iOS App Development

1.2 Objective-C & SDK Frameworks

Programming

- Programming Language
  - Objective-C
- SDK Frameworks
  - Cocoa Touch Layer
  - Media Layer
  - Core Service Layer
  - Core OS Layer

Objective-C

- Interface to iOS SDK
- Superset of C
- Classes
- Method and Messaging

A Superset of C

- ANSI C에 Objective-C Syntax가 추가된 형태
- ANSI C와 동일한 Syntax
- Header file extension: ‘.h’
- Source file extension: ‘.m’
- Objective-C 와 C++ code가 혼합된 Source file extension: ‘.mm’
- #import : #include와 동일하지만 중복 include를 배제

C vs. Objective-C

“C”

```c
//  democ.h
#ifndef Demo1_democ_h
#define Demo1_democ_h

int multiply(int val, int multiplier);

#endif
```

“Objective-C”

```objc
//  DemoObjC.h

#import <UIKit/UIKit.h>

@interface DemoObjC : NSObject

- (int) multiply:(int) val by:(int) multiplier;

@end
```

C vs. Objective-C

“C”

```c
//  democ.c

#include <stdio.h>
#include "democ.h"

int multiply(int val, int multiplier)
{
    return val * multiplier;
}
```

“Objective-C”

```objc
//  DemoObjC.m
#import "DemoObjC.h"

@implementation DemoObjC
// result = [obj multiply:4 by:3];
- (int) multiply:(int) val by:(int) multiplier {
    return val * multiplier;
}
@end
```

Classes

- Class: Data와 Action(method)으로 구성
- Object: Runtime instance of a class
  - @interface Directive
    - Member Variables and Methods
  - @implementation Directive
    - Actual implementation of Methods

Classes

```
@interface MyClass : NSObject
{
    int count;
    id data;
    NSString *name;
}
- (id) initWithName:(NSString *) name;
+ (MyClass *) createWithName:(NSString *)
name;
@end
```

Class name

Parent class name

Member variable declaration

Method declaration

Methods

- Instance method
- Class method
- Method declaration

```
- (void)insertObject:(id)object atIndex:(NSUInteger) index;
```

  - Method type identifier
  - Return type and parameter types
  - Method signature keywords (method 이름)
  - Parameter names

Method vs. Function

```
void insertObject( void * object, unsigned int index);
```

“C”

- Return type and parameter types
- Function name (symbol: ‘_insertObject’)
- Parameter names

```
- (void)insertObject:(id)object atIndex:(NSUInteger) index;
```

- Method type identifier
- Return type and parameter types
- Method signature keywords (method 이름)
- Parameter names

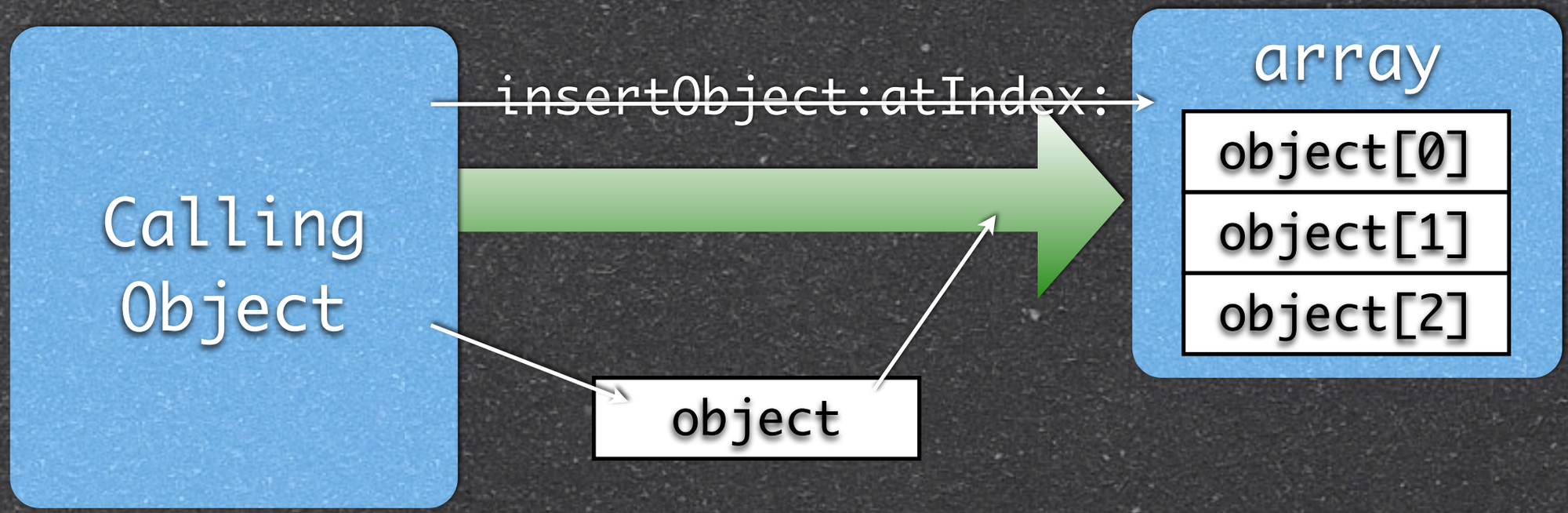
Messaging

- Calling a method: “Messaging to an Object”

```
[array insertObject:object atIndex:0];

[[self array] insertObject:[object randomNewObject]
                   atIndex:0];
```

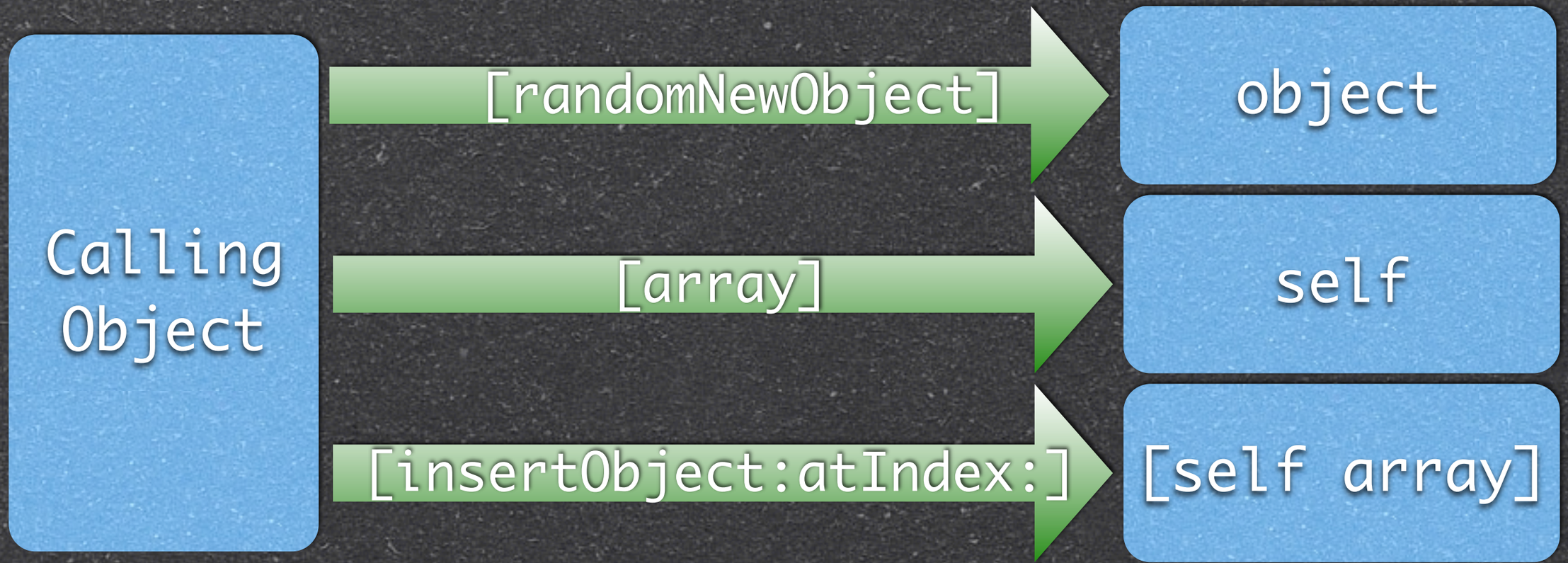
Messaging

```
[array insertObject:object atIndex:0];

[[self array] insertObject:[object randomNewObject]
                   atIndex:0];
```



Messaging

```
[array insertObject:object atIndex:0];


[[self array] insertObject:[object randomNewObject]
                   atIndex:0];
```



Class Definition

- @interface Directive

```
@interface MyClass : NSObject
{
    int count;
    id data;
    NSString *name;
}
- (id) initWithName:(NSString *) aName;
+ (MyClass *) createWithName:(NSString *) aName;
@end
```

Class Implementation

- @implementation Directive

```
@implementation MyClass

- (id) initWithName:(NSString *) aName
{
    if (self = [super init] ) {
        name = [aName copy];
    }
    return self;
}

+ (MyClass *) createWithName: (NSString *) aName
{
    return [[self alloc] initWithString: aName];
}
@end
```

Using Class Object

```
- (void) test {
    // Creating object using an initializer
    MyClass *object1 = [[MyClass alloc] initWithName:@"Object 1"];

    // Creating object using a class method
    MyClass *object2 = [MyClass createWithName:@"Object 2"];

    // Access to member variables is forbidden
    object1.count = object1.count + 1;
}
```

Accessor Methods

- Getter
  - NSLog(@"%@", [object name]);
  - NSLog(@"%@", object.name);
- Setter
  - [object setName:@"New name"];
  - object.name = @"New Name";
- @property Directive
  - In Class Definition
  - @property (nonatomic, strong) NSString *name;

Accessor Methods

```
// Accessor Method Declarations
-(type)propertyName; // Getter
-(void)setPropertyName:(type) n; // Setter

// Example: '.name' access enabled
- (NSString *) name;
- (void) setName:(NSString *) name;
```

Property and Accessor Methods

```
object.array = @[...];
```

Object

MyClass Class

Methods:

```
- (void) setArray:(NSArray *) array;

- (NSArray *) array;
```

Invoking getter by reading the property:

```
arr = object.array;
```

Property

- Declaring @property is almost same as declaring accessor methods pair
  - With some exceptions

```
// Property
@property (nonatomic, strong) NSString *name;

// Or, Accessor methods pair
- (NSString *) name;
- (void) setName:(NSString *) name;
```

Objective-C Summary

- Superset of C
- Class and Object
- Method and Messaging
- Accessor methods and Sneak Peek at @property

SDK

- Cocoa Touch Layer
- Media Layer
- Core Service Layer
- Core OS Layer

Cocoa Touch Layer

- Frameworks
  - Address Book UI
  - UIKit
  - Map Kit
  - Social
- Features
  - Storyboard
  - Printing
  - Push Notification Service
  - Gesture Recognizing

Media Layer

- Graphics/Audio/Video Technologies
- Frameworks
  - Assets Library
  - AV Foundation
  - OpenGLES, GLKit
  - Media Player
  - Core Audio/Graphics/Image/Video

Core Service Layer

- Accounts
- Address Book
- CFNetwork
- Core Data
- Foundation and Core Foundation
- Core Location
- Event Kit

Core OS Layer

- Accelerate (DSP)
- Core Bluetooth
- External Accessory
- Security