iBooks Author

Chapter 3

Objective C - The Basic

ในบทนี้เราจะมาพูดถึงภาษา Objective C กัน ซึ่งภาษานี้จะเป็นภาษาหลักที่เราใช้ในการเขียนโปรแกรมสำหรับ iOS โดยเราจะพูดถึงพื้นฐานต่างๆของตัวภาษาที่จำเป็นสำหรับใช้ในการเขียนโปรแกรมบน iOS แต่จะขอไม่พูดถึงส่วนของภาษา C ซึ่งพื้นฐานของภาษา Objective C

ชนิดของไฟล์

1. **Header File (.h) : ใช้ประกาศคลาส เมธอด ตัวแปร**
2. **Source File (.m) : สำหรับภาษา C และ Objective C ใช้สำหรับ implement สิ่งที่ประกาศใน header file**
3. **Source File (.mm) : สำหรับภาษา C++ และ Objective C++ ใช้สำหรับ implement สิ่งที่ประกาศใน header file**

ภาษา Objective C นั้น ฟังจากชื่อแล้ว เราก็จะรู้สึกว่ามันน่าจะเป็นภาษาที่คล้ายๆกับภาษา C แน่เลย ซึ่งก็เป็นแบบนั้นจริงๆ ภาษา Objective C ถูกพัฒนามาจากภาษา C เพื่อให้รองรับคุณสมบัติของ Object-Oriented Programming (OOP) ซึ่งภาษา Objective C นั้นเป็นส่วนขยายของภาษา C หรือก็คือ เราสามารถนำโค๊ดของภาษา C มารันได้เลย โดยภาษา Objective C นั้นจะใช้ไฟล์นามสกุลต่างๆดังต่อไปนี้

1. Header File (.h): ใช้สำหรับประกาศคลาส เมธอด ตัวแปร ค่าคงที่ต่างๆ

2. Source File (.m): เป็นไฟล์ที่เก็บโค๊ดสำหรับทำงานของโปรแกรม โดยจะใช้เก็บทั้งโค๊ดจากภาษา C และ Objective C

3. Source File (.mm): เป็นไฟล์สำหรับเก็บโค๊ดเช่นเดียวกัน แต่จะใช้เก็บเฉพาะโค๊ดจากภาษา C++ และ Objective C++

สำหรับการเพิ่มไฟล์ header เข้ามาในโค๊ดที่เราต้องการใช้งาน เราจะใช้คำสั่ง

```
#import
```

โดยจะใส่ไว้ที่บริเวณส่วนบนของไฟล์ อย่างที่เราเห็นตัวอย่างกันแล้วในบทก่อนหน้านี้

ชนิดของตัวแปร

1. **ตัวแปรพื้นฐานเหมือนกับภาษา C**
2. **มีชนิดตัวแปรที่เพิ่มเข้ามา คือ**
   - **(1) BOOL : เก็บค่า YES, NO**
   - **(2) id : ใช้ระบุถึง object ใดๆ**
3. **NSString : เป็น object ของภาษา Objective C ใช้สำหรับเก็บข้อมูลที่เป็นข้อความ**

ตัวแปรพื้นฐานที่ใช้ในภาษา Objective C นั้นจะเหมือนกับตัวแปรพื้นฐานของภาษา C และจะมีตัวแปรบางชนิดเพิ่มเข้ามาในภาษา Objective C โดยเฉพาะ โดยชนิดของตัวแปรที่เพิ่มขึ้นมาใหม่จะมีดังนี้

1. BOOL ใช้เก็บค่าจริง เท็จ ซึ่งเป็นตัวแปรชนิดที่เพิ่มเข้ามาในภาษา Objective C จะเก็บค่าเป็น YES, NO

2. id ใช้สำหรับระบุถึง object ใดๆ โดยเราสามารถใช้ตัวแปรชนิด id สำหรับระบุถึง object จากคลาสใดๆก็ได้

นอกจากนี้แล้วยังมีคลาสที่สำคัญอีกคลาสหนึ่งที่จะได้ถูกนำมาใช้กันบ่อย นั่นคือ NSString ซึ่งจะใช้ในการเก็บข้อความ โดยในภาษา C นั้นเราจะเก็บข้อความเป็นลำดับของตัวอักษร ซึ่งเราก็สามารถเก็บข้อความในลักษณะนี้ในภาษา Objective C ได้เช่นกัน แต่โดยทั่วไปแล้วในภาษา Objective C จะเก็บข้อความเป็น object ของคลาส NSString แทน และการสร้างข้อความสำหรับ NSString นั้นจะใช้รูปแบบดังนี้

@”My String”

สำหรับการประกาศตัวแปรพร้อมทั้งกำหนดค่าให้กับตัวแปร NSString นั้นจะทำได้ดังนี้

```
NSString *myString = @”My String”;
```

การเรียกเมธอด

1. **การเรียกใช้งานเมธอด จะเรียกว่าเป็น การส่ง message ไปยัง receiver เพื่อทำงาน**
2. **การเรียกเมธอดจะอยู่ในรูป**

   [aObject aMethod];
3. **ถ้ามีพารามิเตอร์จะอยู่ในรูป**

   [aObject aMethod:aValue];
4. **สำหรับพารามิเตอร์มากกว่า 1 ตัว**

   [aObject aMethodWithParam1:aValue

   param2:bValue ...];
5. **อ้างอิงถึงเมธอดในรูป**

   aMethodWithParam1:param2:...

การเรียกใช้งานเมธอดของภาษา Objective C นั้นจะมีรูปแบบที่แตกต่างจากภาษาอื่นอยู่พอสมควร โดยจะมีรูปแบบการเรียกเมธอด aMethod ของ object aObject ดังนี้

[aObject aMethod];

และถ้ามีพารามิเตอร์ส่งไปด้วย ก็จะอยู่ในรูป

[aObject aMethod:aValue];

สำหรับการส่งพารามิเตอร์มากกว่า 1 ตัว ก็จะมีรูปแบบดังนี้

[aObject aMethodWithParameter1:aValue parameter2:bValue parameter3:cValue ...];

และเวลาที่เราอ้างอิงถึงชื่อเมธอด เราจะอ้างอิงชื่อเมธอดในรูป

aMethod:parameter2:parameter3:

ซึ่งจะเห็นได้ว่า เมธอดจะใช้ : เพื่อบอกตำแหน่งของพารามิเตอร์ และชื่อเมธอดก็จะถูกแบ่งเป็นหลายส่วนตามจำนวนพารามิเตอร์ที่รับเข้ามา ซึ่งการออกแบบภาษาแบบนี้ ก็จะทำให้เราสามารถรู้ได้ในทันที่อ่านชื่อเมธอดว่าต้องรับพารามิเตอร์อะไรมั่ง และพารามิเตอร์ตัวใดใช้เพื่ออะไร ตัวอย่างเช่น

setTitle: forState:

ซึ่งจะถูกใช้ในการกำหนดข้อความบนปุ่ม ในตัวอย่างที่ผ่านมาในบทก่อนหน้า โดยเมธอดนี้จะแบ่งออกเป็น 2 ส่วน คือ

setTitle:

ซึ่งจะรับพารามิเตอร์เข้ามาเพื่อนำไปใช้กำหนดข้อความ

forState:

ซึ่งจะใช้ระบุว่าเราจะให้ข้อความที่กำหนดในส่วนก่อนหน้านี้ แสดงบนปุ่มเมื่อปุ่มอยู่ในสถานะใด

การเรียกใช้เมธอดในภาษา Objective C จะเรียกว่าเป็น การส่ง message ไปยัง object โดย object ที่ถูกส่ง message ไปจะถูกเรียกว่าเป็น receiver

ค่าที่คืนกลับมาจากการเรียกเมธอด จะสามารถนำมาใช้งานได้เหมือนเป็นค่าๆหนึ่ง ซึ่งเราสามารถนำมากำหนดค่าให้กับตัวแปร หรือนำไปส่งต่อให้กับอีกเมธอดนึง ตัวอย่างเช่น

```
int ab = [myObject multiplyA:a withB:b];

int c = [myObject squareRootA:[myObject multiplyA:a withB:b]];
```

ซึ่งการเรียกใช้งานเมธอดในลักษณะนี้ อาจจะดูสับสนในตอนแรก แต่เมื่อเราคุ้นเคยแล้ว เราจะเห็นว่าการเรียกเมธอดในลักษณะจะง่ายกว่าการใช้งานเมธอดในแบบเดิมๆ

การสร้าง OBJECT

1. **มี 2 รูปแบบหลักๆ**

   (1) Class Method : จะอยู่ในรูป

   [ClassName methodName:...];

   (2) Init Method : จะอยู่ในรูป

   [[ClassName alloc] initWith:...];

2. **จะมีข้อแตกต่างกันเล็กน้อยในเรื่องของ memory management**

ในส่วนนี้เราจะมาพูดถึงเรื่องของการสร้าง object กัน ภาษา Objective C เป็นภาษาเชิงวัตถุ ดังนั้นการทำงานต่างๆ ก็จะทำในมุมมองของ การทำงานร่วมกันระหว่างวัตถุต่างๆ ภายในโปรแกรม ซึ่งตรงนี้ หากใครที่เคยเขียนโปรแกรมด้วยภาษาเชิงวัตถุมามั่งแล้ว ก็คงจะคุ้นเคยเป็นอย่างดี แต่สำหรับใครที่ยังไม่เคยเขียนโปรแกรมในลักษณะนี้มาก่อน ขอแนะนำให้ลองไปหาหนังสือที่สอนการเขียนโปรแกรมเชิงวัตถุมาศึกษาเสียก่อนที่จะอ่านต่อไปจากจุดนี้

การสร้างวัตถุของคลาสใดๆขึ้นมาในภาษา Objective C นั้น มีด้วยกัน 2 วิธีหลักๆ ดังนี้

1. Class method ในคลาสบางคลาสจะมีการกำหนดคลาสเมธอดเอาไว้สำหรับใช้ในการสร้าง object ของคลาสนี้ขึ้นมา ตัวอย่างเช่น

[NSString stringWithFormat:...];

2. Init method การสร้าง object ของคลาส โดยปกติ จะต้องทำการจองพื้นที่บนหน่วยความจำ จากนั้นก็ทำการสร้าง object ลงบนหน่วยความจำที่จองเอาไว้ โดยจะมีรูปแบบการใช้งานดังนี้

[[MyObject alloc] init];

โดยที่หลายๆคลาส อาจจะมีเมธอด init ในหลายๆรูปแบบ เพื่อรับค่าเริ่มต้นเข้ามาเพื่อใช้ในการกำหนดค่าต่างๆของ object นี้ ตัวอย่างเช่น

[[NSString alloc] initWithFormat:...];

ซึ่งจะคล้ายกับคลาสเมธอดที่ยกตัวอย่างให้ดูข้างต้น ซึ่งการสร้าง object ทั้ง 2 แบบ จะมีข้อแตกต่างกันหลักๆอีกอย่างนึง นอกจากการเขียนโค๊ดที่สั้นลง ก็คือ ในเรื่องของการจัดการหน่วยความจำ (memory management) ซึ่งเราจะกล่าวถึงเรื่องนี้ในภายหลัง

การจัดการกับหน่อยความจำ

1. มี 3 วิธีหลัก คือ
   (1) Manual Reference Counting (MRC/ MRR) : คือการเขียน retain/release เอง
   (2) Automatic Reference Counting (ARC) : คือการให้ตัว compiler ทำการเพิ่มคำสั่ง retain/release ให้โดยอัตโนมัติ
   (3) Garbage Collection : มีการสร้าง object ขึ้นมาเพื่อจัดการกับ object อื่นๆ ที่ไม่ถูกใช้งานแล้ว จะมีเฉพาะบน Mac OS เท่านั้น
2. คำสั่ง retain คือการเพิ่ม reference count ขึ้น 1
3. คำสั่ง release คือการลด reference count ลง 1
4. คำสั่ง autorelease คือการลด reference count ลง 1 แต่จะรอช่วงระยะเวลานึงก่อนถึงจะลดลง
5. Object ที่สร้างด้วย Class Method โดยส่วนมากจะถูกเรียก autorelease มาแล้ว
6. Object ที่สร้างด้วย Init Method จะมี reference count เป็น 1 หลังจากสร้างขึ้นมา

การจัดการหน่วยความจำเป็นหัวข้อที่สำคัญอีกเรื่องนึงที่เราจำเป็นต้องศึกษา ในการเขียนโปรแกรมไม่ว่าจะด้วยภาษาใดก็ตาม เราจำเป็นที่จะต้องจองพื้นที่หน่วยความจำเพื่อใช้งาน และจะต้องคืนหน่วยความจำนั้นกลับให้กับระบบเมื่อไม่ใช้งานแล้ว นอกจากนี้เราจะต้องระวังให้ไม่ทำการคืนหน่วยความจำในส่วนที่ยังถูกใช้งานอยู่ ไม่เช่นนั้นอาจจะทำให้โปรแกรมทำงานผิดพลาดได้

ภาษา Objective C (เวอร์ชันที่ใช้อยู่ในขณะที่เขียนหนังสือนี้) นั้นมีระบบสำหรับทำการจัดการกับหน่วยความจำด้วยกัน 3 แบบด้วยกันดังนี้

1. Manual Reference Counting (MRC หรือ MRR - Manual Retain/Release) วิธีนี้ เราจะต้องทำการเขียนคำสั่ง retain/release เอง เพื่อกำหนด object lifetime

2. Automatic Reference Counting (ARC) วิธีนี้ ตัว compiler จะทำการเพิ่มคำสั่ง retain/release ลงในโค๊ดให้เหมาะสมเอง

3. Garbage Collection วิธีนี้ จะมี object “collector” ที่รับผิดชอบในการจัดการกับหน่วยความจำให้เองโดยอัตโนมัติ แต่วิธีนี้จะใช้ได้เฉพาะบนระบบปฏิบัติการ Mac OS เท่านั้น ไม่สามารถนำไปใช้กับ iOS ได้

โดยในที่นี้เราจะใช้ได้เพียงวิธีการที่ 1 กับ 2 เท่านั้น ซึ่งจริงๆแล้วทั้ง 2 วิธีนี้จะเหมือนกัน เพียงแต่ ARC นั้น ตัว compiler จะทำหน้าที่แทนเราในการเพิ่มคำสั่ง retain/release ซึ่งในตอนที่

เราสร้างโปรเจคขึ้นมาใหม่ จะมีตัวเลือกให้เราสามารถเปิดการใช้งาน ARC ได้ ดังนั้นในที่นี้จะขอพูดถึงเพียงวิธีการแรกเพียงอย่างเดียว

ภาษา Objective C จะใช้หลักการจัดการกับหน่วยความจำในแบบ Retain Count โดย object ทุกตัวจะมีค่า retain count อยู่ โดยถ้าค่านี้มากกว่า 0 object นี้ก็ยังอยู่ในหน่วยความจำต่อ แต่ถ้าหากค่านี้มีค่าเป็น 0 ก็จะทำให้ระบบทำการทำลาย object นี้ทิ้งและนำหน่วยความจำที่ object นี้เคยใช้งาน คืนให้กับระบบ โดยจะมีคำสั่งที่เกี่ยวข้องอยู่ 3 คำสั่งด้วยกันคือ

1. Retain จะทำการเพิ่มค่า retain count ของ object ขึ้นอีก 1 มีวิธีการใช้งานดังนี้

```
[myObject retain];
```

2. Release จะทำการลดค่า retain count ของ object ลงอีก 1 มีวิธีการใช้ดังนี้

```
[myObject release];
```

3. Autorelease คำสั่งนี้เป็นคำสั่งพิเศษ จะคล้ายกับคำสั่ง release เพียงแต่ object นั้นจะยังไม่ถูก release ในทันที แต่จะรอเวลาสักช่วงระยะนึงก่อนทำการเรียก release นั้นก็คือ หลังจากเราเรียก autorelease ไปแล้วเรายังจะสามารถใช้งาน object นั้นไปได้อีกช่วงเวลานึงก่อนที่ object นั้นจะโดน release

หลักการใช้งาน retain/release โดยทั่วไป ก็คือ เราจะมองเฉพาะภายในเมธอด เราจะทำการเรียก retain กับ object ทั้งหมดที่รับเข้ามาทำงานภายในเมธอดนั้นๆ และจะเรียก release กับ object ทั้งหมดที่เราเรียก retain ไป เมื่อสิ้นสุดการทำงานภายในเมธอดนั้นๆ โดยจะมีลักษณะดังนี้

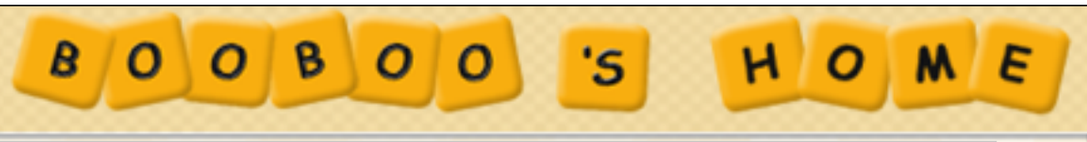
```
- (returnType)myMethodWithParam1:(MyObject1 *)a param2:(MyObject2 *)b
{
    [a retain];
    [b retain];
    // ... คำสั่งอื่นสำหรับทำงานในเมธอด
    [a release];
    [b release];
    return ....
}
```

สำหรับ object ที่เราทำการสร้างขึ้นมาใหม่นั้น ถ้าหากเราสร้างขึ้นด้วยคำสั่ง Init นั้น โดยทั่วไป จะมีค่า retain count เป็น 1 ทำให้เราจำเป็นต้องการทำเรียก release เองเมื่อไม่ใช้งานแล้ว

แต่ถ้าหากเราสร้าง object ขึ้นด้วยคลาสเมธอด โดยทั่วไป object นั้นจะถูกเรียก autorelease ไว้แล้ว ทำให้เราไม่จำเป็นต้องสั่ง release อีกเมื่อไม่ใช้งานแล้ว

เช่นเดียวกัน ถ้าในเมธอดของเรานั้น จำเป็นต้องสร้าง object ขึ้นมาใหม่ แล้วคืน object นั้นออกมาเมื่อเมธอดทำงานเสร็จ เราก็จะต้องทำการเรียก autorelease เพื่อให้เมธอดอื่นที่นำ object นี้ไปใช้งานต่อไม่ต้องเข้ามาจัดการกับ retain count ที่เปลี่ยนแปลงภายในเมธอดนี้ โดยการคืนค่า object ในลักษณะนี้ จะมีรูปแบบดังนี้

```
return [myObject autorelease];
```



Summary

สรุปสิ่งที่กล่าวถึงในบทนี้

1. ชนิดของไฟล์
2. ชนิดของตัวแปร
3. การเรียกใช้งานเมธอด
4. การสร้าง Object
5. การจัดการหน่วยความจำพื้นฐาน

ในบทนี้เราได้พูดถึงพื้นฐานต่างๆที่สำคัญของภาษา Objective C ซึ่งเป็นภาษาที่พัฒนามาจากภาษา C ทำให้พื้นฐานต่างๆของภาษาจะเหมือนกับของภาษา C ทั้งเรื่องชนิดตัวแปรพื้นฐาน ตัวดำเนินการพื้นฐานต่างๆ ไวยากรณ์ของภาษา ลำดับในการประมวลผลเครื่องหมายดำเนินการ เป็นต้น

ในภาษา Objective C ก็จะมีหลายส่วนที่เพิ่มเติมขึ้นมาเพื่อให้เหมาะสมกับพัฒนาโปรแกรมเชิงวัตถุ ซึ่งในบทนี้เราพูดเฉพาะพื้นฐานต่างๆของส่วนที่เพิ่มเติมขึ้นมานี้เท่านั้น เพื่อช่วยให้เราสามารถอ่านโค๊ดในภาษา Objective C ได้เข้าใจ สำหรับหัวข้อที่ยากขึ้นกว่านี้และต้องกล่าวถึงในรายละเอียดนั้นจะพูดถึงในบทถัดๆไป