Classes and Structures

Common things

프로퍼티 (Properties)

메서드 (Methods)

서브스크립트 (Subscript)

초기화 (Initialization)

확장 (Extension)

프로토콜 (Protocol)

Only Classes

상속 (Inheritance)

형변환 (Type Casting)

소멸 (Deinitialize)

참조 (Reference Counting)

```
class SomeClass {

}
```

```
struct SomeStructure {

}
```

```
struct Resolution {
    var width = 0
    var height = 0
}
class VideoMode {
    var resolution = Resolution()
    var interlaced = false
    var frameRate = 0.0
    var name: String?   // nil
}
```

```
let someResolution = Resolution()
let someVideoMode = VideoMode()
```

```
let someResolution
= Resolution(width:100, height:200)
let someVideoMode = VideoMode()
```

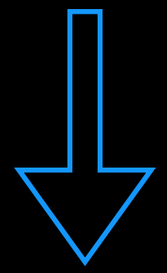
```
let someResolution
= Resolution(width:100, height:200)
let someVideoMode = VideoMode()

someResolution.width   // 100
someResolution.height  // 200

someVideoMode.resolution.width   // 0
someVideoMode.resolution.height = 1280
```

```
struct Resolution {
    var width = 0
    var height = 0
    var density:Int?
}
```

```
struct Resolution {
    var width = 0
    var height = 0
    var density:Int?
}
```



```
let someResolution
= Resolution(width:100, height:200, density:96)
```

```
let hd = Resolution(width: 1920, height: 1080)
var cinema = hd

cinema.width = 2048

cinema.width // 2048
hd.width     // 1080
```

```
enum CompassPoint {
    case North, South, East, West
}
var currentDirection = CompassPoint.West

let rememberedDirection = currentDirection
currentDirection = .East
if rememberedDirection == .West {
    print("The remembered direction is still
          .West")
}
```

```
let tenEighty = VideoMode()
tenEighty.resolution = hd
tenEighty.interlaced = true
tenEighty.name = "1080i"
tenEighty.frameRate = 25.0

let alsoTenEighty = tenEighty
alsoTenEighty.frameRate = 30.0

tenEighty.frameRate // 30.0
```

```
if tenEighty === alsoTenEighty {
    ...
}

if tenEighty !== alsoTenEighty {
    ...
}
```

Identity Operators

```
if tenEighty === alsoTenEighty {
   ...
}

if tenEighty !== alsoTenEighty {
   ...
}
```

```
func ===(lhs: AnyObject?, rhs: AnyObject?) -> Bool
func !==(lhs: AnyObject?, rhs: AnyObject?) -> Bool
```

Choosing Between Classes and Structures

간단한 데이터들의 캡슐화

캡슐화 된 값들의 인스턴스가 참조보단 복사를 필요로 할 때

구조체에 저장되는 프로퍼티 들이 참조보단 복사를 해야하는 값형식 인 경우

구조체가 다른 타입에서 프로토콜이나 상속이 필요없는 경우

Example of good candidates

Double 타입의 width와 height 프로퍼티를 갖는 기하학 모형

Int 타입의 start와 length 프로퍼티를 갖는
범위를 나타내기 위한 방법들

Double 타입의 x, y, z 프로퍼티를 갖는 3D 좌표계 시스템

이외 모든 경우는 클래스를 사용.
사용자 데이터 형은 클래스로 사용하면 된다.

Assignment and Copy Behavior for Collection Types

```
struct Array<T>
struct Dictionary<KeyType : Hashable, ValueType>
```

```
class NSArray
class NSDictionary
class NSMutableArray : NSArray
class NSMutableDictionary : NSDictionary
```