OBJECTIVE-C INHERITANCE

One of the most important concepts in object-oriented programming is that of inheritance. Inheritance allows us to define a class in terms of another class which makes it easier to create and maintain an application. This also provides an opportunity to reuse the code functionality and fast implementation time.

When creating a class, instead of writing completely new data members and member functions, the programmer can designate that the new class should inherit the members of an existing class. This existing class is called the **base** class, and the new class is referred to as the **derived** class.

The idea of inheritance implements the **is a** relationship. For example, mammal IS-A animal, dog IS-A mammal, hence dog IS-A animal as well and so on.

Base & Derived Classes:

Objective-C allows only Multiple inheritance, i.e., it can have only one base class but allows multilevel inheritance. All classes in Objective-C is derived from the superclass **NSObject**.

```
@interface derived-class: base-class
```

Consider a base class **Shape** and its derived class **Rectangle** as follows:

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

@interface Person : NSObject

{
    NSString *personName;
    NSInteger personAge;
}

- (id)initWithName:(NSString *)name andAge:(NSInteger)age;
- (void)print;
@end

@implementation Person

- (id)initWithName:(NSString *)name andAge:(NSInteger)age{
    personName = name;
    personAge = age;
    return self;
}

- (void)print{
    NSLog(@"Name: %@", personName);
    NSLog(@"Age: %ld", personAge);
}

@end

@interface Employee : Person

{
    NSString *employeeEducation;
}

- (id)initWithName:(NSString *)name andAge:(NSInteger)age
  andEducation:(NSString *)education;
- (void)print;

@end
```

```
@implementation Employee

- (id)initWithName:(NSString *)name andAge:(NSInteger)age
  andEducation: (NSString *)education
  {
    personName = name;
    personAge = age;
    employeeEducation = education;
    return self;
}

- (void)print
{
    NSLog(@"Name: %@", personName);
    NSLog(@"Age: %ld", personAge);
    NSLog(@"Education: %@", employeeEducation);
}

@end


int main(int argc, const char * argv[])
{
    NSAutoreleasePool * pool = [[NSAutoreleasePool alloc] init];
    NSLog(@"Base class Person Object");
    Person *person = [[Person alloc]initWithName:@"Raj" andAge:5];
    [person print];
    NSLog(@"Inherited Class Employee Object");
    Employee *employee = [[Employee alloc]initWithName:@"Raj"
    andAge:5 andEducation:@"MBA"];
    [employee print];
    [pool drain];
    return 0;
}
```

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

```
2013-09-22 21:20:09.842 Inheritance[349:303] Base class Person Object
2013-09-22 21:20:09.844 Inheritance[349:303] Name: Raj
2013-09-22 21:20:09.844 Inheritance[349:303] Age: 5
2013-09-22 21:20:09.845 Inheritance[349:303] Inherited Class Employee Object
2013-09-22 21:20:09.845 Inheritance[349:303] Name: Raj
2013-09-22 21:20:09.846 Inheritance[349:303] Age: 5
2013-09-22 21:20:09.846 Inheritance[349:303] Education: MBA
```

Access Control and Inheritance:

A derived class can access all the private members of its base class if it's defined in the interface class, but it cannot access private members that are defined in the implementation file.

We can summarize the different access types according to who can access them in the following way:

A derived class inherits all base class methods and variables with the following exceptions:

- Variables declared in implementation file with the help of extensions is not accessible.
- Methods declared in implementation file with the help of extensions is not accessible.
- In case the inherited class implements the method in base class, then the method in derived class is executed.