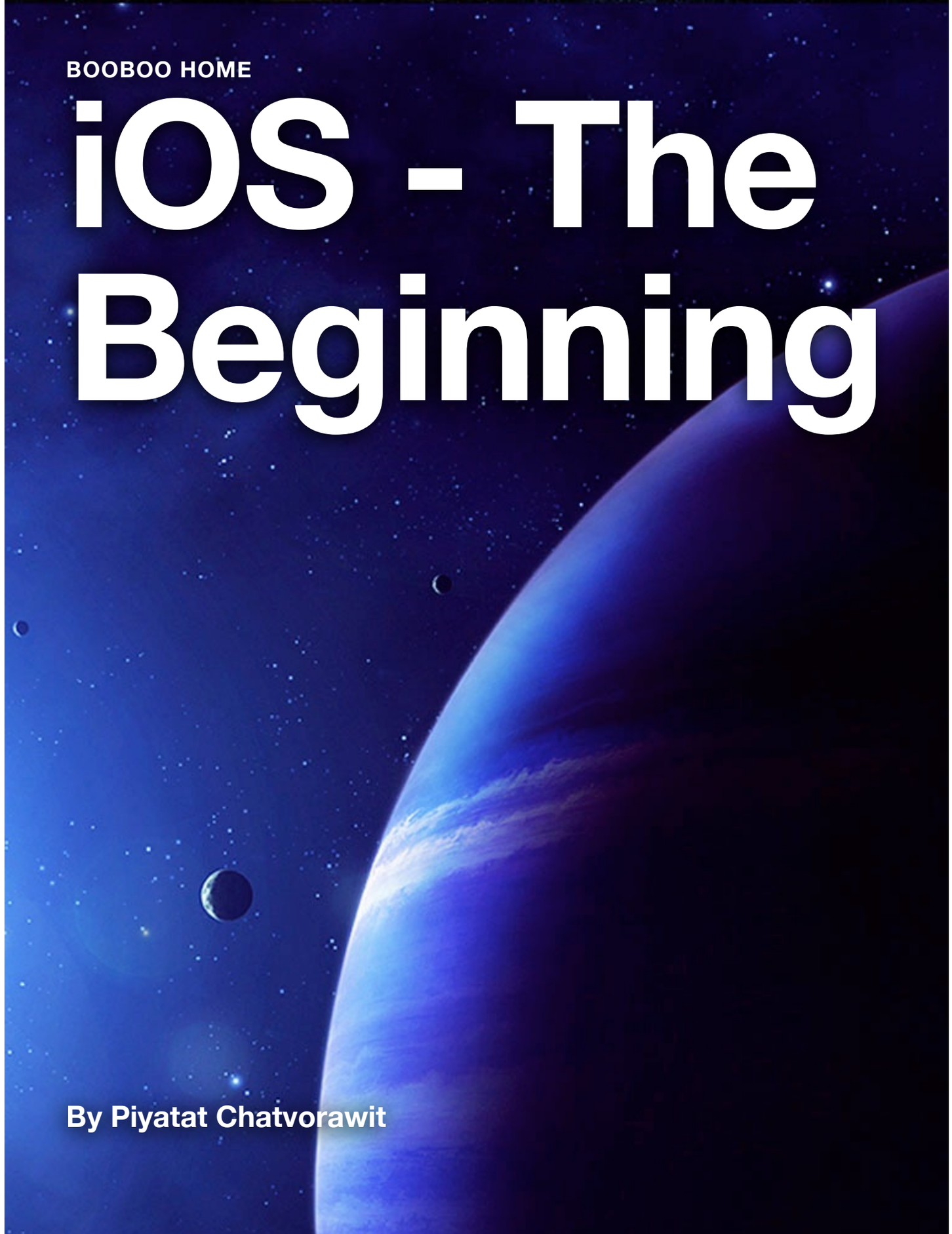

BOOBOO HOME
iOS - The Beginning
By Piyatat Chatvorawit

Chapter 6

iOS - UIKit Part 2: Basic Component

ในบทนี้เราจะมาพูดถึงคลาสต่างๆของ UIKit กันต่อ โดยเราจะเอาคลาสที่เราจะได้ใช้งานบ่อยๆมาอธิบาย ซึ่งได้แก่ คลาส UILabel, UIControl, UITextField, UITextView และ UIButton ซึ่งคลาสเหล่านี้จะเป็นองค์ประกอบพื้นฐานสำหรับใช้ในการติดต่อกับผู้ใช้งาน

UILABEL

1. เป็น subclass ของ UIView
2. ใช้สำหรับแสดงข้อความ โดยสามารถแสดงข้อความแบบหลายบรรทัดได้
3. โดยปกติจะไม่รองรับ input จากผู้ใช้

คลาสแรกที่เราจะพูดถึงกันในบทนี้คือคลาส UILabel ซึ่งจะถูกนำมาใช้ในการแสดงผลข้อความในลักษณะที่อ่านได้อย่างเดียว โดยจะสามารถแสดงผลได้ทั้งแบบบรรทัดเดียวหรือหลายบรรทัด UILabel จะถูกนำมาใช้บ่อยครั้งในการแสดงข้อความเพื่อติดต่อกับผู้ใช้งาน แต่จะไม่ค่อยมีการนำมาใช้เพื่อรับ input จากผู้ใช้ เนื่องจากจะมีคลาสอื่นที่เหมาะสมกว่าอยู่แล้วนั่นคือ UIButton ที่เราจะกล่าวถึงในภายหลัง

คลาส UILabel นั่นจะเป็นคลาสที่ subclass มาจากคลาส UIView ดังนั้นคุณสมบัติต่างๆของ UIView ก็จะสามารถนำมากำหนดให้กับ UILabel ได้เช่นกัน ดังนั้นเราจะขอไม่พูดถึงคุณสมบัติและเมธอดที่มีอยู่ใน superclass แล้ว

คุณสมบัติต่างๆที่สำคัญของคลาส UILabel จะมีดังนี้

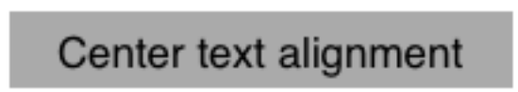
Property	Type	Description
text	NSString	ข้อความที่จะแสดง
font	UIFont	font ของตัวอักษรที่จะแสดง
minimumFontSize	CGFloat	ขนาดของตัวอักษรที่เล็กที่สุดที่จะแสดงผล จะถูกใช้คู่กับ adjustsFontSizeToFitWidth โดยค่า default คือ 0.0
textColor	UIColor	สีของตัวอักษร
textAlignment	UITextAlignment	รูปแบบการจัดวางของข้อความ
lineBreakMode	UILineBreakMode	ใช้กำหนดรูปแบบการตัดข้อความเพื่อขึ้นบรรทัดใหม่
numberOfLines	NSInteger	จำนวนบรรทัดมากที่สุดที่จะใช้แสดงผล โดยค่า default จะเป็น 1 แต่ถ้าต้องการให้แสดงผลโดยไม่จำกัดบรรทัดจะกำหนดให้เป็น 0
shadowColor	UIColor	สีของเงาของตัวอักษร โดยค่า default จะเป็น nil นั้นคือ ไม่มีเงา
shadowOffset	CGSize	offset ของเงาสำหรับตัวอักษร โดยค่า default คือ (0, -1) หรือคือ แสดงเงาเหนือตัวอักษร 1 pixel
highlightedTextColor	UIColor	สีของตัวอักษร เมื่อ UILabel ถูกกำหนดค่า highlighted เป็น YES
highlighted	BOOL	สถานะของ highlight ของตัวอักษร
adjustsFontSizeToFitWidth	BOOL	กำหนดให้ขนาดของตัวอักษรสามารถถูกเปลี่ยนเพื่อให้สามารถแสดงผลในขอบเขตที่กำหนดได้หรือไม่ โดยค่า default คือ NO
baselineAdjustment	UIBaselineAdjustment	กำหนดการเปลี่ยนแปลง baseline ของข้อความเมื่อมีการเปลี่ยนแปลงขนาดของตัวอักษร
userInteractionEnabled	BOOL	กำหนดให้สามารถรับ input จากผู้ใช้งานได้หรือไม่ โดยค่า default คือ NO

ต่อไปจะขออธิบายเกี่ยวกับชนิดของข้อมูลบางประเภทเพิ่มเติม เพื่อให้สะดวกในการนำไปใช้งาน

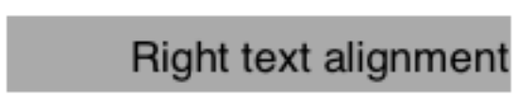
UITextAlignment จะมีค่าที่เป็นไปได้ดังนี้

1. *UITextAlignmentLeft* - กำหนดให้ข้อความจัดชิดด้านซ้ายมือ

2. *UITextAlignmentCenter* - กำหนดให้จัดข้อความไว้ตรงกลาง

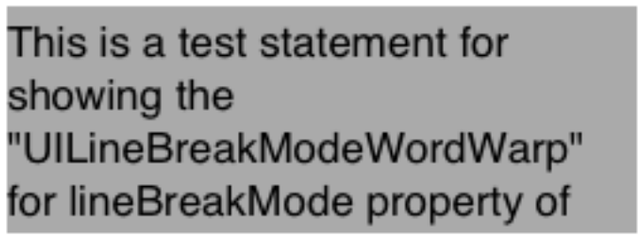


3. *UITextAlignmentRight* - กำหนดให้จัดข้อความชิดด้านขวามือ

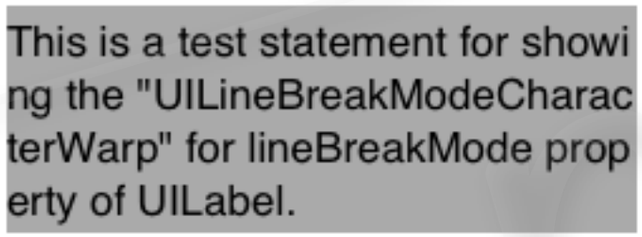


UILineBreakMode จะมีค่าที่เป็นไปได้ดังนี้

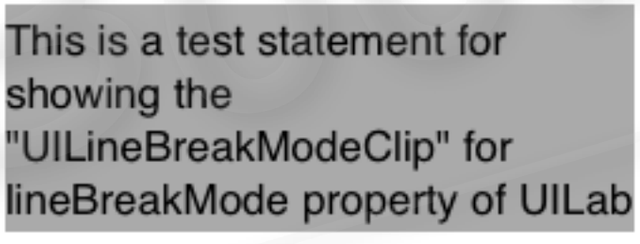
1. *UILineBreakModeWordWarp* - กำหนดให้มีการตัดข้อความตามคำ



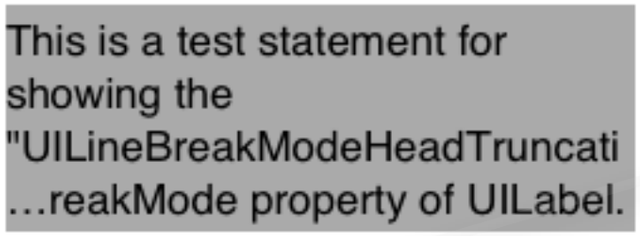
2. *UILineBreakModeCharacterWrap* - กำหนดให้มีการตัดข้อความตามตัวอักษร



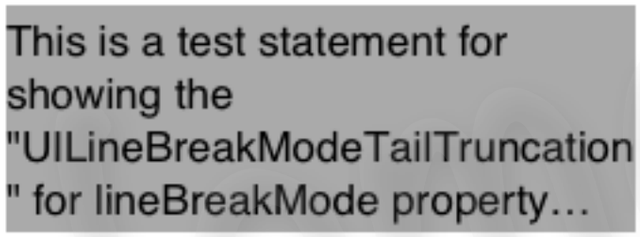
3. *UILineBreakModeClip* - กำหนดให้มีการตัดข้อความที่เกินพื้นที่แสดงผลออก



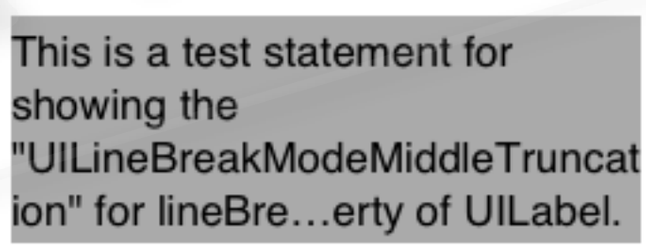
4. *UILineBreakModeHeadTruncation* - กำหนดให้มีการตัดข้อความ (ถ้าจำเป็น) จากตำแหน่งด้านหน้าของข้อความ



5. *UILineBreakModeTailTruncation* - กำหนดให้มีการตัดข้อความ (ถ้าจำเป็น) จากตำแหน่งด้านหลังของข้อความ

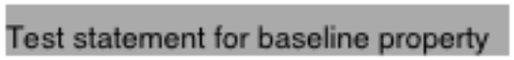


6. *UILineBreakModeMiddleTruncation* - กำหนดให้มีการตัดข้อความ (ถ้าจำเป็น) จากส่วนกลางของข้อความ



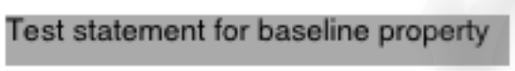
UIBaselineAdjustment จะมีค่าที่เป็นไปได้ดังนี้

1. *UIBaselineAdjustmentAlignBaselines* - จัดข้อความโดยอ้างอิงกับตำแหน่งของ baseline (เส้นบรรทัด)



2. *UIBaselineAdjustmentAlignCenters* - จัดข้อความโดยอ้างอิงกับตำแหน่งกึ่งกลางของพื้นที่แสดงผล

3. *UIBaselineAdjustmentAlignNone* - จัดข้อความจากมุมบนซ้ายของพื้นที่แสดงผล



UIFont จะเป็นคลาสที่ใช้สำหรับกำหนดรายละเอียดเกี่ยวกับ font และขนาดของตัวอักษร ซึ่งโดยปกติเราจะสร้าง object ของคลาสนี้โดยใช้คลาสเมธอด ซึ่งคลาสเมธอดที่ใช้ในการสร้าง object จะมีดังนี้

1. + *(UIFont *)systemFontOfSize:(CGFloat)fontSize* สร้าง UIFont โดยใช้ font ที่ถูกใช้ในระบบ และขนาดที่กำหนด

2. + *(UIFont *)boldSystemFontOfSize:(CGFloat)fontSize* สร้าง UIFont โดยใช้ font ของระบบที่เป็นตัวหนา และขนาดที่กำหนด

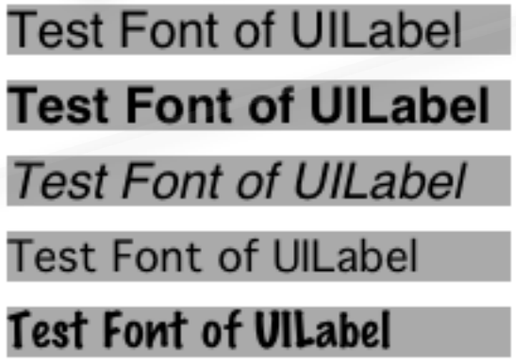
3. + *(UIFont *)italicSystemFontOfSize:(CGFloat)fontSize* สร้าง UIFont โดยใช้ font ของระบบที่เป็นตัวเอียง และขนาดที่กำหนด

4. + *(UIFont *)fontWithName:(NSString *)fontName size: (CGFloat)fontSize* สร้าง UIFont จากชื่อ font และขนาดที่กำหนด

โดยเราสามารถดึงชื่อ font ทั้งหมดของระบบออกมาได้โดยใช้เมธอดดังนี้

1. + *(NSArray *)familyNames* คืนค่า array ของชื่อ font family ของระบบทั้งหมดออกมา

2. + *(NSArray *)fontNamesForFamilyName:(NSString *)familtyName* คืนค่า array ของชื่อ font ทั้งหมดของระบบสำหรับ font family ที่ระบุออกมา



UIColor จะเป็นคลาสที่ใช้ในการกำหนดสี และความโปร่งใส ซึ่งจะมีค่าสีที่ถูกกำหนดมาไว้ให้ใช้อยู่แล้วจำนวนหนึ่ง โดยสามารถสร้างขึ้นมาโดยใช้คลาสเมธอด ดังนี้

1. + *(UIColor *)blackColor*

2. + *(UIColor *)darkGrayColor*

3. + *(UIColor *)lightGrayColor*

4. + *(UIColor *)whiteColor*

5. + *(UIColor *)grayColor*

6. + *(UIColor *)redColor*

7. + *(UIColor *)greenColor*

8. + *(UIColor *)blueColor*

9. + *(UIColor *)cyanColor*

10. + *(UIColor *)yellowColor*

11. + *(UIColor *)magentaColor*

12. + *(UIColor *)orangeColor*

13. + *(UIColor *)purpleColor*

14. + *(UIColor *)brownColor*

15. + *(UIColor *)clearColor* สีนี้จะเป็นสีโปร่งใส

และนอกจากนี้เราก็สามารถระบุค่าสีเองได้อีกด้วย ดังนี้

1. + *(UIColor *)colorWithWhite:(CGFloat)white alpha:(CGFloat)alpha*

2. + *(UIColor *)colorWithHue:(CGFloat)hue saturation:(CGFloat)saturation brightness:(CGFloat)brightness alpha:(CGFloat)alpha*

3. + *(UIColor *)colorWithRed:(CGFloat)red green:(CGFloat)green blue:(CGFloat)blue alpha:(CGFloat)alpha*

4. + *(UIColor *)colorWithPatternImage:(UIImage *)image* จะสร้าง UIColor สำหรับสร้างสีที่เป็นรูปซ้ำๆกัน

เนื่องจาก UILabel จะถูกนำมาใช้หลักๆเพียงแค่แสดงข้อความให้ผู้ใช้เท่านั้น และมันจะไม่ค่อยมีการเปลี่ยนแปลงใดๆเกิดขึ้นกับ UILabel ดังนั้นเมธอดของคลาสนี้จะไม่มีเมธอดที่จะให้ถูกเรียกใช้งานโดยตรง นอกจากเมธอดที่สืบทอดมาจาก superclass เราก็จะขอไม่พูดถึงในที่นี้

UICONTROL

1. เป็น **subclass** ของ **UIView**
2. เป็นคลาสพื้นฐานสำหรับ **control object** อื่นๆ **(object** ที่ใช้ในการรับ **input** จากผู้ใช้**)**
3. ทำหน้าที่เก็บสถานะของ **object** และผูกเมธอดกับ **event** ของ **object**

คลาสต่อมาที่เราจะนำมาพูดถึงในบทนี้คือ UIControl ซึ่งเป็นคลาสที่ subclass มาจาก UIView เช่นเดียวกับ UILabel แต่เราจะไม่นำคลาส UIControl มาใช้โดยตรง เนื่องจาก UIControl จะเป็นคลาสพื้นฐานสำหรับคลาสต่างๆ (control object เช่น UIButton, UITextField, ...) ที่ใช้ในการรับ input จากผู้ใช้ โดย UIControl จะมี property และเมธอดพื้นฐานต่างๆที่จะใช้ใน control object ต่างๆ

การทำงานหลักๆของ UIControl ก็คือการส่ง message ไปยัง target object เมื่อเกิดเหตุการณ์ที่กำหนดไว้ขึ้น อย่างเช่น เมื่อผู้ใช้กดปุ่มและยกนิ้วขึ้น เป็นต้น และการควบคุมสถานะต่างๆของ control object อย่างเช่น object อยู่ในสถานะ highlight, edit เป็นต้น

คุณสมบัติต่างๆที่สำคัญของคลาส UIControl จะมีดังนี้

Property	Type	Description
state	UIControlState	สถานะของ control object
enabled	BOOL	กำหนดให้ control object รับ input จากผู้ใช้หรือไม่ ค่า default คือ YES
selected	BOOL	กำหนดให้ control object มีสถานะเป็น selected หรือไม่ ค่า default คือ YES
highlighted	BOOL	กำหนดให้ control object มีสถานะเป็น highlighted หรือไม่ ค่า default คือ NO
contentVerticalAlignment	UIControlContentVerticalAlignment	กำหนดการจัดวางของ control object ในแนวตั้ง
contentHorizontalAlignment	UIControlContentHorizontalAlignment	กำหนดการจัดวางของ control object ในแนวนอน
tracking	BOOL	มีค่าเป็น YES ถ้า control object ทำการ tracking touch อยู่
touchInside	BOOL	มีค่าเป็น YES ถ้าผู้ใช้ touch ภายในขอบเขตของ control object

โดยค่าชนิดข้อมูลที่ยังไม่ได้กล่าวถึงมาก่อนจะมี

UIControlState จะเป็นค่า bit-mask ของค่า control state ที่เป็นไปได้ ซึ่งจะมีดังนี้

1. *UIControlStateNormal*
2. *UIControlStateHighlighted*
3. *UIControlStateDisabled*
4. *UIControlStateSelected*
5. *UIControlStateApplication*
6. *UIControlStateReserved*

ค่า state อาจจะเป็นได้มากกว่า 1 ค่าพร้อมกัน อย่างเช่น

```
UIControlStateHighlighted | UIControlStateSelected
```

โดยการตรวจสอบนั้นจะทำโดยการนำค่าไป & กับ state ที่ต้องการทดสอบ อย่างเช่น

```
if (controlObject.state & UIControlStateHighlighted)
{
    // Do something ...
}
```

UIControlContentVerticalAlignment จะมีค่าที่สามารถกำหนดได้ดังนี้

1. *UIControlContentVerticalAlignmentCenter* จัด content ไว้ตรงกลางตามแนวตั้ง

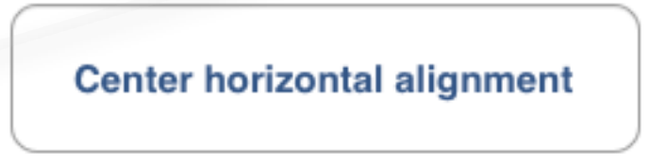
2. *UIControlContentVerticalAlignmentTop* จัด content ชิดด้านบน

3. *UIControlContentVerticalAlignmentBottom* จัด content ชิดด้านล่าง

4. *UIControlContentVerticalAlignmentFill* จัด content ให้เต็มพื้นที่แสดงผลในแนวตั้ง

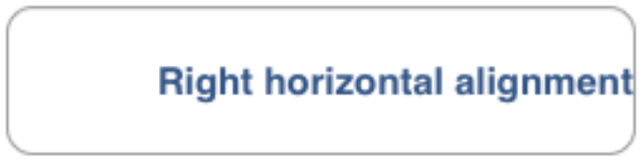
UIControlContentHorizontalAlignment จะมีค่าที่สามารถกำหนดได้ดังนี้

1. *UIControlContentHorizontalAlignmentCenter* จัด content ไว้ตรงกลางตามแนวนอน

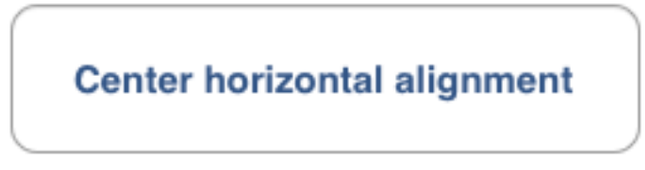


2. *UIControlContentHorizontalAlignmentLeft* จัด content ชิดด้านซ้าย

3. *UIControlContentHorizontalAlignmentRight* จัด content ชิดด้านขวา



4. *UIControlContentHorizontalAlignmentFill* จัด content ให้เต็มพื้นที่แสดงผลในแนวนอน



สำหรับเมธอดที่สำคัญๆของคลาส UIControl ที่เราจะได้ใช้กับ control object ต่างๆนั้นจะมีดังนี้

1. - *(void)addTarget:(id)target action:(SEL)action forControlEvents:(UIControlEvents)events* เมธอดนี้จะใช้ในการเพิ่มเมธอดและ target object ให้กับ events ที่ระบุ

2. - *(void)removeTarget:(id)target action:(SEL)action forControlEvents:(UIControlEvents)events* เมธอดนี้จะใช้ในการลบเมธอดและ target object ออกมาจาก events ที่ระบุ

3. - *(NSArray *)actionsForTarget:(id)target forControlEvent:(UIControlEvents)events* เมธอดนี้จะคืน array ของชื่อของเมธอด (NSString) ของ target object ที่ผูกไว้กับ events ที่ระบุ โดยเราสามารถนำชื่อเมธอดมาแปลงให้เป็นชนิด SEL ได้ (เพื่อให้สามารถเรียกเมธอดใช้งานได้) โดยใช้เมธอด NSSelectorFromString

4. - *(NSSet *)allTargets* คืนค่า array ของ target object ทั้งหมดที่ผูกไว้กับ control object นี้

5. - *(UIControlEvents)allControlEvents* คืนค่า events ทั้งหมดที่เกิดขึ้นได้สำหรับ control object นี้

โดยค่าของ **UIControlEvents** จะเป็นค่า bit-mask ซึ่งค่าที่เป็นไปได้จะมีดังนี้

1. *UIControlEventTouchDown*
2. *UIControlEventTouchDownRepeat*
3. *UIControlEventTouchDragInside*
4. *UIControlEventTouchDragOutside*
5. *UIControlEventTouchDragEnter*
6. *UIControlEventTouchDragExit*
7. *UIControlEventTouchUpInside*
8. *UIControlEventTouchUpOutside*
9. *UIControlEventTouchCancel*
10. *UIControlEventValueChanged*
11. *UIControlEventEditingDidBegin*
12. *UIControlEventEditingChanged*
13. *UIControlEventEditingDidEnd*
14. *UIControlEventEditingDidEndOnExit*
15. *UIControlEventAllTouchEvents*
16. *UIControlEventAllEditingEvents*
17. *UIControlEventApplicationReserved*
18. *UIControlEventSystemReserved*

19. *UIControlEventAllEvents*

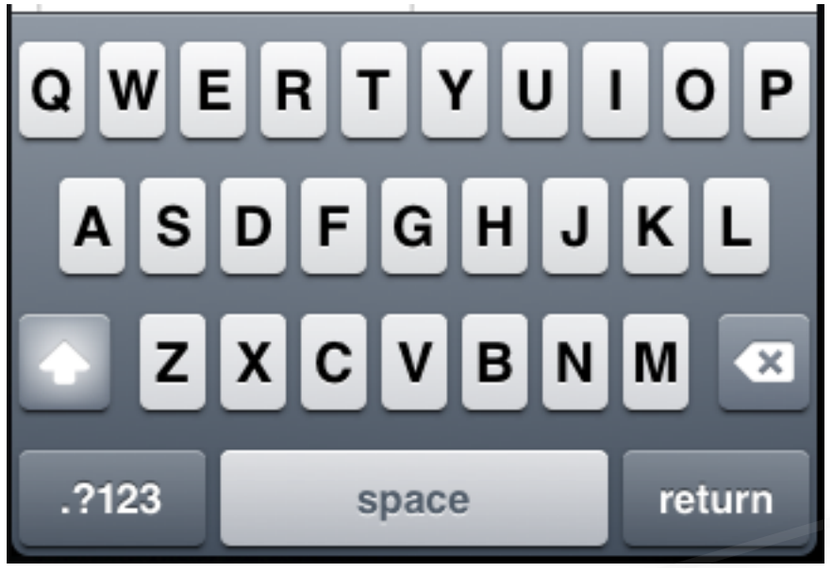
UITEXTFIELD

1. เป็น **subclass** มาจาก **UIControl**
2. ทำหน้าที่เป็นกล่องข้อความรับ **input** จากผู้ใช้ผ่านทาง **keyboard**
3. การเขียนควบคุม **keyboard** จะแตกต่างกันเล็กน้อยสำหรับ **iPhone** และ **iPad**

คลาสต่อไปที่เราจะพูดถึงกันก็คือคลาส UITextField ซึ่งจะมีลักษณะคล้ายๆกับกล่องข้อความ ใช้เพื่อรับ input จากผู้ใช้ โดยเมื่อผู้ใช้กดที่ UITextField ระบบก็จะทำการแสดง keyboard ขึ้นมา เพื่อให้ผู้ใช้พิมพ์ข้อความลงมายัง UITextField โดยเราสามารถที่จะกำหนดรูปแบบของ keyboard ที่จะแสดงขึ้นมาให้ผู้ใช้พิมพ์ได้ โดย keyboard ที่แสดงขึ้นมานั้น จะมีลักษณะไม่เหมือนกันสำหรับบน iPhone และ iPad เนื่องจากพื้นที่แสดงผลของหน้าจอ

โดยจุดที่ต่างกันหลักๆนอกจากการจัดวางตัวอักษรบน keyboard แล้ว บน iPad จะมีปุ่มสำหรับซ่อน keyboard อยู่ ในขณะที่บน iPhone นั้นเราจะต้องรับผิดชอบในการซ่อน keyboard เอง

คลาส UITextField นั่นเป็น subclass ของคลาส UIControl ซึ่งคลาส UIControl จะเป็น subclass ของคลาส UIView มาอีกที นอกจากนั้น UITextField ยังได้ implement property ต่างๆที่กำหนดไว้ใน protocol UITextInputTraits อีกด้วย ซึ่งจะถูกนำมาใช้ในการกำหนดลักษณะของ keyboard ที่จะปรากฏขึ้นมาเพื่อรับ input จากผู้ใช้



คุณสมบัติต่างๆที่สำคัญของคลาส UITextField จะมีดังนี้

Property	Type	Description
text	NSString	ข้อความที่อยู่ในกล่องข้อความ
placeholder	NSString	ข้อความที่จะแสดงเมื่อไม่มีข้อความอยู่ในกล่องข้อความ
font	UIFont	font ของตัวอักษรที่แสดงในกล่องข้อความ
textColor	UIColor	สีของตัวอักษร
textAlignment	UITextAlignment	ตำแหน่งการจัดวางข้อความในกล่องข้อความ
adjustsFontSizeToFitWidth	BOOL	กำหนดให้ทำการลดขนาดของตัวอักษรลงให้พอดีกับกล่องข้อความหรือไม่
minimumFontSize	CGFloat	ขนาดของตัวอักษรที่เล็กที่สุดที่จะแสดงในกล่องข้อความ
editing	BOOL	ใช้บอกว่า UITextField ถูก edit อยู่หรือไม่ โดยค่านี้เราจะสามารถอ่านได้อย่างเดียว
clearsOnBeginEditing	BOOL	กำหนดให้ทำการลบข้อความเดิมในกล่องข้อความก่อนจะเริ่ม edit หรือไม่
borderStyle	UITextBorderStyle	กำหนดรูปแบบขอบของกล่องข้อความ
background	UIImage	รูปพื้นหลังของกล่องข้อความ
disabledBackground	UIImage	รูปพื้นหลังของกล่องข้อความ ซึ่งจะแสดงเมื่อ UITextField ถูก disable

Property	Type	Description
clearButtonMode	UITextFieldViewMode	กำหนดการแสดงผลของปุ่มลบข้อความ ที่แสดงบนกล่องข้อความ
leftView	UIView	UIView สำหรับแสดงผลทางด้านซ้ายของกล่องข้อความ
leftViewMode	UITextFieldViewMode	กำหนดการแสดงผลของ leftView
rightView	UIView	UIView สำหรับแสดงผลทางด้านขวาของกล่องข้อความ
rightViewMode	UITextFieldViewMode	กำหนดการแสดงผลของ rightView
inputView	UIView	กำหนด input ของกล่องข้อความ โดยถ้าหากไม่กำหนด จะแสดงเป็น keyboard ธรรมดา
inputAccessoryView	UIView	กำหนด UIView สำหรับแสดงเหนือ inputView
delegate	id<UITextFieldDelegate>	delegate ของ UITextField

ข้อมูลชนิดใหม่ที่ยังไม่ได้กล่าวถึงมาก่อนก็จะมี

UITextBorderStyle จะมีค่าที่กำหนดได้ดังนี้

1. *UITextBorderStyleNone*

2. *UITextBorderStyleLine*

3. *UITextBorderStyleBezel*

4. *UITextBorderStyleRoundedRect*

UITextFieldViewMode จะมีค่าที่กำหนดได้ดังนี้

1. *UITextFieldViewModeNever* กำหนดให้ไม่แสดง

2. *UITextFieldViewModeWhileEditing* กำหนดให้แสดงเมื่อกล่องข้อความถูก edit อยู่

3. *UITextFieldViewModeUnlessEditing* กำหนดให้แสดงเมื่อกล่องข้อความไม่ถูก edit อยู่

4. *UITextFieldViewModeAlways* กำหนดให้แสดงตลอดเวลา

สำหรับ **UITextFieldDelegate** นั้น จะเป็น protocol ที่จะใช้ในการติดต่อระหว่าง UITextField กับโปรแกรมของเรา โดยเราจะต้องกำหนด object ที่ต้องการให้เป็น delegate ของ UITextField และเมื่อ UITextField เกิดการทำงานบางประการที่กำหนดไว้ ก็จะทำการเรียกเมธอดที่ประกาศไว้ใน protocol ของ delegate object โดยเรื่อง protocol และ delegate นั้นเราจะกล่าวถึงรายละเอียดกันอีกทีในบทต่อไป โดย UITextFieldDelegate นั้นจะมีเมธอดที่กำหนดไว้ดังนี้

Optional Method

1. - *(BOOL)textFieldShouldBeginEditing:(UITextField *)textField* เมธอดนี้จะถูกเรียกเมื่อผู้ใช้ทำการโฟกัสที่ UITextField โดยเมธอดนี้จะใช้กำหนดว่า UITextField จะถูก edit หรือไม่ (แสดง keyboard ขึ้นมาให้ผู้ใช้พิมพ์ลงในกล่องข้อความ) ถ้าคืนค่า YES จะหมายถึงให้ edit ได้ โดยค่า default จะเป็น YES หากเมธอดนี้ไม่ถูก implement ใน delegate object

2. - *(void)textFieldDidBeginEditing:(UITextField *)textField* เมธอดนี้จะถูกเรียกเมื่อ UITextField เปลี่ยนสถานะเป็น edit (หรือก็คือหลังจากเมธอด textFieldShouldBeginEditing: คืนค่า YES ออกมา)

3. - *(void)textFieldShouldEndEditing:(UITextField *)textField* เมธอดนี้จะถูกเรียกเมื่อ UITextField กำลังจะเสียโฟกัส โดยเมธอดนี้จะใช้กำหนดว่า UITextField จะเสียโฟกัสหรือไม่ (หรือคือจะทำการซ่อน keyboard) ถ้าโดยค่า default จะเป็น YES หากเมธอดนี้ไม่ถูก implement ใน delegate object

4. - *(void)textFieldDidEndEditing:(UITextField *)textField* เมธอดนี้จะถูกเรียกเมื่อ UITextField เสียโฟกัสแล้ว (หรือคือหลังจากเมธอด textFieldDidEndEditing: คืนค่า YES ออกมา)

5. - *(BOOL)textField:(UITextField *)textField shouldChangeCharactersInRange:(NSRange)range replacementString:(NSString *)string* เมธอดนี้จะถูกเรียกเมื่อข้อความใน UITextField ถูกเปลี่ยน (หรือคือเมื่อผู้ใช้พิมพ์ตัวอักษรใหม่ หรือลบตัวอักษร) โดยเมธอดนี้จะใช้กำหนดว่าข้อความใน UITextField จะถูกเปลี่ยนแปลงหรือไม่ โดยค่า range จะบอกตำแหน่งที่มีการเปลี่ยนแปลง ส่วน string คือข้อความที่จะเปลี่ยน โดยค่า default จะเป็น YES หากเมธอดนี้ไม่ถูก implement ใน delegate object

6. - *(BOOL)textFieldShouldClear:(UITextField *)textField* เมธอดนี้จะถูกเรียกเมื่อผู้ใช้กดปุ่ม clear บนกล่องข้อความ โดยเมธอดนี้จะกำหนดว่าข้อความที่มีอยู่ใน UITextField จะถูกลบออกหรือไม่ โดยค่า default จะเป็น YES หากเมธอดนี้ไม่ถูก implement ใน delegate object

7. - *(BOOL)textFieldShouldReturn:(UITextField *)textField* เมธอดนี้จะถูกเรียกเมื่อผู้ใช้กดปุ่ม return บน keyboard โดยเมธอดนี้จะกำหนดว่า UITextField ควรจะทำการประมวลผลปุ่ม return หรือไม่

สำหรับคุณสมบัติที่กำหนดไว้ใน UITextInputTraits นั้นจะมีดังนี้

Property	Type	Description
autocapitalizationType	UITextAutocapitalizationType	กำหนดรูปแบบการเปลี่ยนตัวอักษรเล็กใหญ่ของข้อความ
autocorrectionType	UITextAutocorrectionType	กำหนดรูปแบบการทำงานของ auto-correction
spellCheckingType	UITextSpellCheckingType	กำหนดวิธีการตรวจสอบคำที่พิมพ์ลงไป
enableReturnKeyAutomatically	BOOL	ถ้าเป็น YES ปุ่ม return จะถูก disable จนกว่าจะมีข้อความอยู่ในกล่องข้อความ โดยค่า default จะเป็น NO (คือ ปุ่ม return จะไม่ถูก disable)
keyboardAppearance	UIKeyboardAppearance	กำหนดรูปแบบของ keyboard
keyboardType	keyboardType	กำหนดชนิดของ keyboard
returnKeyType	returnKeyType	กำหนดชนิดของปุ่ม return บน keyboard
secureTextEntry	BOOL	กำหนดเป็น YES ถ้าต้องการให้ซ่อนข้อความที่พิมพ์ลงไป ค่า default จะเป็น NO

UITextAutocapitalizationType จะมีค่าที่สามารถกำหนดได้ดังนี้

1. *UITextAutocapitalizationTypeNone* ไม่ต้องแปลงเป็นตัวอักษรใหญ่

test statement

2. *UITextAutocapitalizationTypeWords* แปลงเฉพาะตัวอักษรแรกของแต่ละคำให้เป็นตัวอักษรใหญ่

Test Statement

3. *UITextAutocapitalizationTypeSentences* แปลงเฉพาะตัวอักษรแรกในแต่ละประโยคให้เป็นตัวอักษรใหญ่

Test statement

4. *UITextAutocapitalizationTypeAllCharacters* แปลงตัวอักษรทั้งหมดให้เป็นตัวอักษรใหญ่

TEST STATEM...

UITextAutocorrectionType จะมีค่าที่สามารถกำหนดได้ดังนี้

1. *UITextAutocorrectionTypeDefault* กำหนดให้ใช้ค่าตามที่กำหนดไว้ในระบบ

2. *UITextAutocorrectionTypeNo* ปิดการทำงานของ auto-correction

3. *UITextAutocorrectionTypeYes* เปิดการทำงานของ auto-correction

UITextSpellCheckingType จะมีค่าที่สามารถกำหนดได้ดังนี้

1. *UITextSpellCheckingTypeDefault* จะกำหนดให้ทำงานเมื่อ auto-correction ทำงาน

2. *UITextSpellCheckingTypeNo* ปิดการทำงานของ spell checking

3. *UITextSpellCheckingTypeYes* เปิดการทำงานของ spell checking

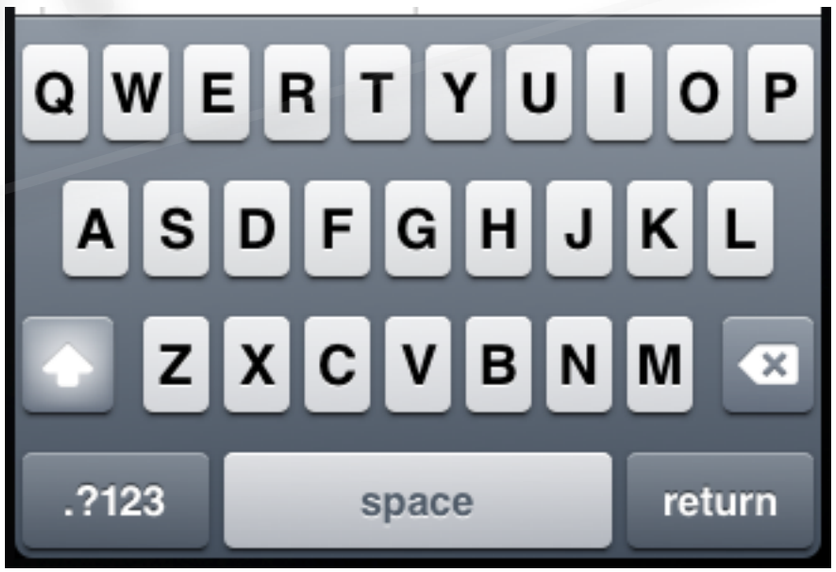
UIKeyboardAppearance จะมีค่าที่สามารถกำหนดได้ดังนี้

1. *UIKeyboardAppearanceDefault*

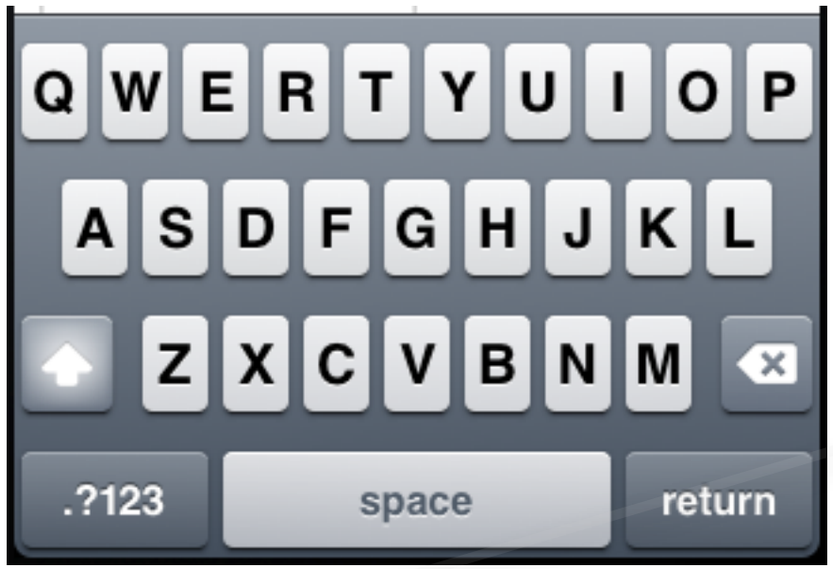
2. *UIKeyboardAppearanceAlert*

UIKeyboardType จะมีค่าที่สามารถกำหนดได้ดังนี้

1. *UIKeyboardTypeDefault*



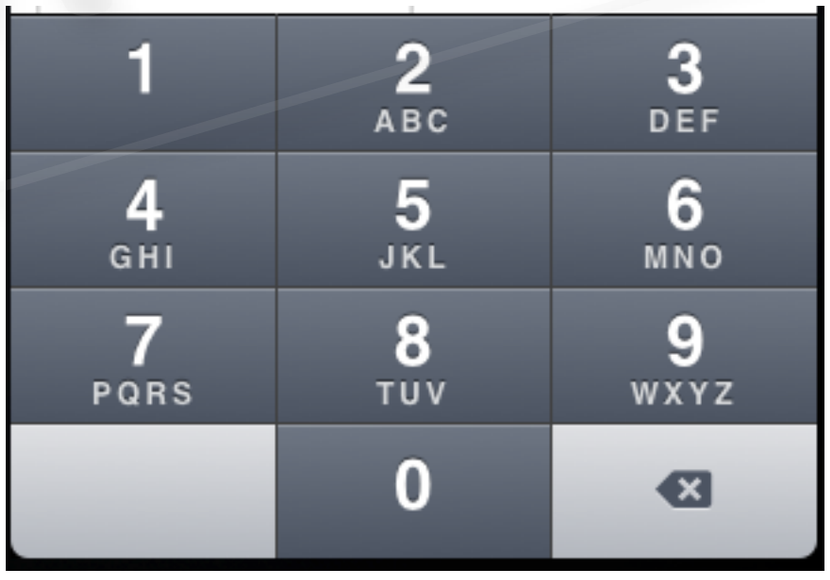
2. *UIKeyboardTypeASCIICapable*



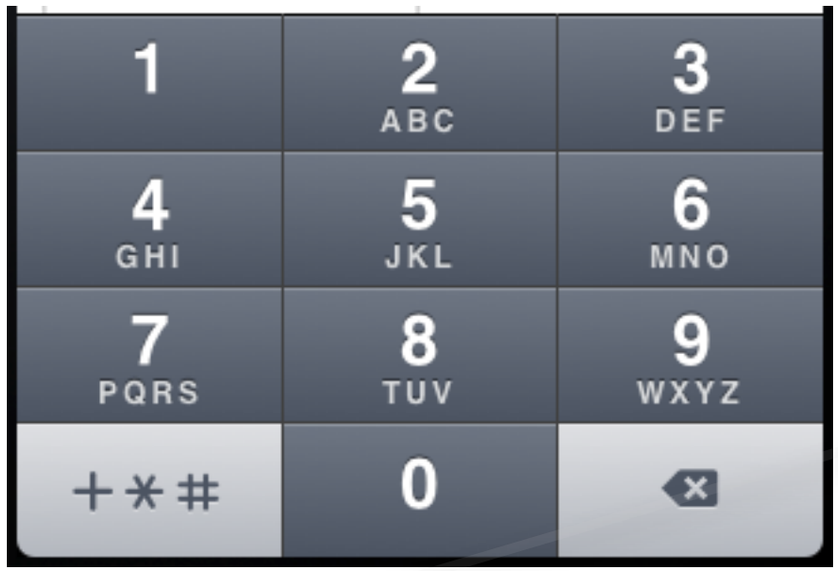
3. *UIKeyboardTypeNumbersAndPunctuation*

4. *UIKeyboardTypeURL*

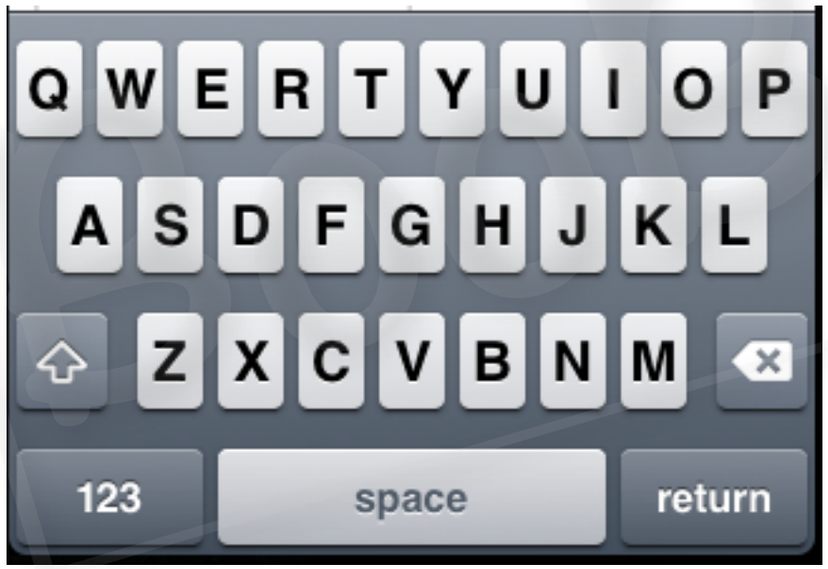
5. *UIKeyboardTypeNumberPad*



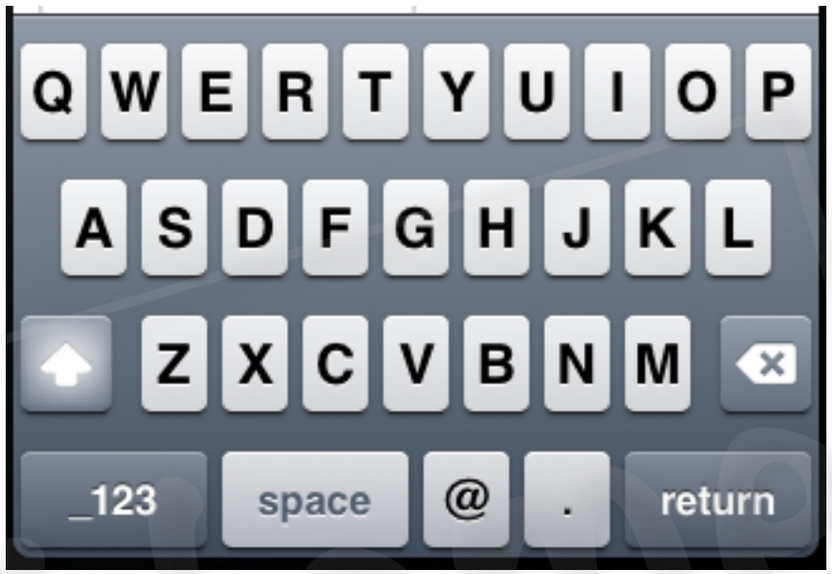
6. *UIKeyboardTypePhonePad*



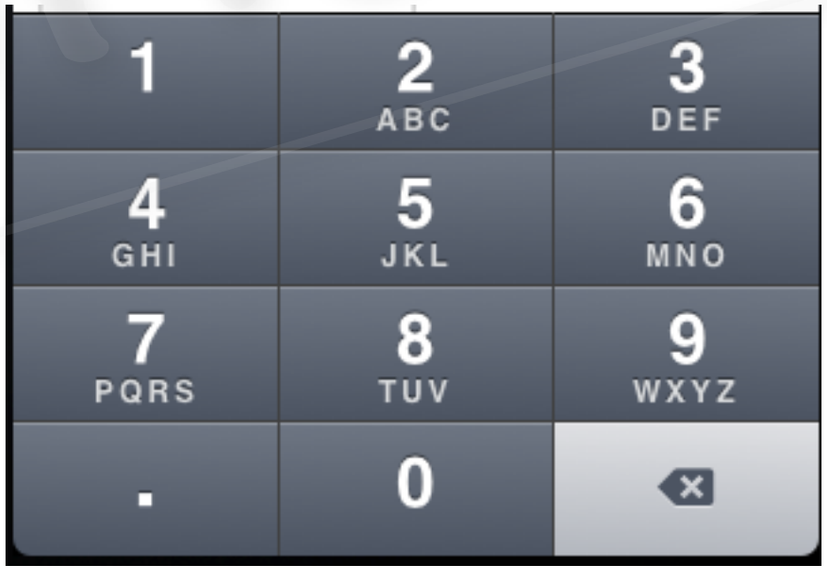
7. *UIKeyboardTypeNamePhonePad*



8. *UIKeyboardTypeEmailAddress*



9. *UIKeyboardTypeDecimalPad*

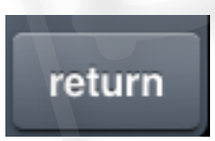


10. *UIKeyboardTypeTwitter*

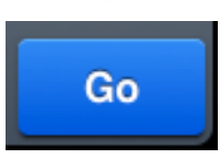
11. *UIKeyboardTypeAlphabet* ซึ่งจะมีค่าเหมือนกับ UIKeyboardTypeASCIICapable

UIReturnKeyType จะมีค่าที่สามารถกำหนดได้ดังนี้

1. *UIReturnKeyDefault*



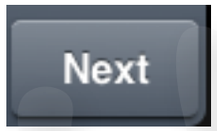
2. *UIReturnKeyGo*



3. *UIReturnKeyGoogle*

4. *UIReturnKeyJoin*

5. *UIReturnKeyNext*



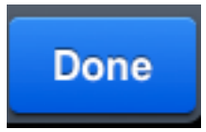
6. *UIReturnKeyRoute*

7. *UIReturnKeySearch*

8. *UIReturnKeySend*

9. *UIReturnKeyYahoo*

10. *UIReturnKeyDone*



11. *UIReturnKeyEmergencyCall*

UITEXTVIEW

1. **เป็น subclass ของ UIScrollView**
2. **ทำหน้าที่เป็นกล่องข้อความเพื่อรับข้อความจากผู้ใช้ แต่จะต่างจาก UITextField ตรงที่ UITextView จะรองรับข้อความหลายบรรทัดได้ และจะมี scroll หากข้อความเกินกว่าพื้นที่แสดงผล**

คลาส UITextView จะคล้ายกับคลาส UITextField แต่จะต่างกันตรงที่คลาส UITextView นั้นจะสามารถใช้กับข้อความหลายบรรทัดได้ และจะมี scroll bar ให้สามารถเลื่อนเข้าลงได้ นั้นคือเราสามารถพิมพ์ข้อความลงใน UITextView ได้กี่บรรทัดก็ได้ เช่นเดียวกับคลาส UITextField เราจะต้องจัดการเรื่องการซ่อน keyboard เอง สำหรับโปรแกรมที่ทำงานบน iPhone

คลาส UITextView จะทำการ subclass มาจากคลาส UIScrollView ซึ่งจะใช้ในการแสดงผล content ที่ขนาดใหญ่กว่าพื้นที่แสดงผล โดยเราจะกลับมาพูดถึงคลาสนี้กันอีกทีในภายหลัง

ถึงแม้ว่า UITextView จะไม่ได้ subclass มาจาก UITextField แต่เนื่องจากคลาสทั้ง 2 นี้จะทำหน้าที่คล้ายคลึงกัน คือเป็นกล่องข้อความสำหรับให้ผู้ใช้ทำการพิมพ์ข้อความลงในโปรแกรม ทำให้คลาสทั้ง 2 นี้มีคุณสมบัติ และเมธอดคล้ายๆกัน

โดยหน้าตาของ UITextView นั้นจะเป็นเหมือนกับ UILabel หรืออาจจะจะเรียกได้ว่าเป็น UILabel ที่สามารถแก้ไขข้อความได้

โดยคุณสมบัติที่สำคัญของ UITextView จะมีดังนี้

Property	Type	Description
text	NSString	ข้อความที่อยู่ในกล่องข้อความ
font	UIFont	font ของตัวอักษรที่แสดงในกล่องข้อความ
textColor	UIColor	สีของตัวอักษร
textAlignment	UITextAlignment	ตำแหน่งการจัดวางข้อความในกล่องข้อความ
editable	BOOL	กำหนดให้ผู้ใช้สามารถพิมพ์ข้อความลงใน UITextView ได้หรือไม่ โดยค่า default คือ YES
dataDetectorType	UIDataDetectorTypes	กำหนดชนิดของข้อมูลที่จะถูกแปลงให้เป็นปุ่มที่สามารถกดได้
selectedRange	NSRange	ระบุตำแหน่งข้อความที่ถูกเลือกอยู่
inputView	UIView	กำหนด input ของกล่องข้อความ โดยถ้าหากไม่กำหนด จะแสดงเป็น keyboard ธรรมดา
inputAccessoryView	UIView	กำหนด UIView สำหรับแสดงเหนือ inputView
delegate	id<UITextViewDelegate>	delegate ของ UITextView

สำหรับ **UIDataDetectorTypes** จะมีค่าที่สามารถกำหนดได้ดังนี้

1. *UIDataDetectorTypePhoneNumber*
2. *UIDataDetectorTypeLink*
3. *UIDataDetectorTypeAddress*
4. *UIDataDetectorTypeCalendarEvent*
5. *UIDataDetectorTypeNone*
6. *UIDataDetectorTypeAll*

โดยค่านี้จะใช้ในการบอกให้ UITextView ทำการแปลงข้อมูลที่ตรงกับชนิดที่กำหนดให้กลายเป็นปุ่มที่สามารถกดได้ ซึ่งเมื่อผู้ใช้ทำการกดที่ปุ่มนี้ก็จะเป็นการเรียกโปรแกรมที่เหมาะสมสำหรับจัดการกับข้อมูลชนิดนั้นๆ แต่คุณสมบัตินี้จะใช้ได้ก็ต่อเมื่อ UITextView อยู่ในสถานะที่แก้ไขข้อความไม่ได้ หรือก็คือต้องมีค่า editable เป็น NO นั้นเอง

ในส่วนของ **UITextViewDelegate** นั้นจะมีเมธอดที่กำหนดไว้จะคล้ายกับเมธอดที่กำหนดใน UITextFieldDelegate ดังนี้

Optional Method

1. - *(BOOL)textViewShouldBeginEditing:(UITextView *)textView*
เมธอดนี้จะใช้ในการกำหนดว่าจะให้ edit UITextView ได้หรือไม่ ค่า default จะเป็น YES

2. - *(void)textViewDidBeginEditing:(UITextView *)textView* เมธอดนี้จะถูกเรียกเมื่อ UITextView เริ่มทำการ edit

3. - *(BOOL)textViewShouldEndEditing:(UITextView *)textView* เมธอดนี้จะใช้ในการกำหนดว่า UITextView จะเสีย focus หรือไม่ โดยค่า default คือ YES

4. - *(void)textViewDidEndEditing:(UITextView *)textView* เมธอดนี้จะถูกเรียกเมื่อ UITextView เสีย focus ไปแล้ว

5. - *(BOOL)textView:(UITextView *)textView shouldChangeTextInRange:(NSRange)range replacementText:(NSString *)text* เมธอดนี้จะใช้กำหนดว่าข้อความในตำแหน่งที่ระบุจะสามารถถูกแทนที่ด้วยข้อความที่ระบุได้หรือไม่ โดยค่า default คือ YES

6. - *(void)textViewDidChange:(UITextView *)textView* เมธอดนี้จะถูกเรียกเมื่อผู้ใช้ทำการเปลี่ยนแปลงข้อความใน UITextView แต่จะไม่ถูกเรียกหากข้อความถูกเปลี่ยนแปลงผ่านทางโปรแกรมโดยตรง

7. - *(void)textViewDidChangeSelection:(UITextView *)textView* เมธอดนี้จะถูกเรียกเมื่อข้อความที่ถูกเลือกใน UITextView เปลี่ยนไป

สำหรับเมธอดของ UITextView ที่สำคัญจะมีดังนี้

1. - *(BOOL)hasText* จะคืนค่า YES ถ้า UITextView มีข้อความใดๆอยู่ภายใน

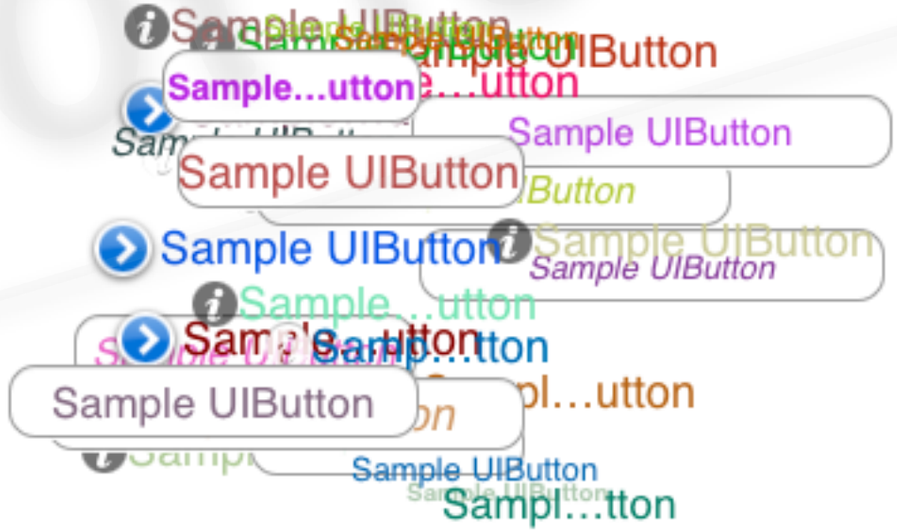
2. - *(void)scrollRangeToVisible:(NSRange)range* จะสั่งให้ UITextView เลื่อนไปยังตำแหน่งที่ระบุ

UIBUTTON

1. เป็น **subclass** มาจาก **UIControl**
2. ทำหน้าที่เป็นปุ่มให้ผู้ใช้กดเพื่อส่ง **event** ออกมาเพื่อเรียกเมธอดที่ระบุไว้ให้ทำงาน

คลาสสุดท้ายที่เราจะมาพูดถึงกันในบทนี้คือ คลาส UIButton ซึ่งจะมีลักษณะเป็นปุ่ม โดย UIButton จะต่างจากคลาส UITextField และ UITextView ที่พูดถึงก่อนหน้านี้ UIButton จะไม่มี delegate สำหรับใช้เรียกเมธอดต่าๆ เมื่อเกิด event บางอย่างขึ้น เราจะต้องทำการผูกเมธอดที่ต้องการให้กับ event ต่างๆของ UIButton เอง

คลาส UIButton จะ subclass มาจากคลาส UIControl เช่นเดียวกับคลาส UITextField ซึ่งเราจะใช้เมธอดต่างๆของ UIControl ในการผูกเมธอดกับ event ที่ต้องการ



คุณสมบัติต่างๆที่สำคัญของ UIButton จะมีดังนี้

Property	Type	Description
titleLabel	UILabel	UILabel สำหรับข้อความบนปุ่ม
buttonType	UIButtonType	ชนิดของปุ่ม
currentTitle	NSString	ข้อความที่แสดงอยู่บนปุ่ม
currentTitleColor	UIColor	สีของข้อความที่แสดงบนปุ่ม
currentTitleShadowColor	UIColor	สีของเงาของข้อความที่แสดงบนปุ่ม
currentImage	UIImage	รูปที่แสดงอยู่บนปุ่ม
currentBackgroundImage	UIImage	รูปที่ใช้เป็นพื้นหลังของปุ่ม
imageView	UIImageView	UIImageView ของปุ่ม
reversesTitleShadowWhenHighlighted	BOOL	กำหนดว่าเงาของข้อความจะเปลี่ยนเมื่อปุ่มอยู่ในสถานะ highlight หรือไม่ ค่า default คือ NO
adjustsImageWhenHighlighted	BOOL	กำหนดว่ารูปจะเปลี่ยนแปลงหรือไม่ เมื่อปุ่มอยู่ในสถานะ highlight ค่า default คือ YES
adjustsImageWhenDisabled	BOOL	กำหนดว่ารูปจะเปลี่ยนแปลงหรือไม่ เมื่อปุ่มอยู่ในสถานะ disable ค่า default คือ YES
showsTouchWhenHighlighted	BOOL	กำหนดให้ปุ่มแสดงรูปตอนถูกกดหรือไม่ ค่า default คือ NO
contentEdgeInsets	UIEdgeInsets	ระยะระหว่างขอบของพื้นที่แสดงผลกับตัวปุ่ม โดยค่า default คือ UIEdgeInsetsZero
titleEdgeInsets	UIEdgeInsets	ระยะระหว่างขอบของพื้นที่แสดงผลกับข้อความบนปุ่ม โดยค่า default คือ UIEdgeInsetsZero
imageEdgeInsets	UIEdgeInsets	ระยะระหว่างขอบของพื้นที่แสดงผลกับรูปบนปุ่ม โดยค่า default คือ UIEdgeInsetsZero

โดยค่า **UIButtonType** นั้นจะมีค่าที่เป็นไปได้ดังนี้

1. *UIButtonTypeCustom* โดยค่านี้จะทำให้ปุ่มเป็นเพียงพื้นที่สี่เหลี่ยมว่างๆเท่านั้น
2. *UIButtonTypeRoundedRect* 

3. *UIButtonTypeDetailDisclosure* 
4. *UIButtonTypeInfoLight*
5. *UIButtonTypeInfoDark* 
6. *UIButtonTypeContactAdd* 

สำหรับคลาส UIImage และ UIImageView จะพูดถึงอีกทีในภายหลัง โดย UIImage จะเป็น object ของรูปซึ่งจะใช้เก็บข้อมูลต่างๆเกี่ยวกับรูป ส่วน UIImageView คือคลาสที่ใช้ในการแสดงรูปบนหน้าจอโปรแกรม

ส่วน UIEdgeInsets จะเป็น struct ที่ประกอบด้วยตัวแปร 4 ตัวคือ top, bottom, left และ right ซึ่งตัวแปรทั้งหมดจะเป็นชนิด CGFloat ซึ่งใช้ในการระบุระยะห่างของ view กับพื้นที่แสดงผล โดยการประกาศตัวแปรชนิดนี้จะประกาศได้โดยใช้คำสั่ง

UIEdgeInsetsMake(top, left, bottom, right)

สำหรับเมธอดที่สำคัญต่างๆของ UIButton จะมีดังนี้

1. + (UIButton *)buttonWithType:(UIButtonType)buttonType เมธอดนี้เป็นคลาสเมธอด ซึ่งจะใช้ในการสร้าง object ของคลาส UIButton ตามประเภทที่ระบุ

2. - (void)setTitle:(NSString *)title forState:(UIControlState)state ใช้สำหรับกำหนดข้อความบนปุ่มสำหรับสถานะที่กำหนด

3. - (void)setTitleColor:(UIColor *)color forState:(UIControlState)state ใช้สำหรับกำหนดสีของข้อความบนปุ่มสำหรับสถานที่กำหนด

4. - (void)setTitleShadowColor:(UIColor *)color forState:(UIControlState)state ใช้สำหรับกำหนดสีของเงาของข้อความบนปุ่มสำหรับสถานะที่กำหนด

5. - (NSString *)titleForState:(UIControlState)state คืนค่าข้อความบนปุ่มสำหรับสถานะที่ระบุ

6. - (UIColor *)titleColorForState:(UIControlState)state คืนค่าสีของข้อความบนปุ่มสำหรับสถานะที่ระบุ

7. - (UIColor *)titleShadowColorForState:(UIControlState)state คืนค่าสีของเงาของข้อความบนปุ่มสำหรับสถานะที่ระบุ

8. - (void)setBackgroundImage:(UIImage *)image forState:(UIControlState)state ใช้สำหรับกำหนดรูปพื้นหลังของปุ่มสำหรับสถานะที่ระบุ

9. - (void)setImage:(UIImage *)image forState:(UIControlState)state ใช้สำหรับกำหนดรูปของปุ่มสำหรับสถานะที่ระบุ

10. - (UIImage *)backgroundImageForState:(UIControlState)state คืนค่ารูปพื้นหลังของปุ่มสำหรับสถานะที่ระบุ

11. - (UIImage *)imageForState:(UIControlState)state คืนค่ารูปของปุ่มสำหรับสถานะที่ระบุ

ซึ่งเราจะเห็นได้ว่าเมธอดต่างๆด้านบนนี้จะเกี่ยวข้องกับสถานะของปุ่ม นั้น คือ เราอาจจะกำหนดให้ข้อความบนปุ่มแตกต่างกันไปในแต่ละสถานะก็ได้

```
UIButton *normalButton = [UIButton
buttonWithType:UIButtonTypeRoundedRect];
normalButton.frame = CGRectMake(10, 240, 100, 30);
[normalButton setTitle:@"Normal"
                forState:UIControlStateNormal];
[normalButton setTitle:@"Highlighted"
                forState:UIControlStateHighlighted];
[normalButton setTitle:@"Selected"
                forState:UIControlStateSelected];
[self.view addSubview:normalButton];
```


```
UIButton *highlightedButton = [UIButton
buttonWithType:UIButtonTypeRoundedRect];
highlightedButton.frame = CGRectMake(10, 280, 100,
30);
[highlightedButton setTitle:@"Normal"
                forState:UIControlStateNormal];
[highlightedButton setTitle:@"Highlighted"
               forState:UIControlStateHighlighted];
[highlightedButton setTitle:@"Selected"
                forState:UIControlStateSelected];
highlightedButton.highlighted = YES;
[self.view addSubview:highlightedButton];
```


```
UIButton *selectedButton = [UIButton
buttonWithType:UIButtonTypeRoundedRect];
selectedButton.frame = CGRectMake(10, 320, 100, 30);
[selectedButton setTitle:@"Normal"
                forState:UIControlStateNormal];
[selectedButton setTitle:@"Highlighted"
               forState:UIControlStateHighlighted];
[selectedButton setTitle:@"Selected"
                forState:UIControlStateSelected];
selectedButton.selected = YES;
[self.view addSubview:selectedButton];
```

สำหรับการผูกเมธอดกับ event ของ UIButton นั้น โดยมาก เราจะใช้เมธอด

- (void)addTarget:(id)target action:(SEL)action forControlEvents:(UIControlEvents)events

โดยใช้ events เป็น

UIControlEventTouchUpInside

```
UIButton *clickMeButton = [UIButton
buttonWithType:UIButtonTypeRoundedRect];
clickMeButton.frame = CGRectMake(10, 370, 100, 30);
[clickMeButton setTitle:@"Click Me"
                forState:UIControlStateNormal];
[clickMeButton addTarget:self
                  action:@selector(myMethod:)
      forControlEvents:UIControlEventTouchUpInside];
```

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

Project 6.1 - SimpleCalculator

PROJECT SUMMARY

1. สร้างกล่องข้อความขึ้นมาเพื่อแสดงตัวเลข เครื่องหมาย รวมถึงใช้ในการรับ input จากผู้ใช้ผ่านทาง keyboard
2. สร้างปุ่มตัวเลข และเครื่องหมายต่างๆ
3. ถ้าผู้ใช้ keyboard ในการป้อนข้อมูล ทำการตรวจสอบตัวอักษรที่ป้อนเข้ามาว่าเป็นตัวเลขหรือเครื่องหมายที่กำหนดหรือไม่

ในตัวอย่างนี้ เราจะสร้างโปรแกรมเครื่องคิดเลขอย่างง่ายๆกัน โดยเราจะนำคลาสต่างๆที่เราอธิบายในบทนี้มาใช้เป็นส่วนประกอบของเครื่องคิดเลขนี้ โดยใน project นี้ เราจะทำการสร้างไฟล์ขึ้นมา 2 ไฟล์ด้วยกัน คือ

```
CalculatorViewController.h
CalculatorViewController.m
```

โดยในแต่ละไฟล์จะมีโค๊ดดังนี้

```
//
//  CalculatorViewController.h
//

#import <UIKit/UIKit.h>

@interface CalculatorViewController : UIViewController <UIText-
FieldDelegate>
{
    UITextField *numberField;

    NSMutableArray *numberStack;
    NSMutableArray *signStack;
}

- (void)selectButton:(UIButton *)button;
- (void)calculate;
- (void)clear;
```

```
@end
```

ไฟล์ interface ของคลาส CalculatorViewController จะทำการประกาศตัวแปรอยู่ 3 ตัวด้วยกัน คือ

1. numberField เป็นชนิด UITextField ซึ่งเราจะนำมาใช้สำหรับแสดงผลตัวเลขบนเครื่องคิดเลข

2. numberStack เป็นชนิด NSMutableArray ซึ่งจะนำมาใช้สำหรับเก็บตัวเลขที่ผู้ใช้ป้อนเข้ามาเพื่อนำมาใช้ในการคำนวน

3. signStack เป็นชนิด NSMutableArray จะนำมาใช้สำหรับเก็บค่าสัญลักษณ์เพื่อใช้ในการคำนวน

และจะประกาศเมธอดไว้ 3 เมธอดด้วยกัน คือ

1. - (void)selectButton:(UIButton *)button สำหรับรับ input จากผู้ใช้ทางการกดปุ่มตัวเลขบนหน้าจอ

2. - (void)calculate สำหรับทำการคำนวนผลลัพธ์

3. - (void)clear สำหรับทำการ clear ข้อมูลเดิมออกจากเครื่องคิดเลข

และนอกจากนี้เราจะทำการ implement เมธอดสำหรับ UITextFieldDelegate อีกด้วย

สำหรับไฟล์ implementation จะมีโค๊ดดังนี้

```
//
//  CalculatorViewController.m
//

#import "CalculatorViewController.h"

@implementation CalculatorViewController

- (id)init
{
    self = [super init];
    if (self) {
        // Create variable
        numberStack = [[NSMutableArray alloc] init];
        signStack = [[NSMutableArray alloc] init];

        // Set view property
        self.view.autoresizesSubviews = YES;
        self.view.autoresizingMask = UIViewAutoresiz-
ingFlexibleWidth | UIViewAutoresizingFlexibleHeight;
        self.view.backgroundColor = [UIColor white-
Color];

        // Set view component
        numberField = [[UITextField alloc] init];
        numberField.frame = CGRectMake(10, 10, 300,
50);
        numberField.borderStyle = UITextBorderSty-
leRoundedRect;
        numberField.keyboardType = UIKeyboardTypeNum-
bersAndPunctuation;
        numberField.returnKeyType = UIReturnKeyDone;
        numberField.contentVerticalAlignment = UICon-
trolContentVerticalAlignmentCenter;
        numberField.contentHorizontalAlignment = UI-
ControlContentHorizontalAlignmentRight;
        numberField.font = [UIFont
boldSystemFontOfSize:30];
```

```
        numberField.textAlignment = UITextAlignmen-
tRight;
        numberField.text = @"";
        numberField.delegate = self;
        [self.view addSubview:numberField];

        for (int i = 0; i < 9; i++) {
            int row = floor(i / 3);
            int column = i % 3;

            UIButton *numberButton = [UIButton
buttonWithType:UIButtonTypeRoundedRect];
            numberButton.frame = CGRectMake(10 + 70
* column, 210 - 70 * row, 60, 60);
            [numberButton setTitle:[NSString
stringWithFormat:@"%d", i + 1]

forState:UIControlStateNormal];
            [numberButton addTarget:self

action:@selector(selectButton:)

forControlEvents:UIControlEventTouchUpInside];
            [self.view addSubview:numberButton];
        }

        UIButton *zeroButton = [UIButton
buttonWithType:UIButtonTypeRoundedRect];
        zeroButton.frame = CGRectMake(10, 280, 200,
60);
        [zeroButton setTitle:@"0"
                    forState:UIControlStateNormal];
        [zeroButton addTarget:self

action:@selector(selectButton:)

forControlEvents:UIControlEventTouchUpInside];
        [self.view addSubview:zeroButton];
```

```
        UIButton *divideButton = [UIButton
buttonWithType:UIButtonTypeRoundedRect];
        divideButton.frame = CGRectMake(220, 70, 60,
60);
        [divideButton setTitle:@"/"

forState:UIControlStateNormal];
        [divideButton addTarget:self

action:@selector(selectButton:)

forControlEvents:UIControlEventTouchUpInside];
        [self.view addSubview:divideButton];

        UIButton *timeButton = [UIButton
buttonWithType:UIButtonTypeRoundedRect];
        timeButton.frame = CGRectMake(220, 140, 60,
60);
        [timeButton setTitle:@"*"
                    forState:UIControlStateNormal];
        [timeButton addTarget:self

action:@selector(selectButton:)

forControlEvents:UIControlEventTouchUpInside];
        [self.view addSubview:timeButton];

        UIButton *minusButton = [UIButton
buttonWithType:UIButtonTypeRoundedRect];
        minusButton.frame = CGRectMake(220, 210, 60,
60);
        [minusButton setTitle:@"-"
                    forState:UIControlStateNormal];
        [minusButton addTarget:self

action:@selector(selectButton:)

forControlEvents:UIControlEventTouchUpInside];
        [self.view addSubview:minusButton];
```

```
        UIButton *plusButton = [UIButton
buttonWithType:UIButtonTypeRoundedRect];
        plusButton.frame = CGRectMake(220, 280, 60,
60);
        [plusButton setTitle:@"+"
                    forState:UIControlStateNormal];
        [plusButton addTarget:self

action:@selector(selectButton:)

forControlEvents:UIControlEventTouchUpInside];
        [self.view addSubview:plusButton];

        UIButton *calculateButton = [UIButton
buttonWithType:UIButtonTypeRoundedRect];
        calculateButton.frame = CGRectMake(220, 350,
60, 60);
        [calculateButton setTitle:@"="

forState:UIControlStateNormal];
        [calculateButton addTarget:self

action:@selector(calculate)

forControlEvents:UIControlEventTouchUpInside];
        [self.view addSubview:calculateButton];

        UIButton *resetButton = [UIButton
buttonWithType:UIButtonTypeRoundedRect];
        resetButton.frame = CGRectMake(10, 350, 200,
60);
        [resetButton setTitle:@"Clear"
                     forState:UIControlStateNormal];
        [resetButton addTarget:self
                        action:@selector(clear)

forControlEvents:UIControlEventTouchUpInside];
        [self.view addSubview:resetButton];
```

```
    }

    return self;
}

- (void)dealloc
{
    [numberField release];

    [numberStack release];
    [signStack release];

    [super dealloc];
}

#pragma mark - Instance Method

-(void)selectButton:(UIButton *)button
{
    NSString *tempString = [NSString
stringWithFormat:@"%@%@", numberField.text,
button.titleLabel.text];

    numberField.text = tempString;
}

- (void)calculate
{
    [numberStack removeAllObjects];
    [signStack removeAllObjects];

    NSString *inputString = numberField.text;

    NSError *error = nil;
    NSRegularExpression *regex = [NSRegularExpres-
sion
regularExpressionWithPattern:@"^(\\d)+([\\+\\*-/](\\
d)+)*$"
```

```
options:NSRegularExpressionCaseInsensitive

error:&error];
    NSUInteger numberOfMatches = [regex
numberOfMatchesInString:inputString

options:0

range:NSMakeRange(0, [inputString length])];
    if (numberOfMatches == 0) {
        // Invalid input
        UIAlertView *alertView = [[UIAlertView al-
loc] initWithTitle:@"Invalid Input"

message:@"The input expression is invalid, please
type the valid input"

delegate:nil

cancelButtonTitle:@"OK"

otherButtonTitles:nil];
        [alertView show];
        [alertView release];

        return;
    }

    NSMutableString *tempString = [[NSMutableString
alloc] init];

    for (int i = 0; i < [inputString length]; i++) {
        char c = [inputString characterAtIndex:i];
        switch (c) {
            case '+':
            case '-':
            case '*':
            case '/':
```

```
            {
                if ([tempString length] > 0) {
                    [numberStack addObject:[NSNumber
numberWithInt:[tempString intValue]]];
                    [tempString
deleteCharactersInRange:NSMakeRange(0, [tempString
length])];

                    if ([signStack count] > 0 &&
[numberStack count] > 1) {
                        char sign = [[signStack las-
tObject] charValue];
                        int length = [numberStack
count];
                        int x1 = [[numberStack
objectAtIndex:(length - 2)] intValue];
                        int x2 = [[numberStack
objectAtIndex:(length - 1)] intValue];
                        int y;

                        switch (sign) {
                            case '*':
                            {
                                y = x1 * x2;
                                [numberStack remove-
LastObject];
                                [numberStack remove-
LastObject];
                                [signStack removeLas-
tObject];
                                [numberStack
addObject:[NSNumber numberWithInt:y]];

                                break;
                            }
                            case '/':
                            {
                                y = x1 / x2;
```

```
                                [numberStack remove-
LastObject];
                                [numberStack remove-
LastObject];
                                [signStack removeLas-
tObject];
                                [numberStack
addObject:[NSNumber numberWithInt:y]];

                                break;
                            }
                        }
                    }
                }
                [signStack addObject:[NSNumber
numberWithChar:c]];
                break;
            }
            default:
            {
                [tempString appendFormat:@"%c", c];
                break;
            }
        }
    }

    if ([tempString length] > 0) {
        [numberStack addObject:[NSNumber
numberWithInt:[tempString intValue]]];

        if ([signStack count] > 0 && [numberStack
count] > 1) {
            char sign = [[signStack lastObject] char-
Value];
            int length = [numberStack count];
            int x1 = [[numberStack
objectAtIndex:(length - 2)] intValue];
            int x2 = [[numberStack
objectAtIndex:(length - 1)] intValue];
            int y;

            switch (sign) {
                case '*':
                {
                    y = x1 * x2;
                    [numberStack removeLastObject];
                    [numberStack removeLastObject];
                    [signStack removeLastObject];
                    [numberStack addObject:[NSNumber
numberWithInt:y]];

                    break;
                }
                case '/':
                {
                    y = x1 / x2;
                    [numberStack removeLastObject];
                    [numberStack removeLastObject];
                    [signStack removeLastObject];
                    [numberStack addObject:[NSNumber
numberWithInt:y]];

                    break;
                }
            }
        }
    }

    int x;
    int y = [[numberStack objectAtIndex:0] int-
Value];
    for (int i = 0; i < [signStack count]; i++) {
        x = [[numberStack objectAtIndex:(i + 1)] int-
Value];
        switch ([[signStack objectAtIndex:i] char-
Value]) {
            case '+':
            {
```

```
                y += x;
                break;
            }
            case '-':
            {
                y -= x;
                break;
            }
        }
    }

    numberField.text = [NSString
stringWithFormat:@"%d", y];

    [tempString release];
}

- (void)clear
{
    numberField.text = @"";
}

#pragma mark - UITextFieldDelegate

-           (BOOL)textField:(UITextField *)text-
Field
shouldChangeCharactersInRange:(NSRange)range
            replacementString:(NSString *)string
{
    NSCharacterSet *charSet = [NSCharacterSet
characterSetWithCharactersInString:@"0123456789+-*/"
];
    for (int i = 0; i < [string length]; i++) {
        unichar c = [string characterAtIndex:i];
        if (![charSet characterIsMember:c]) {
            return NO;
        }
    }
    return YES;
}

- (BOOL)textFieldShouldReturn:(UITextField *)text-
Field
{
    [textField resignFirstResponder];
    [self calculate];

    return YES;
}

@end
```

ในเมธอด init นั้น เราจะแบ่งการทำงานออกเป็น 3 ส่วนหลักๆ เพื่อให้ง่ายต่อการทำความเข้าใจ ดังนี้

ส่วนแรกเราจะทำการประกาศตัวแปรต่างๆของโปรแกรม โดยในโปรแกรมนี้เราจะทำการสร้างตัวแปร numberStack และ signStack ซึ่งเป็นชนิด NSMutableArray ขึ้นมา

ในส่วนที่สอง เราจะทำการกำหนดคุณสมบัติต่างๆของ object self

```
self.view.autoresizesSubviews = YES;
```

โดยส่วนใหญ่แล้ว เราจะกำหนดคุณสมบัตินี้ให้กับ view ที่จะถูกใช้เก็บ view อื่นๆไว้ และต้องการให้ view อื่นๆภายในทำการปรับขนาดหรือตำแหน่งใหม่อัตโนมัติ

```
self.view.autoresizingMask = UIViewAutoresizingFlexi-
bleWidth | UIViewAutoresizingFlexibleHeight;
```

สำหรับคุณสมบัตินี้ จะขึ้นอยู่กับว่าเราต้องการให้ view นี้ทำการปรับเปลี่ยนอะไรมั่ง

```
self.view.backgroundColor = [UIColor whiteColor];
```

ส่วนนี้จะทำการกำหนดสีของพื้นหลังให้กับ view

ส่วนสุดท้าย เราจะทำการสร้างองค์ประกอบต่างๆสำหรับติดต่อกับผู้ใช้ โดยในโปรแกรมนี้ จะมีองค์ประกอบหลักๆเพียง 2 อย่างด้วยกันคือ UITextField และ UIButton โดยเราจะใช้ UITextField เพื่อแสดงตัวเลขและเครื่องหมายที่ผู้ใช้ป้อนเข้ามา ซึ่งเราจะให้ผู้ใช้สามารถป้อนข้อมูลเข้ามาผ่านทาง UITextField โดยตรง หรือจะใช้ UIButton เพื่อแทนปุ่มตัวเลขและปุ่มเครื่องหมายต่างๆก็ได้

จะเห็นว่าเราจะทำการประกาศตัวแปรเพียงแค่ UITextField เอาไว้ใน interface เท่านั้น แต่จะไม่ประกาศตัวแปรสำหรับ UIButton ต่างๆ ทั้งนี้ก็เนื่องมาจากว่า เราจำเป็นที่จะต้องเข้าไปอ่านและแก้ไขข้อความของ UITextField ในภายหลัง จึงจำเป็นต้องเก็บตัวแปรไว้ให้สามารถเข้าถึงในภายหลังได้ ในขณะที่ UIButton ต่างๆนั้น เราไม่จำเป็นต้องเข้าถึงในภายหลังจึงไม่จำเป็นที่จะต้องประกาศเป็นตัวแปรใน interface ไว้ นอกจากนี้เราจะทำ implement เมธอดใน UITextFieldDelegate ไว้ให้กับ self เพื่อสำหรับใช้ในการตรวจสอบตัวอักษรที่ผู้ใช้ป้อนเข้ามา

```
- (BOOL)textField:(UITextField *)textField
shouldChangeCharactersInRange:(NSRange)range
replacementString:(NSString *)string
```

โดยจะเมธอดนี้ในการตรวจสอบตัวอักษรที่ผู้ใช้ป้อนเข้ามา โดยเมธอดนี้จะถูกเรียกทุกครั้งเมื่อผู้ใช้พิมพ์ตัวอักษรลงใน UITextField โดยตัวอักษรที่พิมพ์เข้ามาจะถูกส่งเข้ามาผ่านทางตัวแปร string และเราจะนำตัวอักษรแต่ละตัวมาทดสอบดูว่าเป็นตัวเลขหรือเครื่องหมายใดเครื่องหมายหนึ่งที่เราจะนำมาใช้งานหรือไม่

โดยขั้นตอนการทดสอบเราจะสร้างตัวแปรของคลาส NSCharacterSet ขึ้นมาเพื่อเก็บตัวอักษรต่างๆที่เราจะนำมาใช้งาน โดยในที่นี้ก็จะเป็นตัวเลข 0-9 และเครื่องหมาย +-*/

```
NSCharacterSet *charSet = [NSCharacterSet
characterSetWithCharactersInString:@"0123456789+-*/"];
```

จากนั้นเราก็จะดึงตัวอักษรออกมาทีละตัว

```
unichar c = [string characterAtIndex:i];
```

และนำตัวอักษรนี้มาทดสอบว่าอยู่ใน charSet ที่เราสร้างขึ้นมาก่อนหน้านี้หรือไม่

```
if (![charSet characterIsMember:c]) {
    return NO;
}
```

โดยตัวอักษรนี้ไม่อยู่ใน charSet ก็จะคืนค่า NO ออกมาเพื่อไม่ให้ UITextField นำตัวอักษรชุดนี้เพิ่มลงไป (หรือก็คือ ไม่อนุญาติให้พิมพ์อักษรตัวนี้ลงไปนั้นเอง) แต่ถ้าทดสอบจนครบทุกตัวแล้ว ตัวอักษรทุกตัวอยู่ใน charSet เราก็จะคืนค่า YES ออกมา

```
- (BOOL)textFieldShouldReturn:(UITextField *)text-
Field
```

สำหรับเมธอดนี้ เราจะกำหนดให้โปรแกรมทำการคำนวนผลลัพธ์ถ้าผู้ใช้กดปุ่ม Done บน keyboard

กลับมาพูดถึงในส่วนของเมธอด init กันต่อ โดยส่วนต่อมาที่ประกาศขึ้นมาก็คือปุ่มตัวเลขและเครื่องหมายต่างๆ โดยปุ่มเหล่านี้เราจะนำมาผูกกับเมธอด

```
-(void)selectButton:(UIButton *)button
```

โดยเมธอดนี้เราจะนำตัวเลข หรือเครื่องหมายของปุ่มที่ถูกกดมาเพิ่มลงใน UITextField

ถัดมาก็จะเป็นปุ่มเครื่องหมาย “=” ซึ่งจะนำมาผูกกับเมธอด

```
- (void)calculate
```

ซึ่งเมธอดนี้เราจะใช้ในการกำหนดข้อมูลที่พิมพ์เข้ามาใน UITextField โดยเราจะขอข้ามคำอธิบายวิธีการคำนวนในส่วนนี้ไป เนื่องจากวิธีการคำนวนในส่วนนี้จะสามารถทำได้หลายวิธีด้วยกัน ในตัวอย่างนี้ก็เป็นตัวอย่างวิธีการคำนวนวิธีหนึ่ง

ปุ่มสุดท้ายที่เราสร้างขึ้นก็คือปุ่ม Clear ซึ่งเราจะนำมาผูกกับเมธอด

```
- (void)clear
```

โดยเมธอดนี้จะทำการลบข้อมูลการคำนวนต่างๆที่ทำไว้ก่อนหน้านี้ และลบข้อมูลที่อยู่ใน UITextField ออกด้วย

โดยเมื่อทำการรันโปรแกรมนี้แล้ว ก็จะออกมาดังนี้

สรุปสิ่งที่กล่าวถึงในบทนี้

1. คลาส **UILabel**
2. คลาส **UIControl**
3. คลาส **UITextField**
4. คลาส **UITextView**
5. คลาส **UIButton**
6. ตัวอย่างโปรแกรม **SimpleCalculator**

ในบทนี้เราก็ได้พูดถึงคลาสพื้นฐาน 5 คลาสด้วยกัน ซึ่งก็คือ UILabel, UIControl, UITextField, UITextView และ UIButton ที่คลาสเหล่านี้จะถูกใช้เป็นส่วนประกอบหลักๆ สำหรับใช้ในการติดต่อกับผู้ใช้ในโปรแกรม

คลาส UILabel นั้นโดยมากแล้วเราจะนำมาใช้สำหรับแสดงข้อความเพียงอย่างเดียวเท่านั้น ซึ่งอาจจะใช้ข้อมูลต่างๆ หรืออาจจะนำมาใช้แสดงสถานะการทำงานของโปรแกรม อย่างเช่น Loading..., Completed เป็นต้น

คลาส UIControl นั้นจะไม่ถูกนำมาใช้โดยตรง แต่จะเป็นคลาสพื้นฐานของคลาสอื่นๆที่จะนำมาใช้ในการรับ input จากผู้ใช้

คลาส UITextField จะเป็นกล่องข้อความแบบบรรทัดเดียว ถูกนำมาใช้รับข้อความจากผู้ใช้ผ่านทาง keyboard ซึ่งเดียวกับคลาส UITextView ซึ่งจะเป็นกล่องข้อความเช่นเดียวกัน แต่จะเป็นแบบกล่องข้อความหลายบรรทัด

คลาสสุดท้ายที่พูดถึงในบทนี้ก็คือคลาส UIButton ซึ่งทำหน้าที่เป็นปุ่มเพื่อให้ผู้ใช้เลือกการทำงานของโปรแกรมที่ต้องการ

ซึ่งจริงๆแล้ว คลาสพื้นฐานสำหรับใช้ติดต่อกับผู้ใช้งานนั้นยังมีอยู่อีกมากมายหลายคลาสด้วยกัน โดยในบทต่อๆไป เราจะมาพูดถึงคลาสอื่นๆที่ทำงานซับซ้อนมากขึ้น ซึ่งโดยมากมักจะถูกออกแบบมาเพื่อให้ทำงานบางอย่างโดยเฉพาะ