Swift: Struct VS Class

Compare	Struct	Class
Giống	- Đều có Stored Properties + Methods	
	-Hỗ trợ các Subscript Syntax để lấy giá trị theo các Subscript ex: object[index] = ...	
	- Allow customize init()	ex: 1 struct/class có thể có nhiều init với các parameter khác nhau tuỳ mục đích sử dụng
		init(String,Int)
		init(String,Int,Int)
		init() ,....
	- Extension, Protocol allow for both	
Khác		
1)	**- Value type:** mỗi instance giữa 1 bản sao duy nhất data của nó	**- Reference Type:** Mỗi instance chia sẻ một bản sao dữ liệu
	=> Giá trị 1 instance thay đổi không ảnh hưởng giá trị của instance khác	=> Giá trị 1 instance thay đổi thì giá trị của instance khác cũng thay đổi theo
	EX:	EX:
	struct Employee {...}	class Employee {...}

	var a = Employee(code: 5, name: "BXH", address: "Sun*")	var a = Employee(code: 5, name: "BXH", address: "Sun*")
	var b = a	var b = a
	b.name = "MH"	b.name = "MH"
	print(a.name) // **In ra "BXH"**	print(a.name) // **In ra "MH"**
	print(b.name) // In ra "MH"	print(b.name) // In ra "MH"
	=> Khi muốn thay đổi giá trị trực tiếp của thuộc tính :	
2)	Struct cần khai báo **mutating** trước phương thức	Class thay đổi trực tiếp mà không cần khai báo **mutating**
	--> Variable structs can have variable properties changed	--> Variable classes can have variable properties changed
	--> Constant structs **CANNOT** have variable properties changed	--> Constant classes can have variable properties changed
3)	- Struct không có	- Class có hàm **denit()** đảm bảo instance đã bị huỷ

4)	- Struct không có tính kế thừa	- Có tính kế thừa (inheritance). Cho phép có subclass. Subclass kế thừa thuộc tính và phương thức của parent class
5)	- Có memberwise initializer	- Class không có. Do đó khi tạo một class, ta sẽ phải khai báo Optional cho các thuộc tính hoặc phải tự định nghĩa một hàm khởi tạo.
6)	- Struct không hỗ trợ Type Casting	- Class có Type Casting '**is**' và '**as**' để kiểm tra hoặc ép kiểu
7)	-Struct không có	- Do là reference type, class cho phép **===** và **!==** operators để kiểm tra các đối tượng có đang trỏ tới cùng một instance hay không.
Sử dụng khi:		

	- tạo đối tượng với nhiều thuộc tính có kiểu dữ liệu đơn giản (Int, Float, String, ...)	- Class được sử dụng trong trường hợp chúng ta cần những tính chất đặc biệt của nó mà Struct không có như inheritance
	- làm việc đa luồng.	- làm việc với cả trình biên dịch của Swift và Objective - C,
	EX: kết nối database được thực hiện trên một luồng song với luồng Main => an toàn hơn do có thể copy giá trị từ luồng này sang luồng khác.	
		- Việc copy các instance là không hợp lý hoặc không cần thiết.
		EX: với những đối tượng như DatabaseConnection. Việc copy tạo ra nhiều instance cùng trỏ tới 1 file là không cần thiết, bởi chúng luôn kết nối tới cùng một dữ liệu như nhau.

Ưu điểm:	không có phần nào trong code có được tham chiếu tới đối tượng của chúng ta trừ khi ta truy xuất thẳng tới chúng.	- lợi dụng tính kế thừa của Class trong trường hợp này sẽ giúp code chúng ta ngắn gọn và clean
	=> dễ quản lý việc kiểm soát giá trị đối tượng khi bị thay đổi	