**1. What is extension?**

Swift Extension is a useful feature that helps in adding more functionality to an existing [Class](https://www.journaldev.com/15980/swift-class), [Structure](https://www.journaldev.com/15881/swift-struct), [Enumeration](https://www.journaldev.com/15343/swift-enum) or a Protocol type. This includes adding functionalities for types where you don’t have the original source code too (extensions for Int, Bool etc. types).

**Example :**

extension Int {

var square : Int{

return self\*self

}

func cube()->Int{

return self\*self\*self

}

mutating func incrementBy5() {

self = self + 5

}

}

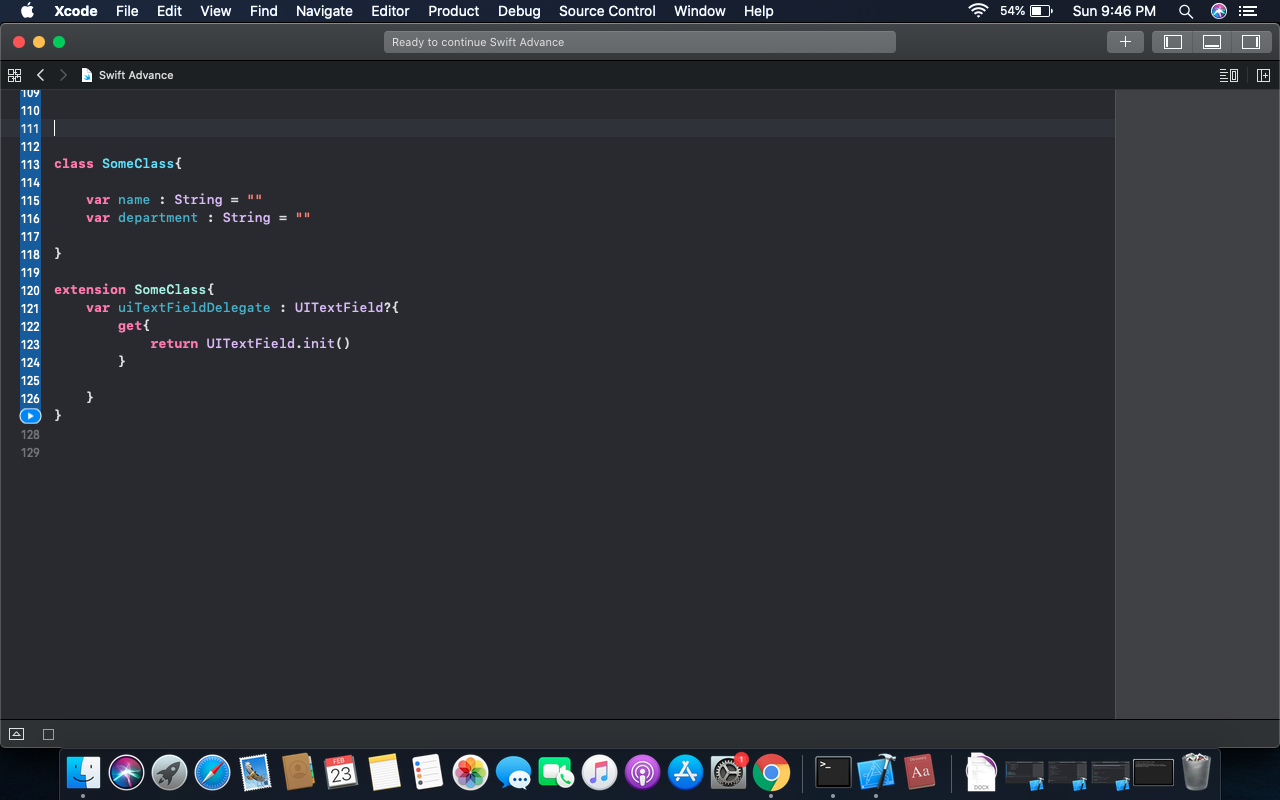
var x : Int = 5

print(x.square) //prints "25\n"

print(x.cube()) //prints "125\n"

x.incrementBy5() // 10

**2. Create a class and write the delegate of UITextField in extension of that class.**

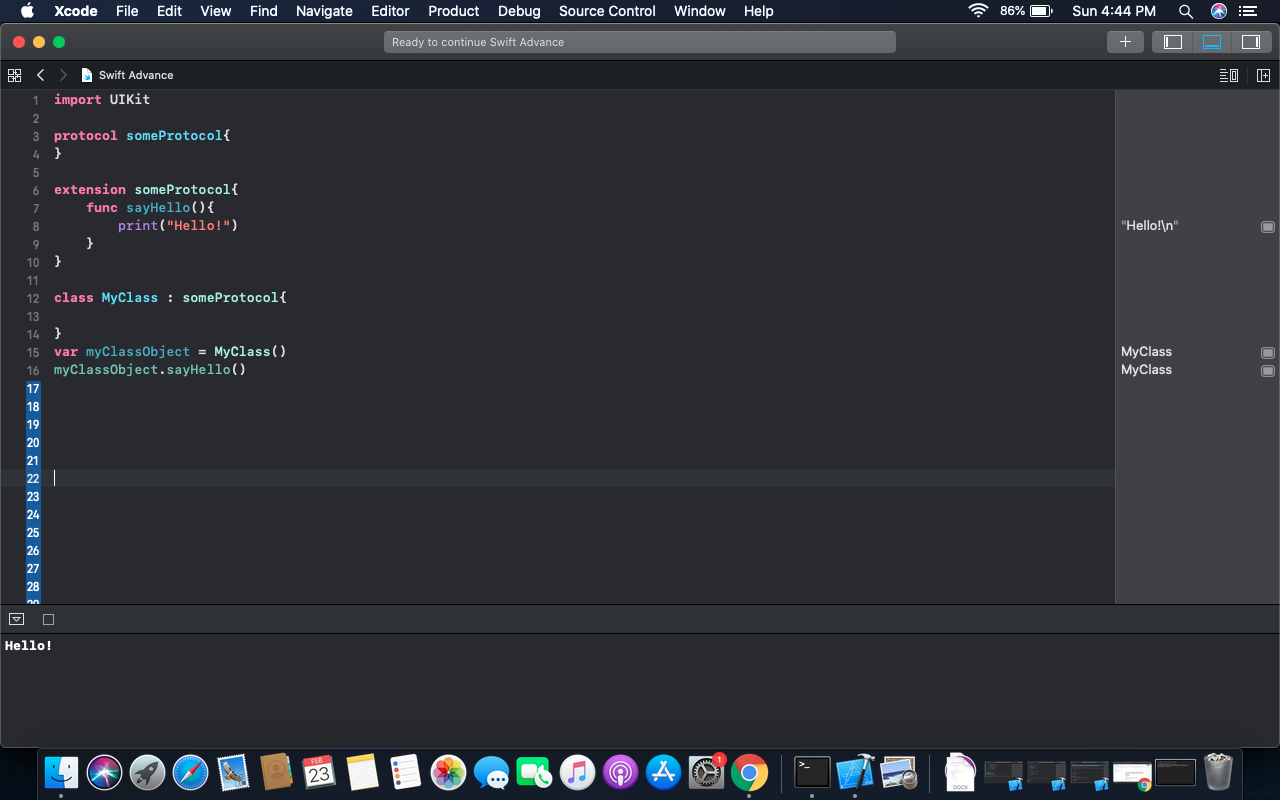
****

**3. Write a protocol and create an extension of the protocol. In extension create a function**

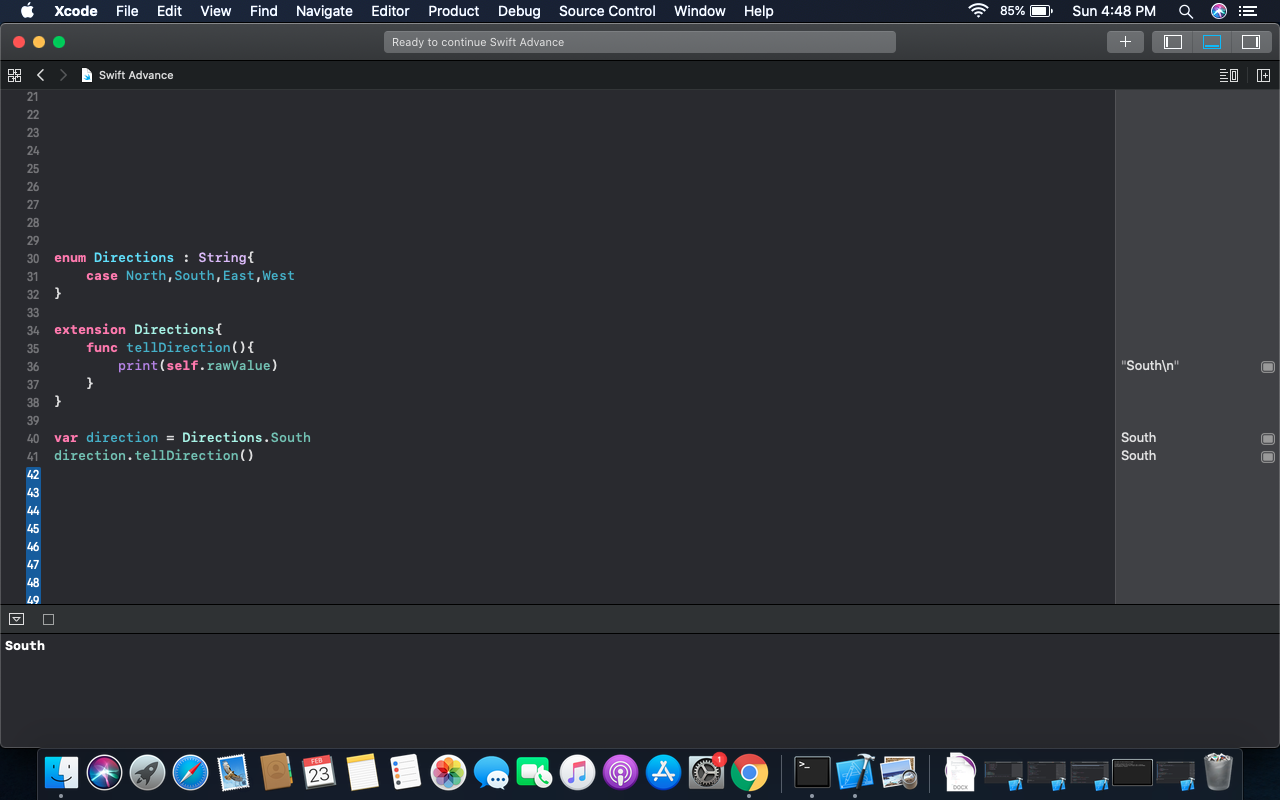
**func sayHello() {**

**print(“Hello!”)**

**}**

****

**4.Write an enum and create an extension of the enum.**

****

**5. What is Generic?**

Generic code enables you to write flexible, reusable functions and types that can work with any type, subject to requirements that you define. You can write code that avoids duplication and expresses its intent in a clear, abstracted manner.

**6. Explain generic with an example?**

**func swapTwoInts(\_ a: inout Int, \_ b: inout Int) {**

**let temporaryA = a**

**a = b**

**b = temporaryA**

**}**

**func swapTwoStrings(\_ a: inout String, \_ b: inout String) {**

**let temporaryA = a**

**a = b**

**b = temporaryA**

**}**

**func swapTwoValues<T>(\_ a: inout T, \_ b: inout T) {**

**let temporaryA = a**

**a = b**

**b = temporaryA**

**}**

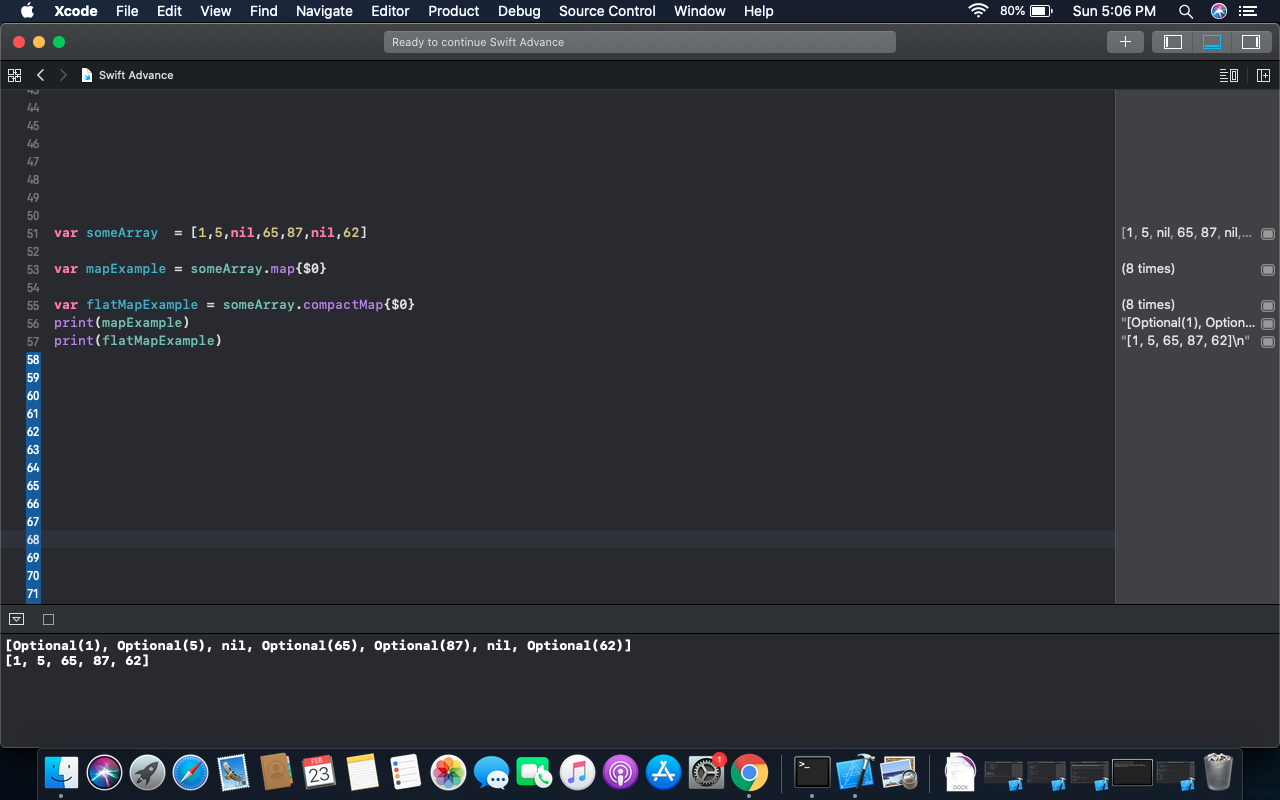
**7. Explain the difference between map and compactMap with an example.**

map and compactMap both are Higher Order Functions(A function that operates on other functions either taking them as argument or returning them) in swift.

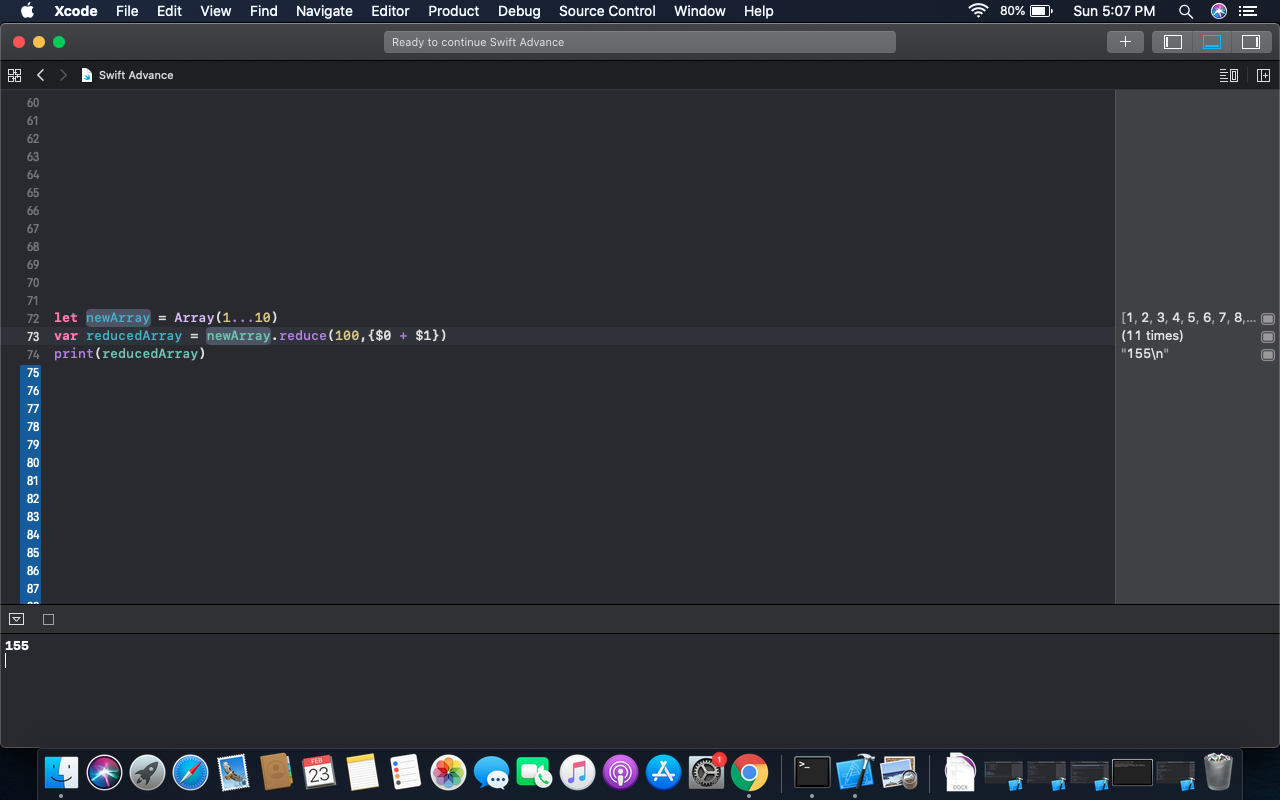
**map Function/Method : -**

The **map**() method allows us to transform arrays (and indeed any kind of collection) using a transformation closure we specify.

The compactMap() method works same as map but with additional functionality of removing nil from the collection



**8. Write an example of reduce function with initial value 1000.**

****

**9. - 2 marks**

**struct Person {**

**var name : String**

**var age : Int**

**}**

**let person1 = Person(name: "Sam", age: 23)**

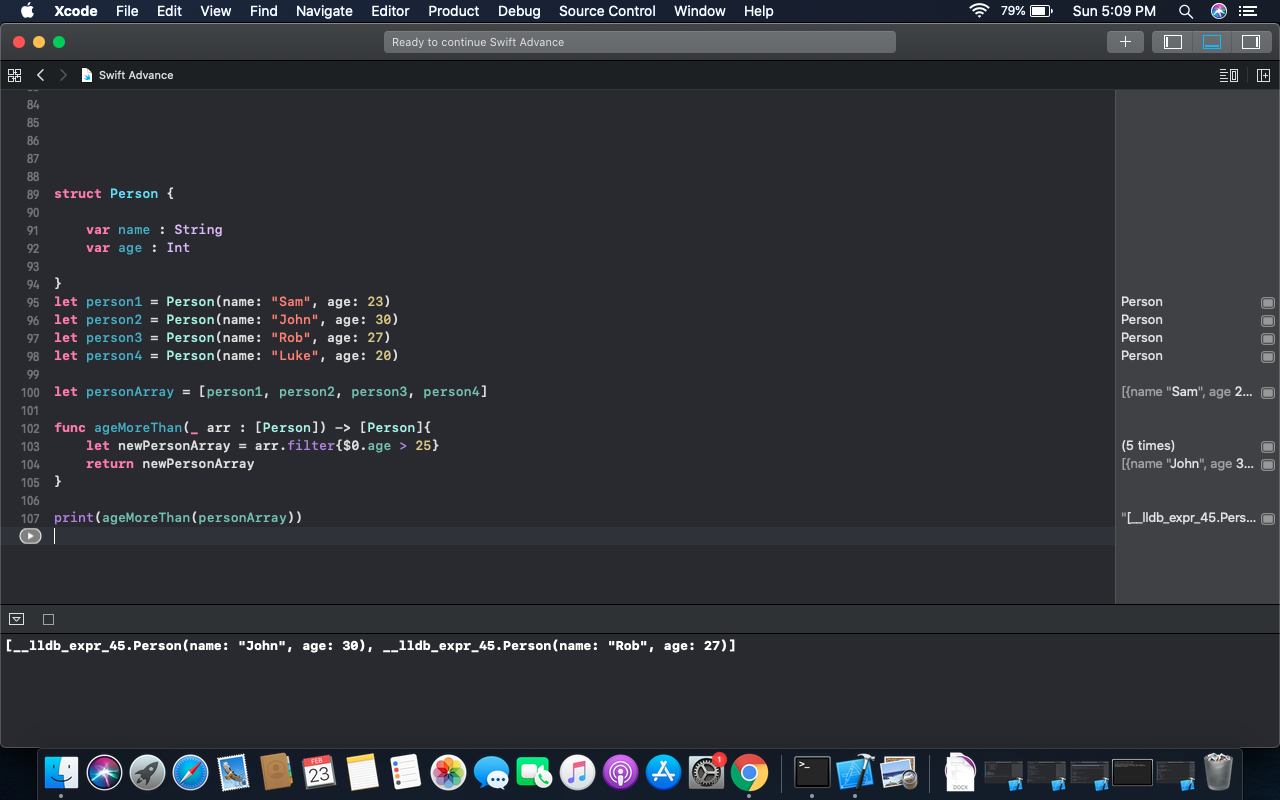
**let person2 = Person(name: "John", age: 30)**

**let person3 = Person(name: "Rob", age: 27)**

**let person4 = Person(name: "Luke", age: 20)**

**let personArray = [person1, person2, person3, person4]**

**Find all person whose age is more than 25 using filter function.**

****

**10. Make a property wrapper @nonNegative and use it to make values to 0 if any negative value added to a variable.**