paths = NSSearchPathForDirectoriesInDomains
```

```
(NSDocumentDirectory, NSUserDomainMask, YES);
```

```
Nsstring * documentsPath = [paths objectAtIndex:0];
```

// get documents path

OR

```
Nsstring * documentDirectory = [NSHomeDirectory()
```

```
stringByAppendingPathComponent:@ "Documents"];
```

- When we required to handle any file into document directory, then we required to create object of type NSFileManager.

Syntax:- NSfileManager * filemgr ;

```
filemgr = [NSfileManager defaultManager];
```

- After creating the instance of fileManager, with the help of this instance we can create any files for reading or writing.

* Syntax to create a text file and writing some data :-

NSString * documentDirectory = [NSHomeDirectory ()

stringByAppendingPathComponent: @"Documents"];

NSString * fileName = [NSString stringWithFormat: @"%.txt",
@"my test"];

NSString * filePath = [documentDirectory stringByAppendingPathComponent: filename];

NSFileManager * filemgr = [NSFileManager defaultManager];

NSString * textString = [NSString stringWithFormat: @"%d
Hello welcome %d", 100, 200];

NSData * dataBuffer = [textString dataUsingEncoding:
NSUTF8StringEncoding];

[filemgr createFileAtPath: filePath content: dataBuffer
attributes: nil];

* Syntax to read the data from file :-

NSDocument

NSString * documents

NSArray * paths = NSSearchPathForDirectoriesInDomains

(NSDocumentDirectory, NSUserDomainMask, YES);

NSString * documentPath = [paths objectAtIndex: 0];

NSString * fileName = [NSString stringWithFormat: @"mytest.txt"];

```

    NSString * filePath = [documentsPath stringByAppendingPathComponent:filename];
    NSFileManager * fileMgr = [NSFileManager defaultManager];
    if ([fileMgr fileExistsAtPath:filePath] != YES)
    {
        NSData * dataBuffer;
        dataBuffer = [fileMgr contentsAtPath:filePath];
        NSStrong * string = [[NSStrong alloc] initWithData:dataBuffer
            encoding:NSUTF8StringEncoding] autorelease];
        NSLog(@"%@", string);
    }
    else
    {
        NSLog(@"File not found");
    }
}

```

* Syntax to compare two file's data & if both files are having same content then remove one file.

```

NSArray * path = [NSSearchPathForDirectoriesInDomains
    (NSDocumentDirectory, NSUserDomainMask, YES)];
NSStrong * documentPath = [path objectAtIndex:0];
NSStrong * file1 = [NSStrong strongWithString:@"my test1.txt"];
NSStrong * file2 = [NSStrong strongWithString:@"my test2.txt"];
NSStrong * filePath1 = [documentsPath stringByAppendingPathComponent:file1];

```

```
NSString * filePath2 = [documentsPath stringByAppendingPathComponent : file2];
```

```
NSFileManager * fileMgr = (NSFileManager *) defaultManager;
if ([fileMgr fileExistsAtPath : filePath2] == YES)
{
    if ([fileMgr fileExistsAtPath : filePath2] == YES)
    {
        NSData * dataBuffer1;
        dataBuffer1 = [fileMgr contentsAtPath : filePath2];
        NSString * string1 = [[NSString alloc] initWithData :
            dataBuffer1 encoding : NSUTF8StringEncoding] autorelease];
        NSData * dataBuffer2;
        dataBuffer2 = [fileMgr contentsAtPath : filePath2];
        NSString * string2 = [[NSString alloc] initWithData :
            dataBuffer2 encoding : NSUTF8StringEncoding] autorelease];
        if ([string2 isEqualToString : string1])
        {
            NSError * errorMsg = nil;
            if ([fileMgr removeItemAtPath : filePath2 error : &errorMsg]
                == YES)
                NSLog(@"%@", @"is deleted", filePath2);
        }
        else
            NSLog(@"both file not same");
    }
    else
        NSLog(@"%@", @"my test2.txt not found");
}
else
    NSLog(@"%@", @"my test2.txt not found");
```

* Create a new iOS app with the name NSFileManagerApp
of singleView app for iPad device.

* code in ViewController.h :-

```
@interface ViewController : UIViewController<UIAlertViewDelegate,  
UITextViewDelegate, UITextFieldDelegate>  
{  
    UITextField *filename;  
    UITextView *fileData;  
    UIButton *save;  
    UIButton *read;  
}  
  
@property (nonatomic, retain) UITextField *filename;  
@property (nonatomic, retain) UITextView *fileData;  
@property (nonatomic, retain) UIButton *save;  
@property (nonatomic, retain) UIButton *read;  
  
-(NSString *)getDocumentsPath;  
-(void)buttonClicked:(id)sender;  
@end
```

* code in ViewController.m :-

```
@synthesize filename, fileData, save, read;  
-(NSString *)getDocumentPath  
{  
    NSArray *paths = NSSearchPathForDirectoriesInDomains  
    (NSDocumentDirectory, NSUserDomainMask, YES);  
    NSString *documentsPath = [paths objectAtIndex:0];  
    return documentsPath;  
}
```

22/11/2013

- (void) viewDidLoad

{

fileData = [[UITextView alloc] initWithFrame:CGRectMake(10, 10,
200, 150)];

[fileData setDelegate:self];

[fileData setReturnKeyType:UIReturnKeyDone];

[fileData setAutoCorrectionType:UITextAutoCorrectionTypeYes];

[self.view addSubview:fileData];

[fileData setBackgroundColor:[UIColor blackColor]];

[fileData setTextColor:[UIColor whiteColor]];

[fileData setFont:[UIFont fontWithName:@"Arial" size:20.6]];

[fileData becomeFirstResponder];

fileName = [[UITextField alloc] initWithFrame:CGRectMake(240, 80,
180, 40)];

[fileName setDelegate:self];

[fileName setBorderStyle:UITextBorderStyleRoundedRect];

[fileName setReturnKeyType:UIReturnKeyDone];

[self.view addSubview:fileName];

save = [UIButton buttonWithType:UIButtonTypeRoundedRect];

[save setTitle:@"Save" forState:UIControlStateNormal];

[save addTarget:self action:@selector(buttonClicked:) forControlEvents:UIControlEventTouchUpInside];

[save setFrame:CGRectMake(20, 220, 80, 40)];

[self.view addSubview:save];

[save setEnabled:FALSE];

```

read = [UIButton buttonWithType:UIButtonTypeRoundedRect];
[read setTitle:@"Read" forState:UIControlStateNormal];
[read addTarget:self action:@selector(buttonClicked:)
    forControlEvents:UIControlEventTouchUpInside];
[read setFrame:CGRectMake(150, 220, 80, 40)];
[self.view addSubview:read];
[read setEnabled:FALSE];
}

-(NSString *)readFileData:(NSFileManager*)mng andFilePath:
(NSString*)fpath
{
    NSData *dataBuffer;
    dataBuffer = [mng contentsAtPath:fPath];
    NSString *string = [[NSString alloc] initWithData: dataBuffer
        encoding: NSUTF8StringEncoding] autorelease];
    return string;
}

-(void)writeFileData:(NSFileManager*)mng andfilePath:(NSString*)
fpath withAppendData:(NSData*)append
{
    NSString* writeData = nil;
    if ([append length] != 0)
        writeData = [NSString stringWithFormat:@"%@\n%@", fpath,
            [append description]];
}

```

```

else
    writeData = [fileData text];

NSData * databuffer = [writeData dataUsingEncoding:
    NSUTF8StringEncoding];
NSError * err;
[err createFilePath : fpath contents : databuffer attributes
    nil];
[fileData setText : @""];
[fileName setText : @""];
[save setEnabled : FALSE];
}

-(void)buttonClicked : (id)sender
{
    UIButton * bt = (UIButton *)sender;
    NSString * file = [NSString stringWithFormat : @"%@.txt",
    [filename text]];
    NSString * filePath = [[self getDocumentsPath]
    stringByAppendingPathComponent : file];
    NSFileManager * fileManager = [NSFileManager defaultManager];
    if ([[bt currentTitle] isEqualToString : @"Save"])
    {
        if ([fileManager fileExistsAtPath : filePath] == YES)
        {
            UIAlertView * alert = [[UIAlertView alloc] initWithTitle :
            @"Alert" message : @"File is Already Exists" delegate : self
            cancelButtonTitle : @"Replace" otherButtonTitles : @"Append",
            @"Cancel", nil];
            [alert show];
        }
    }
}

```

```
    else
        [self writefileData : fileMge and filePath : filePath
            withAppendData : nil];
    }

else
{
    if ([fileMge fileExistsAtPath : filePath] == YES)
    {
        [fileData setText : [self readfileData : fileMge and filePath :
            filePath]];
        [fileData setEditable : FALSE];
        [read setEnabled : FALSE];
        [fileName setText : @""];
    }
    else
    {
        UIAlertView * alert = [[UIAlertView alloc] initWithTitle :
            @"Alert" message : @"File is not Exist" delegate : nil
            cancelButtonTitle : @"OK" otherButtonTitles : @nil, nil];
        [alert show];
        [fileName setText : @""];
        [read setEnabled : FALSE];
    }
}

-(void)alertView:(UIAlertView *)alertView clickedButtonAtIndex:
    NSInteger buttonIndex.
```

```

{
    NSString * file = [NSString stringWithFormat:@"~. %@.txt",
    [fileName text]];

    NSString * filePath = [[self documentsPath] stringByAppendingPathComponent:file];

    NSFileManager * fileMgr = [NSFileManager defaultManager];
    if(buttonIndex == 0) // Replace.

    {
        if([fileMgr fileExistsAtPath:filePath] == YES)
        {
            [fileMgr removeItemAtPath:filePath error:&error];
            [self writefileData:fileMgr andFilePath:filePath
            withAppendData:nil];
        }
    }

    else if(buttonIndex == 1) // Append
    {
        NSString * cdata = [self readfileData:fileMgr andFilePath:
        filePath];
        [fileMgr removeItemAtPath:filePath error:&error];
        [self writefileData:fileMgr andFilePath:filePath
        withAppendData:cdata];
    }
    else;
}

-(BOOL) textFieldShouldReturn:(UITextField *) textField
{
    [textField resignFirstResponder];
    return YES;
}

```

```
- (void) textFieldDidBeginEditing : (UITextField *) textField
{
    [save setEnabled : FALSE];
    [read setEnabled : FALSE];
}

- (void) textFieldDidEndEditing : (UITextField *) textField
{
    if ([textField text] length) != 0 && [[fileData text] length] != 0)

    {
        [save setEnabled : YES];
        {
            else if ([textField text] length) != 0)
                [read setEnabled : YES];
            else;
        }
    }
}

#pragma -mark UITextViewDelegate

- (void) textViewDidBeginEditing : (UITextView *) textView
{
    if ([[fileName text] length] != 0 && [[textView text] length] != 0)
    {
        [save setEnabled : FALSE];
        [read setEnabled : FALSE];
    }
    else;
}

- (void) textViewDidEndEditing : (UITextView *) textView
{
    if ([[fileName text] length] != 0 && [[textView text] length] != 0)
    {
        [save setEnabled : YES];
        {
            else;
        }
    }
}
```

* Code for getting list of specific types of file from Document directory :-

```
# pragma -mark FindFileList.  
-(NSArray *) findFiles:(NSString *) extension  
{  
    NSMutableArray * matches = [[NSMutableArray alloc] init];  
    NSFileManager * fileManager = [NSFileManager defaultManager];  
    NSString * item;  
    NSArray * contents = [fileManager contentsOfDirectoryAtPath:  
        [NSTemporaryDirectory() stringByAppendingPathComponent:  
            @"/Documents"] error:nil];  
    for (item in contents)  
    {  
        if ([[item pathExtension] isEqualToString: extension])  
        {  
            [matches addObject: item];  
        }  
    }  
    return matches;  
}
```

* Saving the image file Data into document directory :-

```
-(void) SaveImage  
{  
    NSArray * paths = NSSearchPathForDirectoriesInDomains  
        (NSDocumentDirectory, NSUserDomainMask, YES);
```

```
NSString * documentsDirectory = [paths objectAtIndex:0];
NSString * savedImagePath = [documentsDirectory stringByAppendingPathComponent:@"image1.png"];
UIImage * image = imageView.image; // get the image object.
NSData * imageData = UIImagePNGRepresentation(image);
BOOL status = [imageData writeToFile:savedImagePath
    automatically:NO];
if (status == YES)
{
    NSLog(@"file is saved");
}
else
    NSLog(@"failed to save");
```

* loading the image into imageView from documents directory :-

```
- (void) getImage
{
    NSArray * paths = NSSearchPathForDirectoriesInDomains(NSDocumentDirectory, NSUserDomainMask, YES);
    NSString * getImagePath = [documentsDirectory stringByAppendingPathComponent:@"image1.png"];
    UIImage * image = [UIImage imageWithContentsOfFile:getImagePath];
```

```
NSData * imageData = [NSData dataWithContentsOfFile:  
                     [getImagePath];  
UImage * image = [UImage imageWithData:imageData];  
imageView.image = image;  
}
```

23/11/2013

* Data Persistence :- (Data Storage)

- This is a concept of storing of appl' related data in device memory
- Every appl' related data is local to that specific appl' only.
- One appl' related data cannot be access by another appl'
- In iOS, it is possible to store the data by using property sheet (Plist), SQLite DB and core Data.

* Working with Plist files :-

- Plist will store the data in the form of xml.
- In xml format, data will be stored with the help of dictionaries.
- In dictionary, data will be manipulated with the help of 'key' structures.
- Within the Plist file, data can be organised in the form of dictionary, boolean, data, date, Number and string and array type.
- In Plist, data will be store with hierarchical relationship like NSDictionary object.

* Create a single view app with the name PlistApp.

- Add a new property sheet to the project with the name Data.

file → new → file → Resource → Property list.

- Add an element with the key "name" & add another field "phone".

Name → Key → String → value → Balu.

Phone → key → Array →

<item0> → string → value → 123456 -

<item1> → " → " → 284333

<item2> → " → " → 883406

* XML structure of Plist file:-

```
<plist version="1.0">
```

```
<dict>
```

```
  <key> Name</key>
```

```
  <string> Balu </string>
```

```
  <key> Phone</key>
```

```
  <array>
```

```
    <string> 123456 </string>
```

```
    <string> 284333 </string>
```

```
    <string> 883406 </string>
```

```
  </array>
```

```
</dict>
```

```
</plist>
```

* code in ViewController.h :-

@ interface ViewController : UIViewController < UITextFieldDelegate,
UIScrollViewDelegate>

```
{  
    UITextField * nameEntered;  
    UITextField * homePhone;  
    UITextField * workPhone;        UIButton * saveButton;  
    UITextField * cellPhone;        CGPoint * offset;  
    NSString * personName;  
    NSMutableArray * phoneNumbers;  
    UITextField * textField;  
}
```

@ property (nonatomic, retain) UITextField * nameEntered ;

----- " ----- * homePhone;

----- " ----- * workPhone;

----- " ----- * cellPhone;

----- " ----- NSString * personName;

----- " ----- NSMutableArray * phoneNumbers;

----- " ----- UIButton * saveButton;

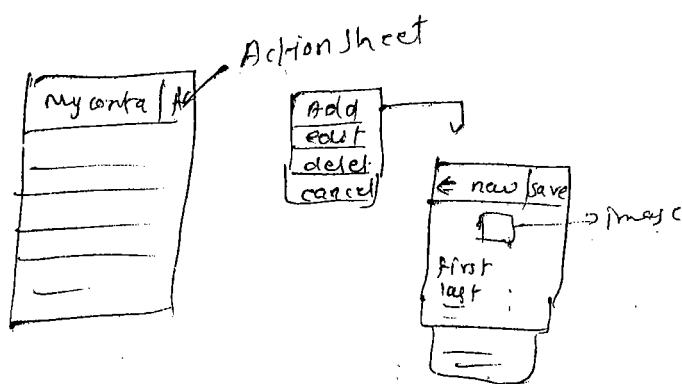
- (void) saveData; @ " ----- UIScrollView * scrollView.

@ end.

① master-Detail Appl'

② MyContacts (custom class)

- FirstName
- LastName
- cell no.
- email ID
- birthDate
- image



<http://www.facebook.com/download/1486793284878400/>

PlstApp1.zip

—XOX—

SQLite

27/11/2013

- It is a light weighted open source database which can be use for any kind of mobile OS.
- SQLite is a serverless database which cannot be use for client server architecture.
- SQLite database is required for standalone appl'.
- for any kind of iOS or OS X appl', by default we will get SQLite data engine because it is preinstalled in devices.
- SQLite database related all resources are available in www.SQLite.org.

- When we are working with SQLite database in iOS app, then we required to go for "libsqlite3.dylib" library file.
- If we required to use SQLite database we need the minimum knowledge of database related queries.

* Minimum App Setting information :-

- When we need to provide app's related information then select project template and choose summary option.
- In iOS app's target section, Bundle Identifier must be match with provisional certificate of App which is downloaded from developer account.
- Version value can be changed depends upon the requirement & build version data is beta version.
- Device type can be iPhone, iPad or universal.
- Deployment target should be proper according to available market status of devices.
- In development info section, Supported interface orientation can be anything according to requirement.
- App icons should be proper according to device-type-
 - for iPhone 67x67 pixels, for retina 114x114.
 - for iPad 72x72 pixels, for retina 144x144.
- Launch images should be place properly according to device type.
 - for iPhone - 320x480 (non-retina), 640x960 (retina).
640x1136 (4 inch screen mobiles)

- for iPad :- Portrait is 768 x 1004, for retina 1836 x 2008.
- for landscape mode : 1024 x 748, for retina 2048 x 1496.

29/11/2013

* Code for opening the SQLite Database :-

- When we are working with any SQLite related API (API Programming Interface) then it must be located from "sqlite3.h" header file.

```
#include "sqlite.h" (This header file is available if we loaded "libsqlite3.dylib" framework.)
```
- To handle sqlite database we need a pointer of type : sqlite3.

* Syntax to create SQLite Database :-

```
- (NSString *) filePath
{
    NSArray * paths = NSSearchPathForDirectoriesInDomains
        (NSDocumentDirectory, NSUserDomainMask, YES);
    NSString * documentDirectory = [paths objectAtIndex:0];
    return [documentDirectory stringByAppendingPathComponent
        : @"/EMPDATA.sqlite"];
}

// Open DB
-(void)openDB
{
    if (sqlite3_open([self filePath] UTF8String], &db) != SQLITE_OK
        SQLITE_OK
```

```

    {
        sqlite3_close(db);
        NSAssert(0, @"Database failed to open");
    }
}

```

- "sqlite3_open" is an API which creates sqlite database in specific path.
- When we require to check API execution status has a success then it returns "SQLITE_OK" constant.
- "sqlite3_close" API will close the database.

* Syntax to code for creating a table with sqlite db :-

```

-(void)createTable
{
    char *error;
    char *sql_stmt = "create table if not exists EMP (ID INTEGER PRIMARY KEY AUTOINCREMENT, FirstName
    varchar(64), LastName varchar(64), Email varchar(40),
    Salary INT INTEGER, Mobile varchar(10), Telate varchar(10)
    About varchar(100))";
    if (sqlite3_exec(db, sql_stmt, NULL, NULL, &error) != SQLITE_OK)
    {
        UIAlertView *alert = [[UIAlertView alloc] initWithTitle:@"Alert"
        message:@"Failed to Create table" delegate:nil cancelButtonTitle:@"OK"
        otherButtonTitles:nil];
        NSLog(@"Failed to create table : %s", sqlite3_errmsg(db));
        sqlite3_close(db);
        sqlite3_free(error);
        [alert show];
        exit(1);
    }
}

```

```
else  
    NSLog(@"Table is created");  
}
```

- When we required to execute any query statement of SQL then we need to use "sqlite3_exec" API.
- At the time of executing any query if we got any error that error message can be find by using "sqlite3_errmsg" API.
- If we required to delete any dynamic data which is created by sqlite3 then we can use "sqlite3_free" API.

* code for inserting a record in table :-

```
-(void)insertData:(NewRecordForm *)data  
{  
    char *error;  
    SQLite_stmt *statement;  
    NSString *str=[NSString stringWithFormat:@"insert into  
EMP (FirstName, LastName, Email, salary, Mobile, Jdate, About)  
values ('%@', '%@', '%@', %i, '%@', '%@',  
'%@')", data.FirstNameValue, data.LastNameValue,  
data.emailValue, [data salValue] int-value],  
data.pnoValue, data.joinDate, data.about];  
    char *sql_stmt=(char*)calloc([str length], sizeof(char));  
    strcpy(sql_stmt, [str UTF8String]);
```

```

    sqlite3_prepare_v2(db, sql_stmt, -1, &statement, NULL);
    if(sqlite3_exec(db, sql_stmt, 0, 0, &error) == SQLITE_OK)
    {
        UIAlertView *alert = [[UIAlertView alloc] initWithTitle:@"Alert"
                                                       message:@"New Record Inserted" delegate:nil cancelButtonTitle:@"OK" otherButtonTitles:nil];
        [alert show];
    }
    else
    {
        UIAlertView *alert = [[UIAlertView alloc] initWithTitle:@"Alert"
                                                       message:@"Failed to insert Data" delegate:nil
                                                       cancelButtonTitle:@"OK" otherButtonTitles:nil];
        NSLog(@"Error: %s", sqlite3_errmsg(db));
        sqlite3_free(error);
        [alert show];
    }
    free(sql_stmt);
}

```

* Code for extracting the data from Table with Specific column values :-

-(void) getData.

```

{
    NSMutableArray *objectList = [NSMutableArray alloc] init];

```

```
[self openDB];
char *error;
char *query_stmt = "select * from EMP";
sqlite3_stmt *statement;
sqlite3_prepare_v2(db, query_stmt, -1, &statement, NULL);
if (sqlite3_exec(db, query_stmt, 0, 0, &error) == SQLITE_OK)
{
    while (sqlite3_step(statement) == SQLITE_ROW)
    {
        NSString *data = [NSString stringWithFormat:@"%.3f %.3f %.3f",
                           sqlite3_column_text(statement, 0),
                           sqlite3_column_text(statement, 1),
                           sqlite3_column_text(statement, 2)];
        NSLog(@"%@", data);
        [objectList addObject:data];
    }
    sqlite3_finalize(statement);
}
else
{
    NSLog(@"Error : %@", sqlite3_errmsg(db));
    sqlite3_free(error);
}
```

- sqlite3_prepare_v2 API will provide the rollback (cancel) option for upcoming executable query statement, if we are not using this API then directly data will be committed into table.
- "select.* from EMP" query will return bundle of the data at a time, from this bundle data if we required to read row by row content, then we need to use "sqlite3_step" API.
- If this API has selected the row data, then we will get SQITE_ROW return value
- From selected row data, if specific column content if we need to extract then use "sqlite3_column_text" API
- When we are using sqlite3_column_text API, then we need to provide column Index.
- By using "sqlite3_finalize" API, we can delete previous query related heap data.

* Code for displaying specific record according to user selected row.

```
-(void) tableView: (UITableView*) tableView didSelectRowAtIndexPath: (NSIndexPath*) indexPath {
    NSString * data = [objectList objectAtIndex:indexPath.row];
    NSRange range = [data rangeOfString:@""];
    NSString * tempID = [data substringWithRange:NSMakeRange(0, range.location)];
    unsigned int ID = [tempID intValue];
    sqlite3_stmt * statement;
    NSString * query = [NSString stringWithFormat:@"SELECT FirstName, LastName, Email, Salary, Mobile, Date, About FROM EMP WHERE ID = '%.4", ID];
    char * error;
    char * query_stmt = (char*) malloc([query length] * sizeof(char));
    strcpy(query_stmt, [query UTF8String]);
    sqlite3_prepare_v2(db, query_stmt, -1, &statement, NULL);
    if(sqlite3_exec(db, query_stmt, 0, 0, &error) == SQLITE_OK) {
        if(sqlite3_step(statement) == SQLITE_ROW) {
            [form setFirstNameValue:[NSString stringWithFormat:@"%@", sqlite3_column_text(statement, 0)]];
        }
    }
}
```

```
[ form.setLastNameValue: [NSString stringWithFormat:@"%@",  
    sqlite3_column_text(statement, 1)]];
```

```
[ form.setEmailValue: _____  
    _____ [statement, 2]]];
```

```
[ form.setSalValue: _____  
    _____, _____, 3]]];
```

```
[ form.setPnoValue: _____  
    _____, _____, 4]]];
```

```
[ form.setJointDate: _____  
    _____, _____, 5]]];
```

```
[ form.setAbout: _____  
    _____, _____, 6]]];
```

5.

else

```
{ NSLog(@"%@", @"No rows are selected");
```

```
}
```

```
sqlite3_finalize(statement);
```

```
}
```

else

```
{ fprintf(stderr, "\nError: %s", sqlite3_errmsg(db));
```

```
sqlite3_free(cello);
```

```
}
```

```
free(query_stmt);
```

```
detailViewController * detailData = [[detailViewController alloc  
init withFormData: self.form] autorelease];
```

```
[self.navigationController pushViewController: detailData  
animated: YES];
```

```
}
```

* code for updating data in table :-

int EMPID; // selected row id from last value.

-(void)onUpdateTouched:(id)sender;

{ if (self.isValid == YES)

{ sqlite3_stmt *statement;

[self openDB];

NSString *querySQL = [NSSString stringWithFormat:

@UPDATE EMP SET FirstName = %@", lastName = %@",

Email = %@", salary = %.i, Mobile = %@", Jdate =

@"%@", About = %@", WHERE ID = %@",

self.form.FirstNameValue;

self.form.LastNameValue;

self.form.EmailValue;

[self.form.salaryValue intValue],

self.form.PHOValue,

self.form.joinDate,

self.form.about, EMPID];

const char *update_stmt = [querySQL UTF8String];

if (sqlite3_prepare_v2(db, update_stmt, -1, &statement, NULL) == SQLITE_OK)

{ // sqlite3_bind_text(statement, 1, update_stmt, -1,

SQLITE_TRANSIENT);

if (sqlite3_step(statement) == SQLITE_DONE)

{ NSLog(@"%@", @"updated");

}

```

    else
    {
        NSLog(@"Unable to update");
    }
    sqlite3_finalize(statement);
}
sqlite3_close(db);
}

```

[self.navigationController pushViewController Animated:YES];

* code for deleting the data from table:-

```

int EMPID;
-(void)onDeleteTouched:(id)sender;
{
    sqlite3_stmt *statement;
    [self openDB];
    NSString *querySQL = [NSString stringWithFormat:
                           @"DELETE FROM EMP WHERE ID=%d", EMPID];
    const char *update_stmt = [querySQL UTF8String];
    if (sqlite3_prepare_v2(db, update_stmt, -1, &statement, NULL)
        == SQLITE_OK)
    {
        //sqlite_bind_text(statement, 1, update_stmt, -1, SQLITE_TRANSIENT);
        if (sqlite3_step(statement) == SQLITE_DONE)
        {
            NSLog(@"Deleted");
        }
    }
    else
    {
        NSLog(@"unable to delete");
    }
}

```

```
    sqlite3_finalize(statement);  
}  
sqlite3_close(db);  
(self.navigationController.popViewControllerAnimated:YES];  
}
```

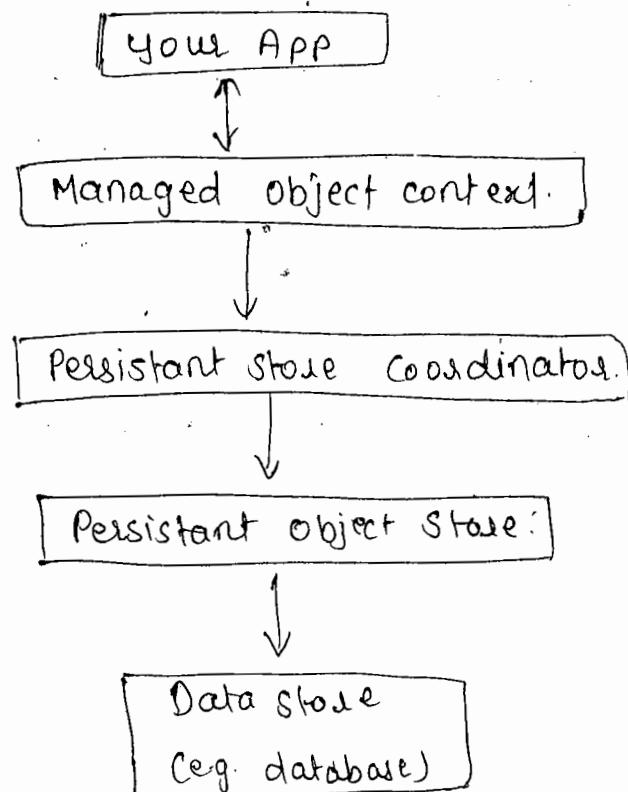
—XOX—

* Working with Core Data :-

30/11/2013

- In iOS data storage is called as Persistent data.
- When we required to work with any database programming, we use the database like Oracle, MS-SQL or MySQL which provides client-server architecture.
- By using SQLite we can provide database for standalone app.
- These all 4 types of databases will work with the help of RDBMS rules.
- According to RDBMS, data required to store in the form of table, row & column format and which allow to manipulate by using SQL queries.
- When we are working with iOS app, if we don't have any query knowledge, then we can use core data.

- Core data is not a database, because it doesn't work with the help of RDBMS rules.
- Core data is a framework which allows us to store the data using inbuilt database i.e. SQLite.
- Core data is not a relational database because it allows to store the data in object oriented format.
- When we are working with core data we need to manage following components properly i.e.
 - 1) Managed object Model.
 - 2) Persistent store Coordinator.
 - 3) Managed object context.
- Core data app's related structure is



* Managed Object Model :-

It describes the schema that you use in the app. If you have a database background, think of this as a database schema. However, the schema is represented by a collection of objects (also known as entities). In Xcode, the managed object Model is defined in a file with the extension .xcdatamodeld. You can use the visual editor to define the entities and their attributes, as well as relationships.

* Persistent Store Coordinator :-

SQLite is the default persistent store in iOS. However core data allows developers to setup the multiple stores containing different entities. The persistent store coordinator is the party responsible to manage different persistent object stores and save the objects to the store. Forget about it if you don't understand what it is. You will not interact with persistent store coordinator directly when using core data.

* Managed object Context :-

Think of it as a "scratch pad" containing objects that interacts with data in persistent store.

Its job is to manage object created and returned using core data. Among the components in the core data stack, the managed object context is the one you'll work with for most of the time.

In general, whenever you need to fetch and save objects in persistent store, the context is the first component you'll talk to.

—xox—

* Create an empty iOS app with enable of core data and autoreference count option with CoreDataApp.

* Creating Managed Object Model :-

- Open CoreDataApp.xcode modeled file.
- click on add entity button & give the name as EMP (entity name is treated as a table).
- Add attributes to EMP entity with the name ~~empid~~^{String} empid, empfname (string type) & emplname (string type).
- Add a new objective c class with the name ListViewController subclass of type UITableViewController.
- (without sub file)

* code for saving the data in context :-

```
-(void) saveButton: (id) sender  
{  
    NSManagedObjectContext * context = [self managedObjectContext];  
    NSManagedObject * newRecord = [NSEntityDescription  
        insertNewObjectForEntityForName: @"EMP"  
        inManagedObjectContext: context];  
  
    [newRecord setValue: self.idtextfield.text forKey: @"empid"];  
    [newRecord setValue: self.fnametextfield.text forKey: @"empname"];  
    [newRecord setValue: self.lnametextfield.text forKey: @"empname"];  
  
    NSError * error = nil;  
    // save the object to persistent store.  
    if ([context save: & error])  
    {  
        NSLog(@"%@", error, [error localizedDescription]);  
    }  
  
    [self.navigationController popViewControllerAnimated: YES];  
}
```

-End/Exn

* code for fetching the data from context :-

```
NSManagedObjectContext * managedObjectContext =  
[self managedObjectContext];  
NSFetchRequest * fetchRequest = [[NSFetchRequest alloc]  
initWithEntityName:@"EMP"];  
self.recordList = [[managedObjectContext executeFetchRequest:  
fetchRequest error:nil] mutableCopy];
```

```
NSManagedObject * empObject = [self.recordList objectAtIndex:  
indexPath.row];
```

```
[cell.textLabel setText:[NSString stringWithFormat:@"%@.%@.  
%@ %@", empObject valueForKey:@"empId"], [empObject  
valueForKey:@"emp fname"], [empObject valueForKey:  
@"emp lname"]];
```

* code for deleting the data :-

```
NSManagedObjectContext * context = [self managedObjectContext];  
if (editingStyle == UITableViewCellStyleDelete)  
{ // Delete the object from database.  
[context deleteObject:[self.recordList objectAtIndex:indexPath.  
row]];  
NSError * error = nil;  
if (! [context save:& error])  
{ NSLog(@"%@", @"Can't delete %@.", error, [error localizedDescription]);  
return;  
}}
```

```

    // Remove device / data from table view.
    [self.recordlist removeObjectAtIndex: indexPath.row];
    [self.tableView deleteRowsAtIndexPaths: [NS Array
        arrayWithObject: indexPath] withRowAnimation:
        UITableViewRowAnimationFade];
}
}

```

02/12/2013

* Working with Audio:-

- whenever we required to work with audio then, we need to use AVAudioPlayer object, SystemSoundID also we need for playing specific sound.
- whenever we are working with audio files then we need to add AVFoundation framework & AudioToolBox framework.

* Create a single view iOS app with the name AVPlayerApp.

- Place some images to the project with play, pause icon files.
- Place two audio files with the extension of .mp3 & .aif.
- Open viewController.xib & place following objects

- Place a button & change the title as "playsound".
- Place another button & change the title as "Play song".
- Place one more button & change the title as "play".
- Place a slider controller.
- Import AVFoundation framework to the project. & AudioToolbox file also.

* Code in ViewController.h :-

```
#import <AVFoundation/AVFoundation.h>
#import <AudioToolbox/AudioToolbox.h>
```

@interface ViewController : UIViewController <AVAudioPlayerDelegate>

{

```
SystemSoundID SystemSoundID;
AVAudioPlayer * player;
UIButton * StartStopSound;
UIButton * PlayPauseSound;
UISlider * volumeControl;
```

}

@property (nonatomic, retain) IBOutlet AVAudioPlayer * player;

UIButton * StartStopSound;

UIButton * PlayPauseSound;

UISlider * volumeControl;

-(IBAction) PlaySound: (id) sender;

-(IBAction) PlaySong: (id) sender;

-(IBAction) PauseSong: (id) sender;

-(IBAction) adjustVolume;

@end

}

- Open .xib file & make the relation bet" Objects & IBoutlets.
- Select Playsound button & make FirstResponser as "playSong".
- Make the relation from file's owner to play song & select the object "StartStopSound".
- Make the relation from playSong to FirstResponder & select "playSong" method.
- Make the relation from file's owner to Play button & select "playPauseSound".
- Make the relation from Play button to FirstResponder & select "pauseSong" method.
- select the slider object & make the relation to first Responder & select "adjustVolume".
- Set the slider's minimum value as 0 & max value is 100 & current value as 30.
- Make the relation from files owner to slider & select "volumeControl".

* Code in viewController.m:-

@synthesize player, playPauseSound, startStopSound,
volumeControl.

-(void) viewDidLoad.

{
player = [[AVAudioPlayer alloc] initWithContentsOfURL
:[NSURL fileURLWithPath: [[NSBundle mainBundle]
pathForResource: @"Song Name" ofType:
@"mp3"]]. error:nil];

[player setDelegate: self];

[player prepareToPlay];

[playPauseSound setAlpha:0];

AudioServicesCreateSystemSoundID((CFURLRef)[NSURL fileURL-
withPath: [[NSBundle mainBundle] pathForResource:
@"Sound" ofType: @"aif"]], &systemSoundID);

}

-(IBAction) playSong : (id)sender

{ if ([startStopSound currentTitle] isEqualToString:@"Play Song"])

{ [player play];

[startStopSound setTitle:@"Stop Song" forState: UIControlStateNormal];

[playPauseSound setTitle:@"Pause" forState:
UIControlStateNormal];

[playPauseSound setAlpha:1];

}

```

else if ([startstopSound currentTitle] isEqualToString:@"stop song"])
{
    [player stop];
    player=nil;
    player=[[AVAudioPlayer alloc] initWithContentsOfURL:
        [NSURL fileURLWithPath:[[NSBundle mainBundle] pathForResource:@"Song Name" ofType:@"mp3"]]
        error:&nil];
    [player prepareToPlay];
    [startstopSound setTitle:@"Play song" forState:UIControlStateNormal];
    [playPauseSound setAlpha:0];
}

-(IBAction)PauseSong:(id)sender
{
    if ([CCPlayPauseSound currentTitle] isEqualToString:@"play"])
    {
        [player play];
        [PlayPauseSound setTitle:@"Pause" forState:UIControlStateNormal];
    }
}

-(void)adjustVolume
{
    if (player != nil)
    {
        player.volume = volumeControlValue;
    }
}

```

```

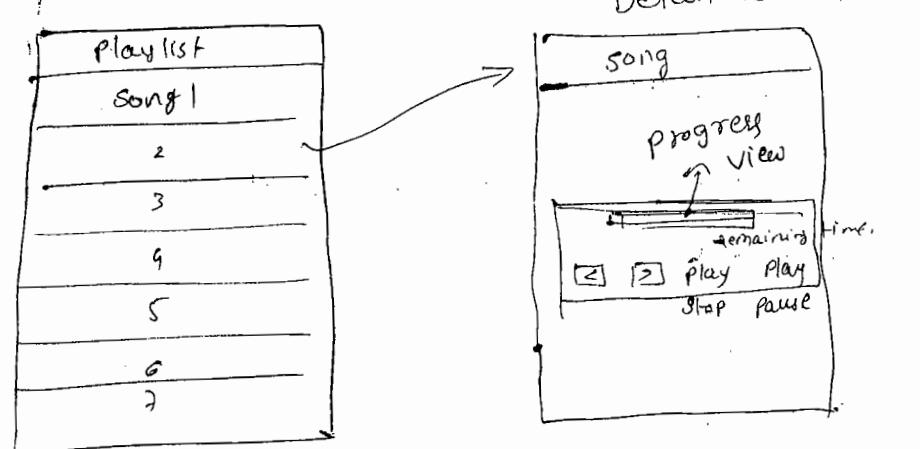
- (IBAction) playSound : (id) sender
{
    AudioServicePlaySystemSound (SystemSoundID);
}

#pragma -mark AVAudioPlayer.

-(void) audioPlay:(AVAudioPlayer*) player
successfully:(BOOL)flag.

{
    self.player = [[AVAudioPlayer alloc] initWithContentsOfURL
        :[NSURL fileURLWithPath:[[NSBundle mainBundle] pathForResource:@"songName" ofType:@"mp3"]]
        error:nil];
    [self.player prepareToPlay];
    [startStopSound setTitle:@"Play Song" forState:UIControlStateNormal];
    [playPauseSound setTitle:@"Pause" forState:UIControlStateNormal];
    [playPauseSound setAlpha:0];
}

```



Master-Detail

* Working with AVAudioRecorder:-

- with the help of this class object we can record sound.
- while the recording is happening automatically device microphone will enable.
- Through AVAudioRecorder what file we constructed those files can be stored in document directory with the help of .caf extension.

* Create a single view app with the name AVAudioRecorderApp.

- import following frameworks.
 - AVFoundation framework
 - AudioToolbox f/w.
- open ViewController.xib file & place 3 buttons with the title record, play, stop.

* Code in Viewcontroller.h :-

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

```
@interface ViewController : UIViewController <AVAudioRecorderDelegate, AVAudioPlayerDelegate>
```

```
{  
    AVAudioRecorder *audioRecorder;  
    AVAudioPlayer *audioPlayer;  
    UIButton *playButton;  
    UIButton *recordButton;  
    UIButton *stopButton;  
}
```

• @property (nonatomic, retain) IBOutlet UIButton * playButton;
" " recordButton;
" " stopButton;
- (IBAction) recordAudio;
- (IBAction) playAudio;
- (IBAction) stopRecord; ~~Play~~
@end.

- * open `.ViewController.xib` & make the relation from file's owner recordButton & select recordButton outlet
- Make the relation from record button toFirstResponder then select record Audio;
- File's owner → play button $\xrightarrow{\text{select}}$ playButton outlet
play button → First Responder $\xrightarrow{\text{select}}$ playAudio method.
- file's owner → stop button $\xrightarrow{\text{select}}$ stopButton outlet.
stop button → First Responder $\xrightarrow{\text{select}}$ stop~~Play~~ method.

* code in Viewcontroller.m :-

@interface ViewController :
@end.
@implementation ViewController
@synthesize recordButton, playButton, stopButton.

```

-(void) ViewDidLoad
{
    [PlayButton setAlpha:0];
    [StopButton setAlpha:0];
    NSArray * dirPath = NSSearchPathForDirectoriesInDomains(
        NSDocumentDirectory NSUserDomainMask, YES);
    NSString * docDir = [dirPath objectAtIndex:0];
    NSDate * date = [NSDate date];
    NSString * tempName = [NSString stringWithFormat:@"sound%@.caf",
                           date];
    NSString * soundFilePath = [docDir stringByAppendingPathComponent:tempName];
    NSURL * soundfileURL = [NSURL fileURLWithPath:soundFilePath];
    NSDictionary * recordSettings = [NSDictionary dictionaryWithObjectsAndKeys:
        [NSNumber numberWithInt:AVAudioQualityMax], AVEncoderAudioQualityKey,
        [NSNumber numberWithInt:256], AVEncoderBitRateKey,
        [NSNumber numberWithInt:2], AVNumberOfChannelsKey,
        [NSNumber numberWithFloat:44100.0], AVSampleRateKey, nil];
}

// AVAudioQualityMin = 0 → 6.
// low = 0x20 → 32.
// Medium = 0x40 → 64
// High = 0x60 → 128
// Max = 0x7F → 256

```

```
    NSNull * error = nil;
    audioRecorder = [[AVAudioRecorder alloc] initWithURL:
        soundfileURL settings:recordSettings error:&error];
    audioRecorder.delegate = self;
    if (error)
        NSLog(@"%@", [error localizedDescription]);
    else
        [audioRecorder prepareToRecord];
}

-(IBAction)recordAudio
{
    [audioRecorder record];
    [recordButton setEnabled:FALSE];
    [recordButton setTitleColor:[UIColor redColor] forState:UIControlStateNormal];
    [stopButton setAlpha:1];
}

-(IBAction)stop
{
    if ([audioPlayer isPlaying]==YES)
    {
        [audioPlayer stop];
        audioPlayer=nil;
        [recordButton setEnabled:YES];
        [playButton setAlpha:0];
    }
}
```

```
else
{
    [audioRecorder stop];
    [recordButton setEnabled : YES];
    [recordButton setTitleColor: [UIColor blackColor]
        forState: UIControlStateNormal];
    [playButton setAlpha: 1];
}
-(IBAction) play Audio
{
    NSError *error;
    audioPlayer = [[AVAudioPlayer alloc] initWithContentsOfURL
        : audioRecorder.url error:&error];
    [audioPlayer prepareToPlay];
    [audioPlayer play];
    [recordButton setEnabled : FALSE];
}
#pragma -mark AVAudioRecorderDelegate Methods.
-(void) audioRecorderDidFinishRecording: (AVAudioRecorder*)
    recorder successfully: +
    (BOOL) flag
{
    UIAlertView * alert = [[UIAlertView alloc] initWithTitle:
        @"Record Alert" message: @"File is saved" delegate: nil
        cancelButtonTitle: @"OK" otherButtonTitles: nil, nil];
    [alert show];
}
```

```

○ -(void)audioRecorderEncoderErrorDidOccur : (AVAudioRecorder*)
    recorder error: (NSError *)error
{
    NSLog(@"%@", @"Encode Error occurred");
}
@ end.

```

→ XOX →

03/12/2013

* Working With MPMoviePlayerController :-

- This class is required when we are working with video files.
- When we are working with video files then we need to import MediaPlayer framework , MediaToolbox fw.

* Create a single view app with the name MoviePlayerApp & import MediaPlayer & MediaToolbox fw.

- Open viewController.xib & place a button with the title : "Play."

* code in viewController.h :-

```

#import <MediaPlayer/MediaPlayer.h>
#import <MediaToolbox/MediaToolbox.h>

```

○ interface ViewController : UIViewController

```

{
    MPMoviePlayerController *player
    IBOutlet UIButton *playbut;
}
```

}

```
@ Property (nonatomic, retain) MPMoviePlayerController *player;
----- IBoutlet -----
IBOutlet UIButton *playbtn; // play btn;

-(IBAction) playAction : (id) sender;
@end.
```

- Open viewcontroller.xib & make the relation from file's owner to "play" button & select IBoutlet.
- Make the relation play button to First Responder & select "playAction" method.
- Place a video file to the project.

* Code in Viewcontroller.m :-

```
@synthesize player, playbtn;
-(void) dealloc
{
    [player release];
    [super dealloc];
}

-(IBAction) playAction : (id) sender
{
    NSString * website = [[NSBundle mainBundle]
        pathForResource: @"video name . extension" ofType: nil];
    NSURL * url = [NSURL fileURLWithPath: website];
    player = [[MPMoviePlayerController alloc] initWithContentURL: url];
```

```
- [[NSNotificationCenter defaultCenter] addObserver: self  
    selector:@selector(moviePlaybackDidFinish:) name:  
        MPMoviePlayerPlaybackDidFinishNotification object: player];  
  
player.controlStyle = MPMovieControlStyleDefault;  
player.shouldAutoplay = YES;
```

```
[self.view addSubview: player.view];  
[player setFullscreen: YES animated: YES];
```

```
-(void) moviePlaybackDidFinish : (NSNotification*) notification  
{  
    MPMoviePlayerController *player = [notification object];  
    [[NSNotificationCenter defaultCenter] removeObserver: self  
        name: MPMoviePlayerPlaybackDidFinishNotification object: player];  
    if ([player respondsToSelector:@selector(setFullscreen:animated:)]  
    {  
        [player.view removeFromSuperview];  
    }  
}
```

```
- (void) viewDidLoad
```

```
{
```

* UIWebView :-

- With the help of this view we can display web page on view.
- If we required any specific actions on web view then we required to handle webView related delegate methods.
- When we required to display any specific webpage then we need to pass NSURL request. to load that specific web page.

* Create a single view appl' with the name webViewApp.

- Open ViewController.xib & place web view.

* code in ViewController.h :-

```
@interface ViewController : UIViewController < UIWebViewDelegate,
```

```
{ IBOutlet UIWebView * webView;
```

```
@property (nonatomic, retain) IBOutlet UIWebView * webView;
```

```
@property (nonatomic, retain) UIActivityIndicatorView * loading;
```

```
@end.
```

- Go to viewController.xib & make the relation from file's owner to UIWebView & select webview.

* code in viewController.m :-

```

@synthesize webView, loading;
-(void) viewDidLoad.
{
    [self.view setBackgroundColor:[UIColor blackColor]];
    Loading = [UIActivityIndicatorView alloc] initWithFrame:Activity -
        IndicatorStyle: UIActivityIndicatorViewStyleWhiteLarge];
    [loading setFrame:CGRectMake(100, 180, 80, 80)];
    [loading setHidden:YES];
    NSString * urlAddress = @"http://epaper.sakshi.com";
    // url page address

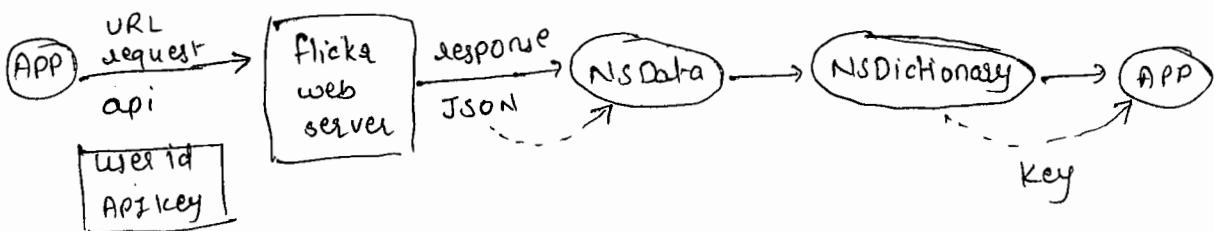
    // create an URL obj
    NSURL * url = [NSURL URLWithString:urlAddress];
    // URL request obj
    NSURLRequest * requestObj = [NSURLRequest requestWithURL:url];
    // load the request in UIWebView.
    [webView loadRequest:requestObj];
    [webView setDelegate:self];
    [webView setBackgroundColor:[UIColor blackColor]];
    [webView addSubview:loading];
}

```

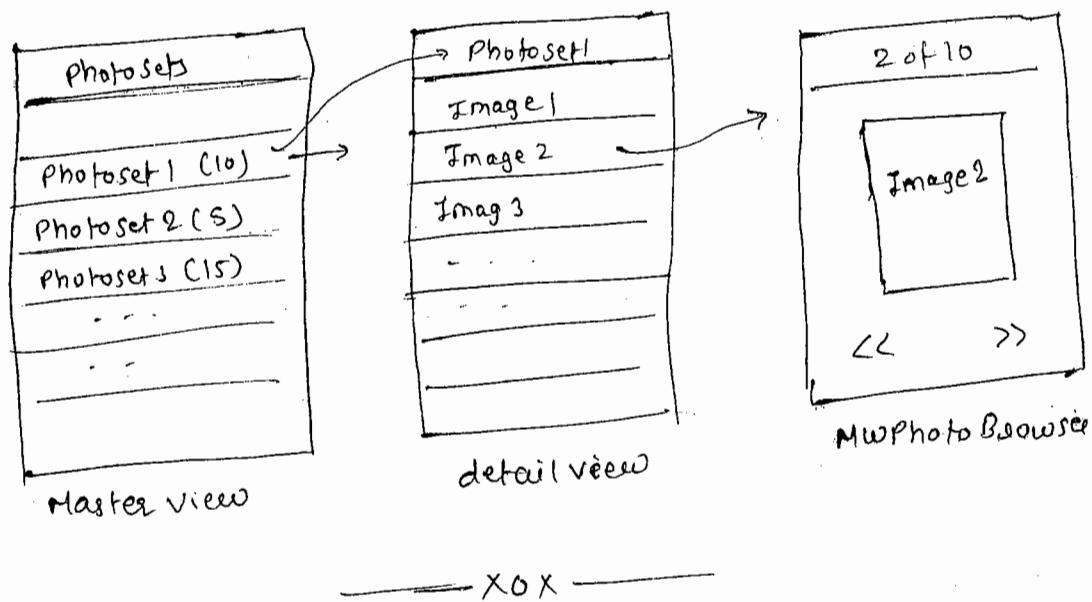
```
# pragma -mark UIWebViewDelegate  
  
-(void)webViewDidStartLoad:(UIWebView *)webView  
{  
    [loading startAnimating];  
}  
  
-(void)webViewDidFinishLoad:(UIWebView *)webView  
{  
    [loading stopAnimating];  
}  
  
-(void)webView:(UIWebView *)webView didFailLoadWith-  
Error:(NSError *)error.  
{  
    UIAlertView *alert=[[UIAlertView alloc] initWithTitle:  
    @"Alert" message:@"Network Error" delegate:nil  
    cancelButtonTitle:@"OK" otherButtonTitles:nil];  
    [alert show];  
}  
@end.
```

* Flicker Integration through MuPhotoBrowser :-

- Flicker is a social network which is maintaining by Yahoo group for sharing the images.
- When we required to work with flicker integration we doesn't having any in built fw.
- Flicker integration will works with the help of flicker API which is provided by flicker services.
- Complete flicker API reference we can get from <http://www.flickr.com/services/api/>.
- When we required to work on flicker services we need to get an unique user id along with API key also which can be generated after logging in the account.
- For different apps we can use common id but API key can be changed.
- For finding the user id we can use <http://www.flickr.com/services/api/explore/flickr.people.findByUsername>.
- By using following link we can provide API key value.



* Create a master-detail video app with the name FlickrApp.



05/12/2013

* Integration of Youtube:-

- YouTube is a video hosting social n/w which is maintaining by Google.
- when we are working with youtube server it maintain the data in the form of xml.
- YouTube related APIs are available in gdata.youtube.com
- when we are working with gdata APIs then directly we can get authentication related data without any key also.

- When we are working with youtube then, xml format parsers should be required.



* Create a master-detail view appl with the name youtube App.

- Import libxml2.dylib, & libxml2.2.dylib files.
- Create a new objective c class with the name XMLReader subclass of type NSObject.

⇒ R.T.Zip

* Working with Twitter, Facebook flw :-

- In iOS 5 twitter flw is provided by Apple, in iOS 6 Facebook flw is introduced by Apple.
- In previous versions developer have to make Twitter or Facebook API for integration. & it became too complex to work with APIs which is provided by social n/w.
- In iOS 6.0 a new flw is introduced with the name social flw which is supported social n/w services.

- At present this flow will support Facebook, Twitter and weibo.
- This flow will provide standard interface to interact with different types of social flow.
- This social flow will come with a class called "SLComposeViewController".
- The SLComposeViewController class will provide standard view controller for the users to compose ~~text or~~ tweet or Facebook post.
- This class will allow to the developer to add initial text, url, & images.
- When we are composing the Twitter data, not recommended to compose image because it is limited memory but still if we are composing then it will convert image into url, when we click on that url then it shows image.
- When we are composing the Facebook then, it allows to add images directly.
- This social flow is provided basic posting services only, if we required to implement any other services then must be required to go for APIs only.
- For Twitter API reference we can use <https://dev.twitter.com/docs/api/1.1>.

- For Facebook APIs we can use <https://developers.facebook.com/ios>
<https://developers.facebook.com/docs/tutorials/ios-sdk-tutorial/>,
<http://developers.facebook.com/docs/reference/ios/3.2/>.
- * Create a single view Application with the name socialnetworkApp.
- open ViewController.xib & place two RoundedRect buttons & change the title as Twitter & Facebook respectively.

* code in ViewController.h :-

```
@interface ViewController : UIViewController
```

```
{
```

```
}
```

```
- (IBAction) postToTwitter:(id)sender;
```

```
- (IBAction) postToFacebook:(id)sender;
```

```
@end
```

- open ViewController.xib & make the relation -twitter button to First Responder & select postToTwitter method.
- Make the relation facebook button to First Responder & select postTofacebook method.
- Go to project template file & add a new file called ~~Social.flw~~ Social.framework.

* code in viewcontroller.m :-

```
@implementation ViewController  
-(IBAction)postToTwitter:(id)sender  
{  
    if ([SLComposeViewController isAvailableForServiceType:  
        SLServiceTypeTwitter])  
{  
        SLComposeViewController * tweetSheet = [SLComposeViewController  
        composeViewControllerForServiceType:SLServiceTypeTwitter];  
        [tweetSheet setInitialText:@"Great to learn iOS"];  
        [tweetSheet addImage:[NSURL URLWithString:@"http://  
        www.nareshit.com"]];  
        [self presentViewController:tweetSheet animated:YES  
        completion:nil];  
        SLComposeViewControllerCompletionHandler completion = ^block  
        (SLComposeViewControllerResult result)  
        {  
            switch (result)  
            {  
                case SLComposeViewControllerResultDone:  
                    NSLog(@"%@", @"posted successfully");  
                    break;  
  
                case SLComposeViewControllerResultCancelled:  
                    NSLog(@"%@", @"post failed");  
                    break;  
            }  
        };  
    }  
}
```

```
[tweetsheet dismissViewControllerAnimated:YES  
completion:nil];  
};  
  
tweetsheet.completionHandler = completion;  
}  
}  
  
-(IBAction)postTofacebook:(id)sender  
{  
if([SLComposeViewController isAvailableForServiceType:  
SLServiceTypeFacebook]  
{  
SLComposeViewController *facebookSheet = [SLComposeViewController  
serviceType:SLServiceTypeFacebook];  
  
[facebookSheet setInitialText:@"Great fun to learn iOS"];  
[facebookSheet addImage:[UIImage imageNamed:@"default.png"]];  
  
[facebookSheet addURL:[NSURL URLWithString:@"http://  
www.nareshit.com"]];  
  
[self presentViewController:facebookSheet animated:YES  
completion:nil];  
  
SLComposeViewControllerCompletionHandler completion = ^{  
(SLComposeViewControllerResult result)  
{  
switch(result)  
{  
case SLComposeViewControllerResultDone:  
NSLog(@"posted successfully");  
break;  
}};
```

case SLComposeViewControllerResultCancelled:

 NSLog(@"could not post");

 break;

 default;

 break;

}

}

[facebookSheet dismissViewControllerAnimated:YES

 completion: nil];

}

facebookSheet.completionHandler = completion;

{

}

07/12/2013

* Google Map Integration with annotations:-

- Google maps are very popular services for any kind of appⁿ i.e. web appⁿ or desktop appⁿ or mobile appⁿ.
- When we are making the integration of google maps in iOS appⁿ, then google map SDK must be downloaded.
- Google Map SDK can be downloaded from
<https://developers.google.com/maps/documentation/ios/start>
- If you required to work with Google Map services then must be required API key. which can be created from
- When we make any services enable mode, automatically API key will be generated.

* Working with Google SDK in current appⁿ:

* Create a single view appⁿ with the name GoogleMapApp.

- Google Map SDK required to place in current project.
- Drag GoogleMaps.framework from SDK folder to current appⁿ.
- Then choose file→add files to the GoogleMapApp & select Resource directory. which will add GoogleMaps.bundle file.
- Go to the project build phase & following libraries.

- | | |
|---------------------------------|--------------------------|
| ① SystemConfiguration.framework | ⑦ GLKit.framework |
| ② QuartzCore.framework | ⑧ CoreData.framework |
| ③ OpenGLES.framework | ⑨ CoreText.framework |
| ④ libicucore.dylib | ⑩ CoreLocation.framework |
| ⑤ libctt.dylib | ⑪ AVFoundation.framework |
| ⑥ ImageIO.framework | ⑫ libz1.1.3.dylib |

- Goto project template & choose build setting tab & select other linker flag section. & add -ObjC.

* Code in AppDelegate.m :-

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

write following statement in didFinishLaunchingWithOptions method:-

```
#warning API_KEY_REQUIRED_HERE
```

```
[GMServices provideAPIKey:@"API_KEY"];
```

* Code in ViewController.h :-

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

```
@interface ViewController : UIViewController <GMSMapViewDelegate>
```

```
{
    GMSMapView *mapView_;
```

```
}
```

```
@property(nonatomic,retain) GMSMapView *mapView_;
```

```
@end.
```

* Code in ViewController.m :-

```
#import <GoogleMaps/GoogleMap.h>
```

```
@synthesize mapView_;
```

```
- (void)viewDidLoad {
```

```
    {
        GMSCameraPosition *camera = [GMSCameraPosition
            cameraWithLatitude:17.3660 longitude:78.4760 zoom:3];
```

```
mapView_ = [GMSMapView mapWithFrame: CGRectMake(0, 0, cameraWidth, cameraHeight) camera: camera];
mapView_.myLocationEnabled = YES;
self.view = mapView_;
// creates marker in the center of the map.
GMSMarker *marker = [[GMSMarker alloc] init];
marker.position = CLLocationCoordinate2DMake(17.3660, 78.4760);
marker.title = @"Hyderabad";
marker.snippet = @"Ameerpet";
marker.map = mapView_;
[self.mapView setDelegate: self];
self.view = mapView_;
UILongPressGestureRecognizer *longPressGesture =
[[UILongPressGestureRecognizer alloc] initWithTarget: self
action: @selector(handleLongPressGesture:)];
[self.mapView addGestureRecognizer: longPressGesture];
[longPressGesture setMinimumPressDuration: 3];
}
#pragma - mark GMSMapViewDelegate
-(void)mapView: (GMSMapView *)mapView didLongPressAtCoordinate: (CLLocationCoordinate2D)coordinate
{
GMSMarker *marker = [[GMSMarker markerWithPosition: coordinate] init];
marker.title = @"Hello World";
marker.map = self.mapView;
}
```

```
- (BOOL) mapView:(AMSTMapView *) mapView didTapMarker:(AMSMarker *) marker.  
{  
    marker.map = nil;  
    return TRUE;  
}
```

* Working with Apple MapKit :-

- when we required to implement mapView in ios appl' then go for MKMapView object.
- when we are using MKMapViewController then we need to import

- * Create a single view appl' with the name MKMapKitApp.
- Add corelocation & MapKit fw.
- Create a new objective C class with the name DisplayPin. subclass of type NSObject.

* code in DisplayPin.h :-

```
#import <MapKit/MapKit.h>  
@interface DisplayPin : NSObject <MKAnnotation>  
{  
    CLLocationCoordinate2D coordinate;  
    NSString * title;  
    NSString * subtitle;  
}
```

```
@property (nonatomic, retain) CLLocationCoordinate2D coordinate;
" "
" "
NSStrинг * title;
NSStrинг * subtitle;
```

@ end

* code in DisplayPin.m :-

```
@synthesize coordinate, title, subtitle;
```

@ end

* code in viewController.h :-

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

```
@interface ViewController : UIViewController < MKMapViewDelegate
```

```
UIGestureRecognizedDelegate
```

```
{
```

```
IBOutlet MKMapView * mapView;
```

```
}
```

```
@property (nonatomic, retain) IBOutlet MKMapView * mapView;
```

@ end.

- Go to viewController.xib & place a MapView.
- Make the relation from File's owner to MKMapView & choose MapView outlet.
- Make the relation from mapView to file's owner & select delegate.

* code in ViewController.m :-

#import "DisplayPin.h" #import <CoreLocation/CoreLocation.h>

@synthesize mapView;

- (void) viewDidLoad

{

[mapView setMapType: MKMapTypeStandard];

[mapView setZoomEnabled: YES];

[mapView setScrollEnabled: YES];

CLLocationCoordinate2D position = CLLocationCoordinate2DMake (17.3660, 78.4760);

[mapView setCenterCoordinate: position];

[mapView setDelegate: self];

DisplayPin * pin1 = [[DisplayPin alloc] init];

pin1.title = @"Hyderabad";

pin1.subtitle = @"Ameerpet";

pin1.coordinate = CLLocationCoordinate2DMake (17.3660, 78.4760);

[mapView addAnnotation: pin1];

UILongPressGestureRecognizer * lgr1 = [[UILongPressGestureRecognizer alloc] initWithTarget: self action: @selector(handleGesture:)];

lgr1.minimumPressDuration = 2.0;

[mapView addGestureRecognizer: lgr1];

}

```

- (void) handleGesture : (UIGestureRecognizer *) gestureRecognizer
{
    if (gestureRecognizer.state != UIGestureRecognizerStateEnded)
        return;

    CGPoint touchPoint = [gestureRecognizer locationInView: mapView];
    CLLocationCoordinate2D touchMapCoordinate =
    [mapView convertPoint: touchPoint toCoordinateFromView: mapView];
    MKPointAnnotation * pa = [[MKPointAnnotation alloc] init];
    pa.coordinate = touchMapCoordinate;
    pa.title = @"Hello";
    [mapView addAnnotation: pa];
}

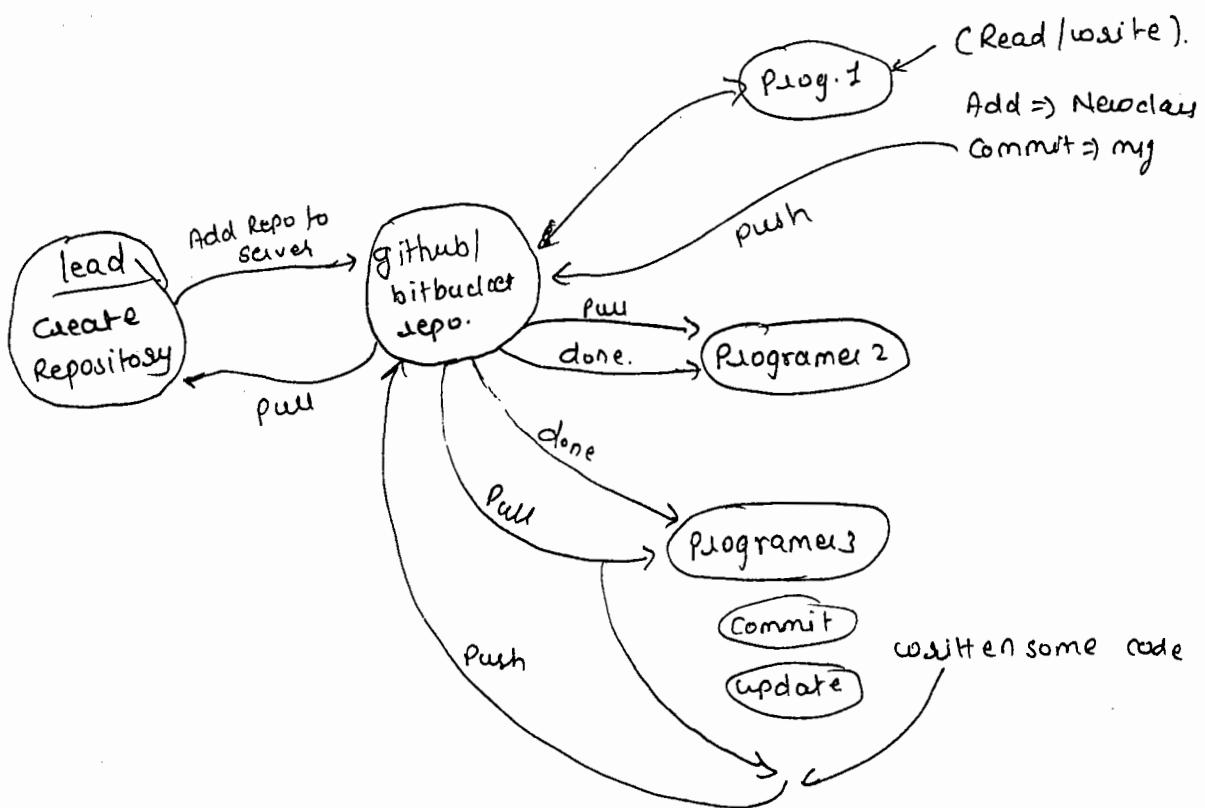
```

—xox—
Repository:-

- A Repository is a directory which can allow to work multiple programmer at a time on single project.
- Repository can be local or remote location
- Remote repositories always handled by web services only.
- Generally in iOS appl' when source code is controlling through repository then we are using "Github" or "bitbucket" servers.
- Github & bitbucket both are free source controlling tools. which works with the help of Mercurial.

- A mercurial is a command tool appⁿ which works through terminal for controlling source code activities like cloning, pulling, pushing & updating of repository.
- Mercurial tool can be downloaded from <http://mercurial.selenic.com/>.
- Before installing the mercurial tool we need to install Python in Mac
- Python can be downloaded from <http://python.org/getit/mac/>.

* Working Flow with github/bitbucket server:-



* Creation of .ipa file :- (iPhone Archive)

- For every app we required to create a provisional certificate from Apple Developer Tool. (If it is premium account then only possible).
- Once after creating the provisional certificate we need to download & add into current system. (adding to keychain access).
- If we required to add the provisional certificate to current project then choose project template → build setting → code signing identity & select specific provisional certificate. (app bundle identifier must be match with provisional certificate identifier).
- Change the active scheme to iOS device. (which indicates run the app in simulator or iOS device).
- Go to product option & make "Archive".
- When the Archive process is completed then we'll get .ipa file with validation & distribute option.
- Validation will take place the app's validation with App Store.
- When we make distribute option we can directly distribute app to App Store or choose Ad-hoc deployment.
- When we choose Ad-hoc deployment then it saves copy of .ipa file into local system.

- When we required to test the applⁿ in devices before submitting into App Store we need to go for testing through "TestFlight".
- TestFlight related more info we can get from testflightapp.com.
- The procedure of TestFlight submission can be get from

<http://help.testflightapp.com/customer/portal/topics/184165-tutorials/articles>.