Objective-C

Presented By
Abdelrahman Sayed

Java™ Education and Technology Services

Invest In Yourself, Develop Your Career

Lecture One

Agenda

- Introduction.
- Objective-C.
- Classes & Objects.
- Methods.
- Choosing Names
- Data Types.
- Loops.
- Conditions.
- Comments.

[Introduction]

Apple History

How it all started !!

Apple History

Founded in 1976 by Steve Jobs, Steve Wozniak and Ronald Wayne, the company developed one of the first personal computers, called the Apple 1. The early Apple computers, led the way in terms of their user-friendly graphical operating systems, at a time when computers were dominated by IBM and their 'less than intuitive' operating system, called MS-DOS.


Original Logo

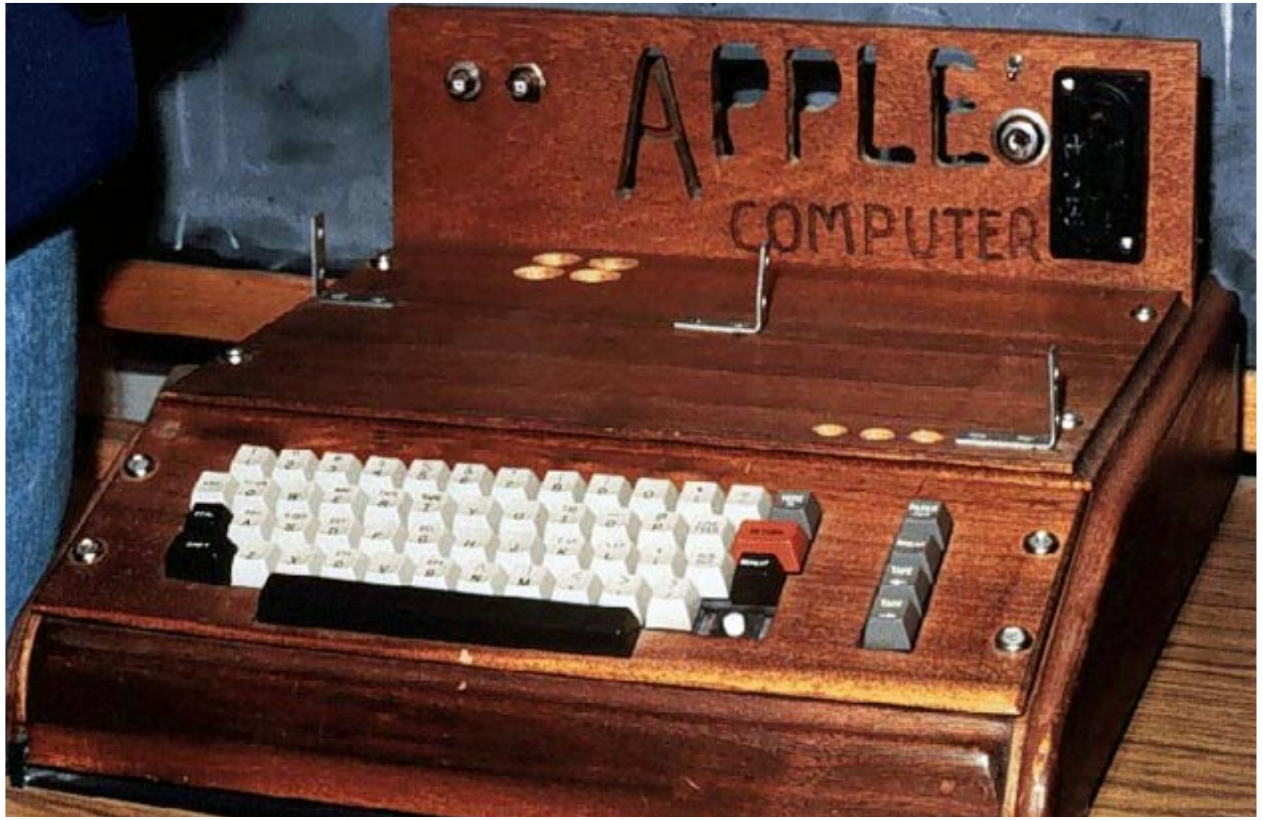

1976-1998


1998-2000


2001-2007


From 2007

Apple I



Apple II

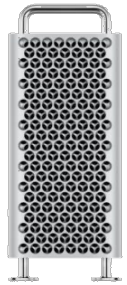
Apple III

Apple Inc.

Java™ Education and Technology Services

Apple Products

- Computers
  - MacBook Pro
  - MacBook Air
  - Mac mini
  - iMac
  - Mac Pro
  - Apple TV



Apple Products Cont.

- Mobile devices
  - iPhone
  - iPad - iPad mini
  - iPod
  - iWatch
  - Home Pods

Apple Products Cont.

- Operating Systems
  - macOS for computers
  - iOS for mobile devices
  - tvOS
  - watchOS

[Why iOS ??]

Why iOS Development ??

- Every iPhone holder is a Loyal customer
- Ease of marketing
  - Upload to the store
  - Customers will try, no convincing is needed
- In Egypt:
  - Growing market
  - Rare developers

[Development Environment]

Development Environment

- In order to develop you should:
  - Know Objective-C
  - Have Mac machine
  - Have Developer license
  - Have iOS SDK
    - ➢Xcode
    - ➢iOS Simulator
    - ➢iOS developer documentation

[Objective-C]

Objective-C

- Objective-C is an object oriented programming language
- It's layered on top of the C language
  - Designed in early 1980s
- NeXT Software licensed Objective-C language in 1988, developed its libraries and a development environment called NEXTSTEP

Objective-C Cont.

- Apple Computer acquired NeXT in 1996
- NEXTSTEP environment became the basis for the next major release of Apple's operating system, OS X (Cocoa)
- In 2007, Apple released an update to the Objective-C language and labeled it Objective-C 2.0.
- Today: Objective-C and Swift are the native languages for developing Mac OS X and iOS applications

[Program Structure]

Program Structure

- Objective-C source files are identified by the .m file extension.
- Program entry point is: Main function

```
int main(int argc, char *argv[])
{
    // your code will go here
}
```

- **argc:** number of arguments
- **argv:** pointer to the list of arguments

Example

```
int main(int argc, char *argv[])
{
    printf("Hello world !!");
    return 0;
}
```

[Hello World Demo]

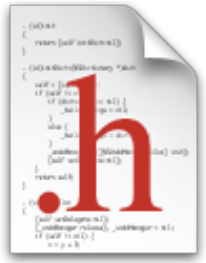
[Classes]

Definition

- Each class must have a parent class
- NSObject is the default parent of all classes (Root class)
- Objective-C classes definition is separated into two sections: interface (declaration) and implementation

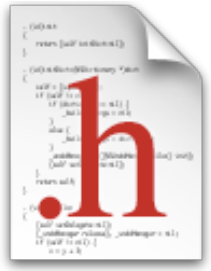
Definition Cont.

- Usually the interface section is in a .h file, and the implementation section in a .m file
- However both could be defined in one file
- More than one class could be defined in the same file



Interface

- The interface section defines:
  - The parent of the class.
  - Attributes and behaviors (methods).



Interface Cont.

- Syntax

```
@interface JETSClassName : ParentClassName{

    // here we define our variable

}

// here we define our methods

@end
```

Example

```
@interface JETSClassName : NSObject{

    int x;
    int y;

}

// here we define our methods

@end
```

Implementation

- The implementation section includes:
  - The actual implementation code of the predefined methods in the Interface file (.h file)

Implementation Cont.

```
#import "JETSClassName.h"

@implementation JETSClassName

    // methods implementation goes here

@end
```

- **Note:** No need for defining parent class here as it's already defined in the interface section

[Methods]

Methods

- There are two types of methods in Objective-C:
  - Instance methods
  - Class methods (Static)

Instance Methods

- To define an instance method that returns void and takes no parameters.

```
-(void) firstMethod;
```

- The – sign is used to identify instance methods

- Defining a method that returns void and takes one parameter : (int) x;

```
-(void) firstMethod : (int) x;
```

Instance Methods Cont.

- Defining a method that returns int and takes two parameters: int x, int y

```
-(int) setArea : (int) x : (int) y;
```

Class Methods

- To define a class method that returns void and takes no parameters

```
+(void) firstMethod;
```

- The + sign is used to identify class methods

[Sending Messages]

Sending Messages

- In Objective-C calling a method is named sending a message
- Syntax

```
[receiver message];

[receiver message:x]; // passing a variable x to the method
```

- **receiver:** is the class or object receiving the message
- **message:** is the method name (could be class or instance method)

[Creating objects]

Objects

- There are no constructors in Objective-C
- There are two ways of object creation
  - Object creation is performed on two stages
    - ➢Allocation of memory for the object using alloc Class method
    - ➢Initialization of the object using init instance method
  - Object creation is done using the new Class method
- new is effectively the same as calling alloc and init with no arguments

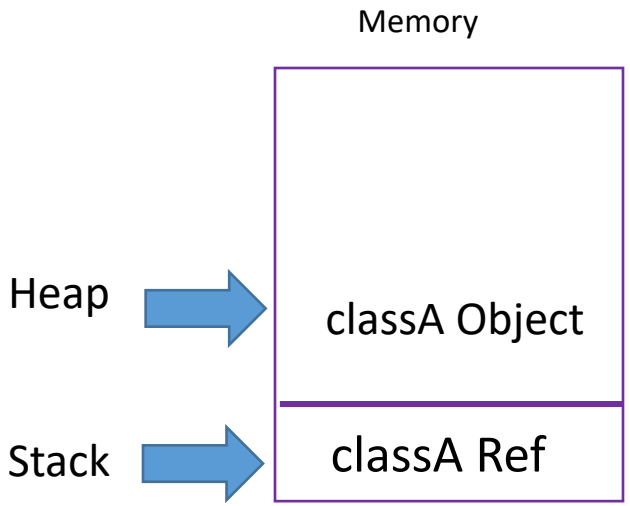
Example: Creating Objects

- Allocating memory:

```
JETSClassName *classA =[JETSClassName alloc];
```

- Initialize Object:

```
classA = [classA init];
```



Example Cont.

- Combined

```
JETSClassName *classA =[[JETSClassName alloc] init];
```

- Using new class method

```
JETSClassName *classA =[JETSClassName new];
```

Examples

```
JETSClassName *classA =[[JETSClassName alloc] init];

[classA firstMethod];

[classA firstMethod:x]; // passing int x to the method

int result = [classA setArea:x :y]; // passing two arguments to the method

[JETSClassName secondMethod]; // sending a message to a class method
```

Named Arguments Methods

```
int result = [classA setArea:x :y];
```

- What if we had more arguments ??

```
-(int) setWidth : (int) x andHeight :(int) y;
```

```
int result = [classA setWidth:x andHeight:y];
```

[Class Demo]

[Choosing Names]

Choosing Names

- Names in Objective-C must begin with a letter or underscore(_), and they can be followed by any combination of letters (upper - or lowercase), underscores, or the digits 0–9
- When naming a class, start it with a capital letter.
- Instance variables, objects, and method names, on the other hand, typically begin with lowercase letters
- Capital letters are used inside names to indicate the start of a new word

Choosing Names Cont.

- The following is a list of valid names:
  - sum
  - pieceFlag
  - i
  - myLocation
  - numberOfMoves
  - sysFlag
  - ChessBoard

Choosing Names Cont.

- **The following is a list of valid names:**
  - AddressBook → This could be a class name.
  - currentEntry → This could be an object.
  - current_entry → Some programmers use underscores as word separators.
  - addNewEntry → This could be a method name.

Choosing Names Cont.

- **On the other hand, the following names are not valid for the stated reasons:**
  - sum$value → $ is not a valid character.
  - piece flag → Embedded spaces are not permitted.
  - 3Spencer → Names can't start with a number.
  - int → This is a reserved word.

[Data types]

Main Data Types

- The vast majority of Objective-C's primitive data types are adopted from C
  - char
  - int
  - float
  - double

Data Types

- Data types are common on many programming languages, but they do not represent the same size as it depends on the machine
- In general, size of: char < int < float < double

[Loops]

For Loop

```
for (initialization; condition; increment) {
    statements
}
```

- Example

```
for (int i=0; i<100; i++) {

    printf("%d" , i);
}
```

For Loop Cont.

- Used when the number of iterations is known
- Infinite loop is due to:
  - Absence of condition statement
  - Condition is unreachable

While Loop

```
while (condition) {
    statements
}
```

- Example

```
while (i<100) {

    printf("i am number %d" , i);
    i++;
}
```

While Loop Cont.

- Condition based
- Number of iterations is not known
- Infinite loop due to:
  - Condition is unreachable
- Condition is checked first and then the body is executed

Do - While Loop

```
do {
     statements
} while (condition);
```

- Example

```
do {

     printf("i am number %d" , i);
     i++;

} while (i<100);
```

Do - While Loop Cont.

- Body is executed first, then condition is checked
- Body is executed at least once

Custom Looping

- Using continue:
  - Skips the current iteration and jumps to the next one
- Using break:
  - Skips the whole body of the loop

[Conditions]

if Statement

```
if (condition) {
    statements
}
```

- Example

```
if (i==50) {
    printf("This is the middle");
}
```

if - else Statement

```
if (condition) {
    statements-if-true
} else {
    statements-if-false
}
```

- Example

```
if (i==50) {
    printf("This is the middle");
} else {
    printf("Not there yet");
}
```

Switch-Case

```
switch (expression) {
    case constant:
        statements
        break;

    default:
        break;
}
```

Switch-Case Example

```
switch (value) {
    case 0:
        printf("I am number 0");
        break;
    case 1:
        printf("I am number 1");
        break;
    default:
        printf("I am not in range");
}
```

Ternary Operator

```
result = condition expression ? true expression : false expression;
```

- Example

```
int x = flag ? 5 : 6 ;
```

flag = true → x = 5

flag = false → x = 6

[Comments]

Comments

- To comment a single line:

```
// write a comment here
```

- To comment multiple lines:

```
/*  comment line 1
    comment line 2
     comment line 3 */
```

Remember

- Only *class* receivers receive *class* methods, and *object* receivers receive *instance* methods

- Declaration of objects from any class is preceded by an * sign (including Objective-C classes)

```
JETSClassName *classA =[JETSClassName new];
```

Remember

- Declaration of variables does not require * sign

```
int x;
```

- Method arguments could be variables (passed by value) or objects (passed by reference)

- Objective-C is case sensitive

[Lab Exercise]

1.Hello World

- Create a program that displays “Hello World”
- Use “printf”, and NSLog and state what is the difference between both.

2.Rectangle

- Create a program that calculates the area of a given rectangle
- The rectangle’s dimension should be obtained from the user, result should be displayed back using the command prompt

2.Rectangle Cont.

- The class structure should be as the following

```
@interface MyRect : NSObject{

    int width;
    int height;

}


-(void) setWidth:(int) w;
-(void) setHeight:(int) h;

-(int) getWidth;
-(int) getHeight;
-(int) printArea;

+(int) calcAreaWithWidth :(int) w andHeight: (int) h;


@end
```

3.Complex

- Create a program that represent complex number and has two methods to add and subtract complex numbers:

Complex number: **x + yi , 5+6i**

4.Star Diamond

- Create a program that prints the following pattern