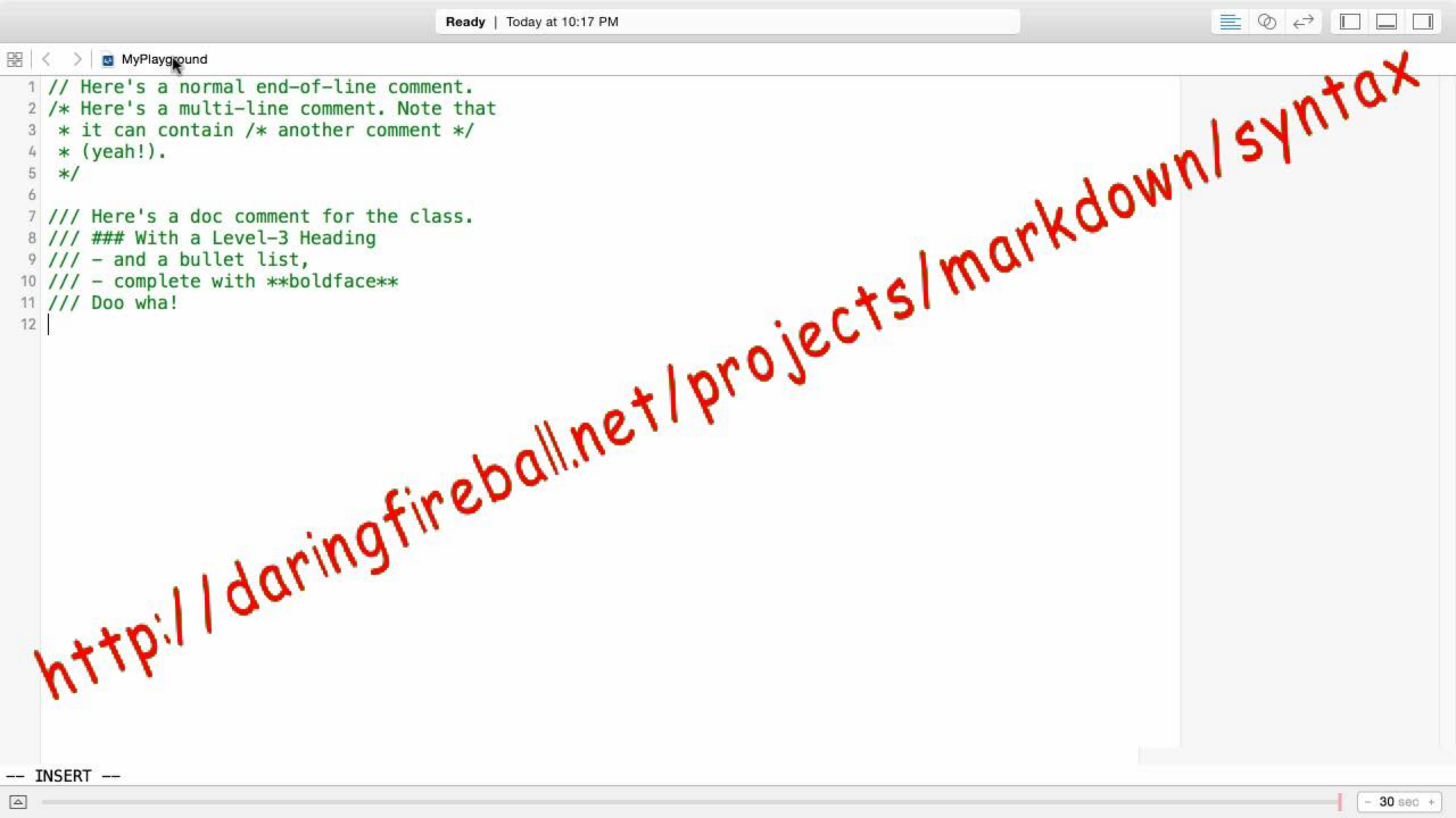
Operators, Types, and Collections

Allen Holub

http://holub.com | Allen Holub | @allenholub


Ready | Today at 10:17 PM
MyPlayground
1 // Here's a normal end-of-line comment.
2 /* Here's a multi-line comment. Note that
3 * it can contain /* another comment */
4 * (yeah!).
5 */
6
7 /// Here's a doc comment for the class.
8 /// ### With a Level-3 Heading
9 /// - and a bullet list,
10 /// - complete with **boldface**
11 /// Doo wha!
12
http://daringfireball.net/projects/markdown/syntax
-- INSERT --
30 sec

MyPlayground

```swift
// Here's a normal end-of-line comment.
/* Here's a multi-line comment. Note that
 * it can contain /* another comment */
 * (yeah!).
 */

/// Here's a doc comment for the class.
/// ### With a Level-3 Heading
/// - and a bullet list,
/// - complete with **boldface**
/// Doo wha!

class MyClass {
    /// Here's a function-level doc comment
    /// - Returns: something interesting
    /// - Parameters:
    ///   - x: does something
    ///   - y: does something else
    /// - parameter z: does even more

    func f( x:Int, y:Int, z:Int ) -> String {
        return "hello"
    }
}

let c = MyClass()
```

"hello"

MyClass
"hello"

Description Here's a doc comment for the class.

With a Level-3 Heading

- and a bullet list,
- complete with **boldface** Doo wha!

Declared In MyPlayground.playground

30 sec

```
import Foundation
@testable import MyFramework



print("hello")
```

```
import Foundation
@testable import MyFramework



print("hello"); print(" !")
```

```
import Foundation
@testable import MyFramework


let s = "world"

print("hello \(s)", appendNewline:false )
```

```
if #available(iOS 8.0, OSX 10.10, *) {
    //...
}
```

```
@available(iOS 8.0, OSX 10.10, *)
func worksOnlyWithNewerOS(){ /*...*/ }
```

$$y = -x$$

$$y = -x$$

doesn't evaluate to anything

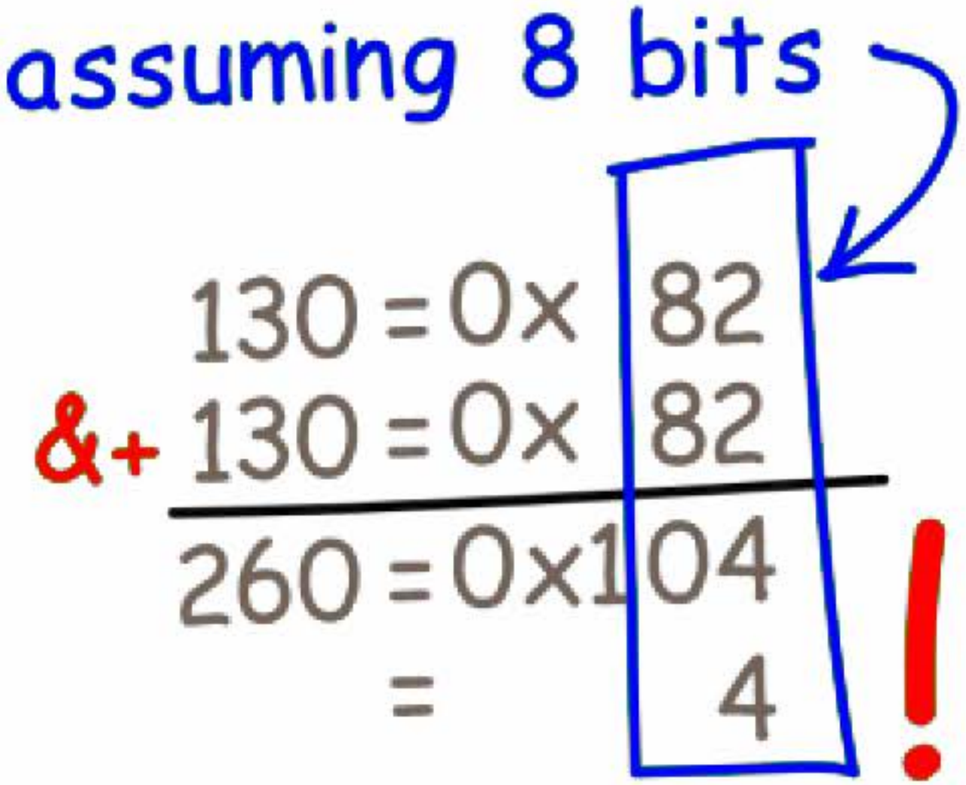
```
if a-b == 0 { /*...*/ }
```

```
a = b = c
```

```
123.5 % 123
```

```
a &+ b
a &* b
```

```
a &- b
a &/ b
```



```
123.5 % 123

a &+ b        a &- b

a &* b        a &/ b

1...3
1..<3

let i = 5
if 1...10 ~= i { /*...*/ }
```

Swift Precedence Chart

```
    .
    ++ -- ! ~ + - &

160 << >>
150 * / % &     &*
140 + - | ^     &+ &-
135 ..<   ...
132 is as as? as!
131 ??
130 < <= > >= == != ~= === !==
120 &&
110 ||
100 ?:
90  = *= /= %= += -= <<= >>= ||= &&= ^= |= &=
```

```swift
var someVariable = 10

var anotherVariable:Int
anotherVariable = 10


let someConstant = "123"

let anotherConstant: String
anotherConstant = "hello"
```

```
UInt
Int
Double
Float
Bool
String
```

```
Int8   Int16   Int32   Int64
UInt8  UInt16  UInt32  UInt64
UTF8   UTF16   UTF32   UTF64
Float80
```

```
Int.min   Int.max
```

```
typealias Unsigned = UInt16
```

```
let anInt    = 10
let aDouble = 1.0

let x = aDouble + Double(anInt)
```

```
import Foundation

var swiftString: String   = "abc"
var objcString : NSString = "def"

objcString  = swiftString
swiftString = objcString as String
```

```
var bird:String = "chicken"
```

```
var bird = "chicken"
```

```
var bird = "chicken"

bird += "🐔"
bird = "road" + " runner"
```

```
bird.isEmpty
```

```
bird.characters.count()
```

```
bird.hasPrefix("road")
```

```
bird.hasSuffix("runner")
```

```
s1 != s2
```

s1 < s2

е == é

```
for eachCharacter in s {
    /*...*/
}
```

```
bird.utf8
bird.utf16
```

```
for scalar in s.unicodeScalers{
    f(scalar.value)
}
```

```
var s = "0123456789"
```

```
s[s.startIndex]                          → "0"
s[s.startIndex.successor()]              → "1"
s[s.endIndex.predecessor()]              → "9"
s[advance(s.startIndex, 3)]              → "3"
s[advance(s.endIndex,  -2)]              → "8"
```

```
var s = "0123|456789"

s.insert("|",
    atIndex: advance(s.startIndex,4))
```

```
var s = "0123|"

s.insert("|",
    atIndex: advance(s.startIndex,4))

let range =
advance(s.endIndex,-6) ..< s.endIndex

s.removeRange( range )
```

```swift
let someString = "hello"
var possibleNum: Int?

possibleNum = Int(someString)
let x = possibleNum!

if possibleNum != nil {
    let theNum = possibleNum!
}
```

```swift
let someString = "hello"
var possibleNum: Int!

possibleNum = Int(someString)
let x = possibleNum

if possibleNum != nil {
    let theNum = possibleNum!
}
```

```
if let a = optionalValue
{
    // safe to use theNum here
}
```

```
func cow() -> Int? {/*…*/}
func bar() -> Int? {/*…*/}

if let a = optionalValue,
       b = bar() where a < b,
   let c = cow()

{
    // safe to use theNum here
}
```

```
if someValue > 42 && otherValue < 19,
    let a = getOptionalThing()
                where a > someValue {

}
```

```
var someArray = Array<String>()
```

```
var someArray = [String]()
```

```
var someArray : [String] =[ ]
```

```
var someArray : [String] =["a","b","c" ]
```

```
var someArray : [String] =["a","b","c" ]


var someOtherArray =
    [String](count:3, repeatedValue:"")
```

```
var someArray : [String] =
        [ "a","b","c","d","e","g" ]


someArray += ["e","g"]
```

```
var someArray : [String] =
        [ "a","b","c","d","e","f","g" ]


someArray.insert("f", atIndex:5)
```

```
var someArray : [String] =
        [ "a","b","c","d","e","f"]


someArray.removeAtIndex(
                        someArray.count-1)
```

```
var someArray : [String] =
        [ "a","b","c","d","e"]


someArray.removeLast()
```

```
var someArray : [String] =
        [ "a","B","C","d","e"]


someArray [1...2] = ["B", "C"]
```

```
var someArray : [String] =
        [ "a","x","y","z" ,"d","e"]


someArray [1...2] = ["x","y","z"]
```

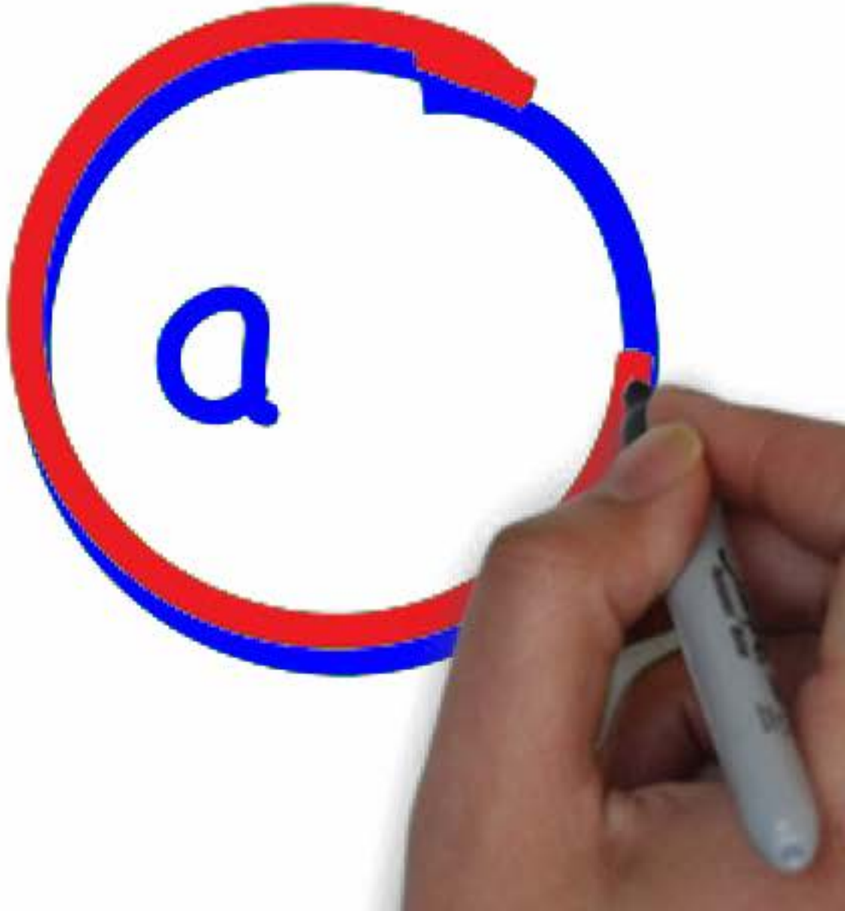
```
var twoD: [[String]]=[ ["a","b","c"],
                       ["d","e","f"] ]

twoD[0] = ["A", "B", "C"]
twoD[0][0] = "x"
```

```
var mySet: Set<String> = [ ]
```

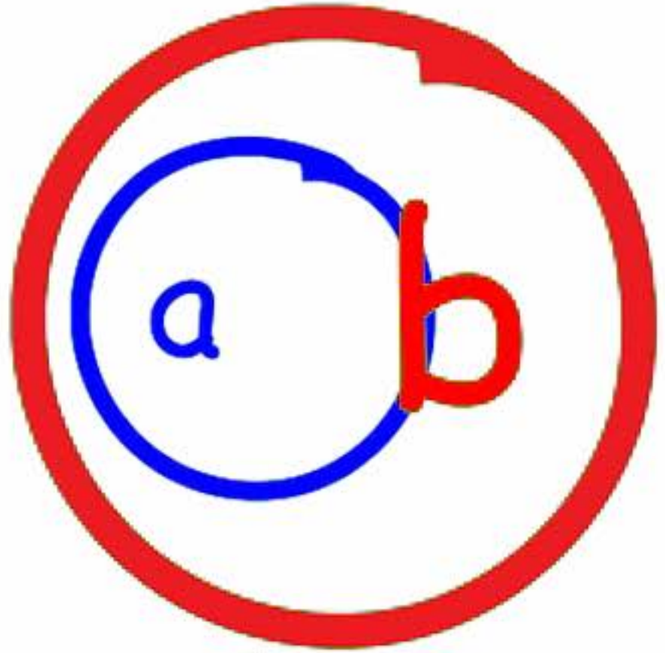
```
var mySet: Set<String> = ["A","B","C" ]
```

```
var c = Set<String>( ["A", "B", "C"] )
```

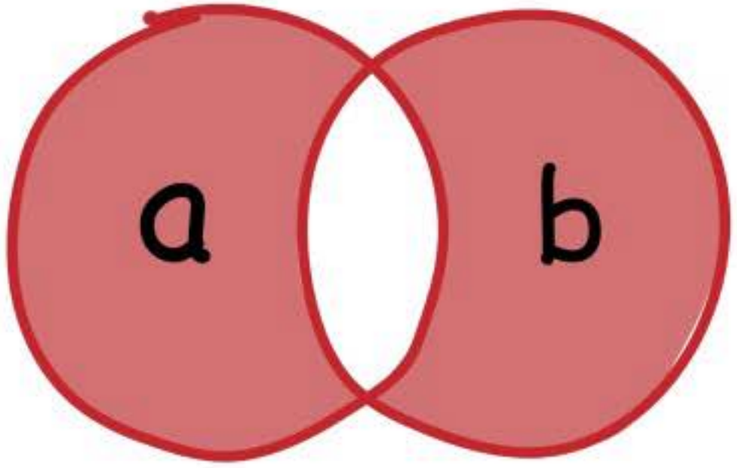
```
a.insert("A")
if let oldValue = a.remove("A") {/*…*/}
a.removeAll()
a.isEmpty
a.count
a.contains("A")
a.isDisjointWith(b)
a.isSupersetOf   (b)
a.isSubsetOf     (b)
```



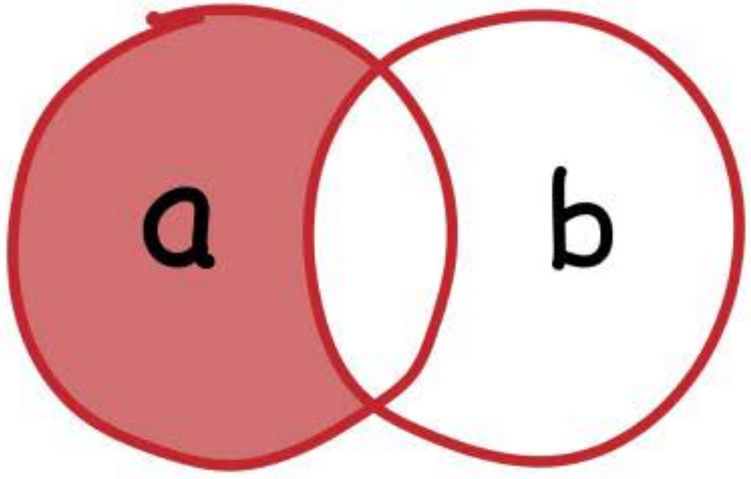
```
a.insert("A")
if let oldValue = a.remove("A") {/*…*/}
a.removeAll()
a.isEmpty
a.count
a.contains("A")
a.isDisjointWith(b)
a.isStrictSupersetOf  (b)
a.isStrictSubsetOf    (b)
```

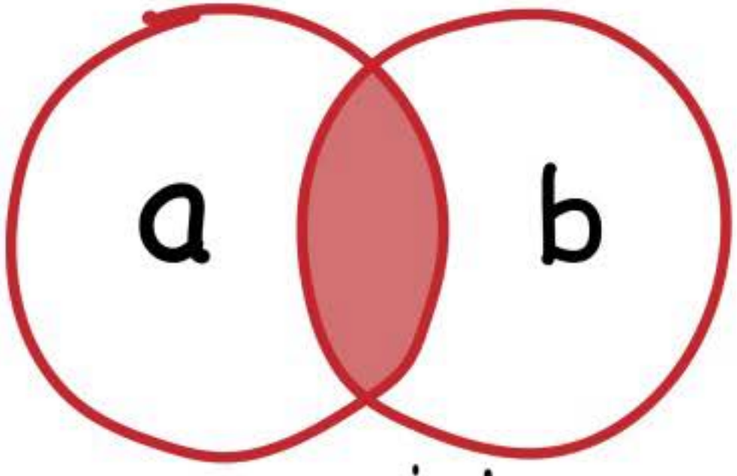


a.exclusiveOr(b)

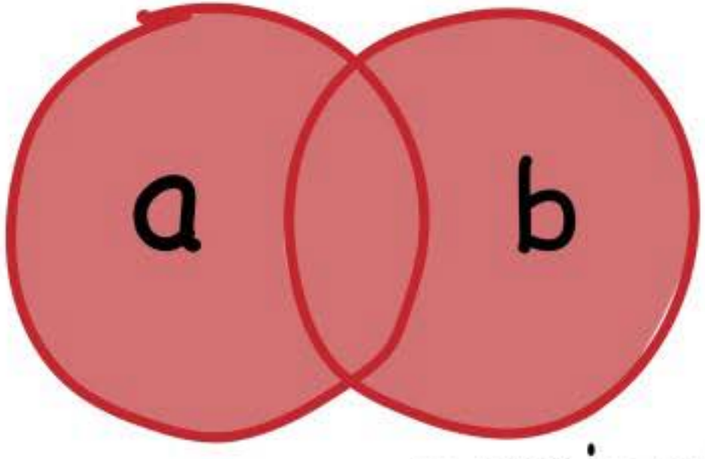


a.subtract(b)





a.intersect(b)



a.union(b)

```
let s1 = a.exclusiveOrInPlace(b)
let s2 = a.intersectInPlace  (b)
let s3 = a.subtractInPlace   (b)
let s4 = a.unionInPlace      (b)
```

```
var httpStatus: [Int:String] = [:]


httpStatus[200] = "OK"
httpStatus[404] = "Not Found"
```

```
var httpStatus =
        [200:"OK", 404:"Not Found"]

httpStatus[404] = "Page Not Found"
```

```swift
var httpStatus =
        [200:"OK", 404:"Not Found"]

httpStatus[404] = "Page Not Found"

if let old = httpStatus.updateValue(
        "PageNotFound", forKey:404) {
    /*…*/
}
```

```swift
var httpStatus =
        [200:"OK", 404:"Not Found"]

for (code, message) in httpStatus {
    print("\(code):  \(message)")
}

let allKeys   = httpStatus.keys
let allValues = httpStatus.values
```

```
var httpStatus =
        [200:"OK", 404:"Not Found"]

httpStatus[404] = nil

if let old =
    httpStatus.removeValueForKey(404){
}

httpStatus.isEmpty
```