# iOS : Swift Advance

Name : Shreyansh Raj Keshri
Date : 19/02/2020

1. What is extension?

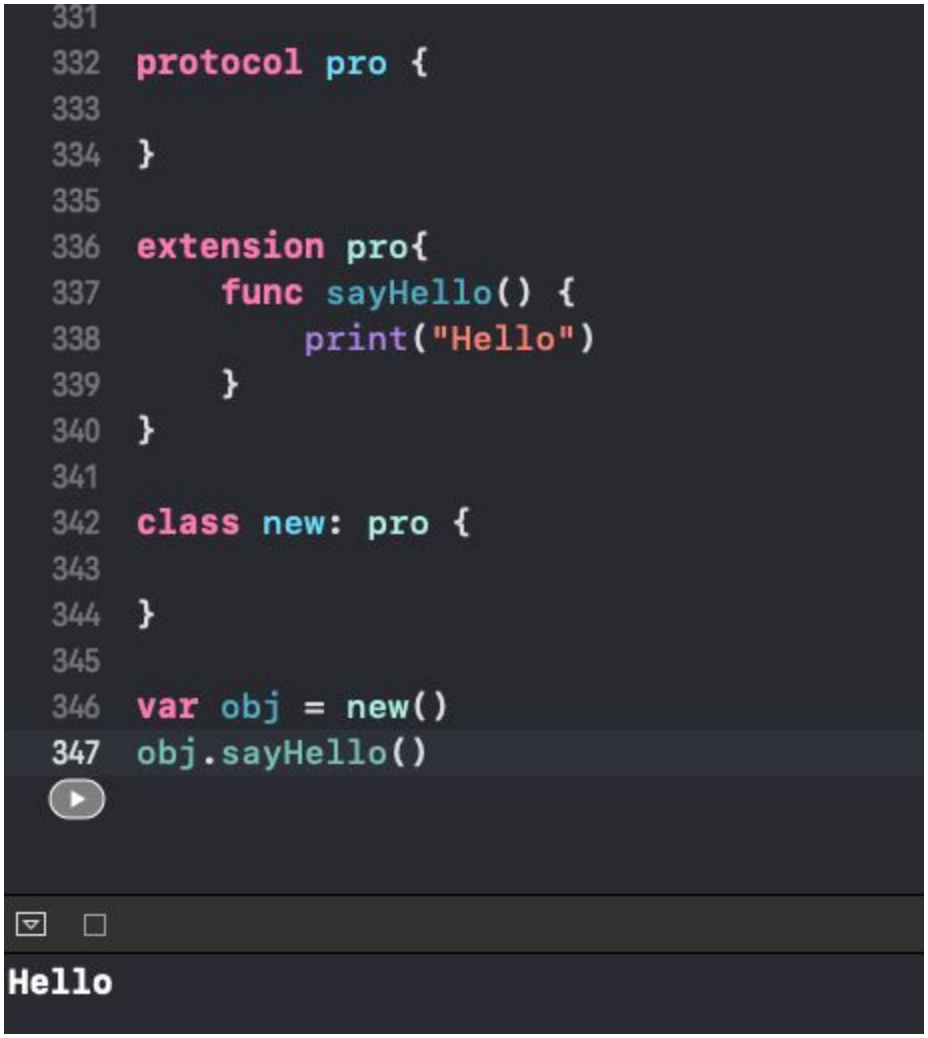
Extensions add new functionality to an existing class, structure, enumeration, or protocol type.

```
extension sometype {
//some Code
}
```

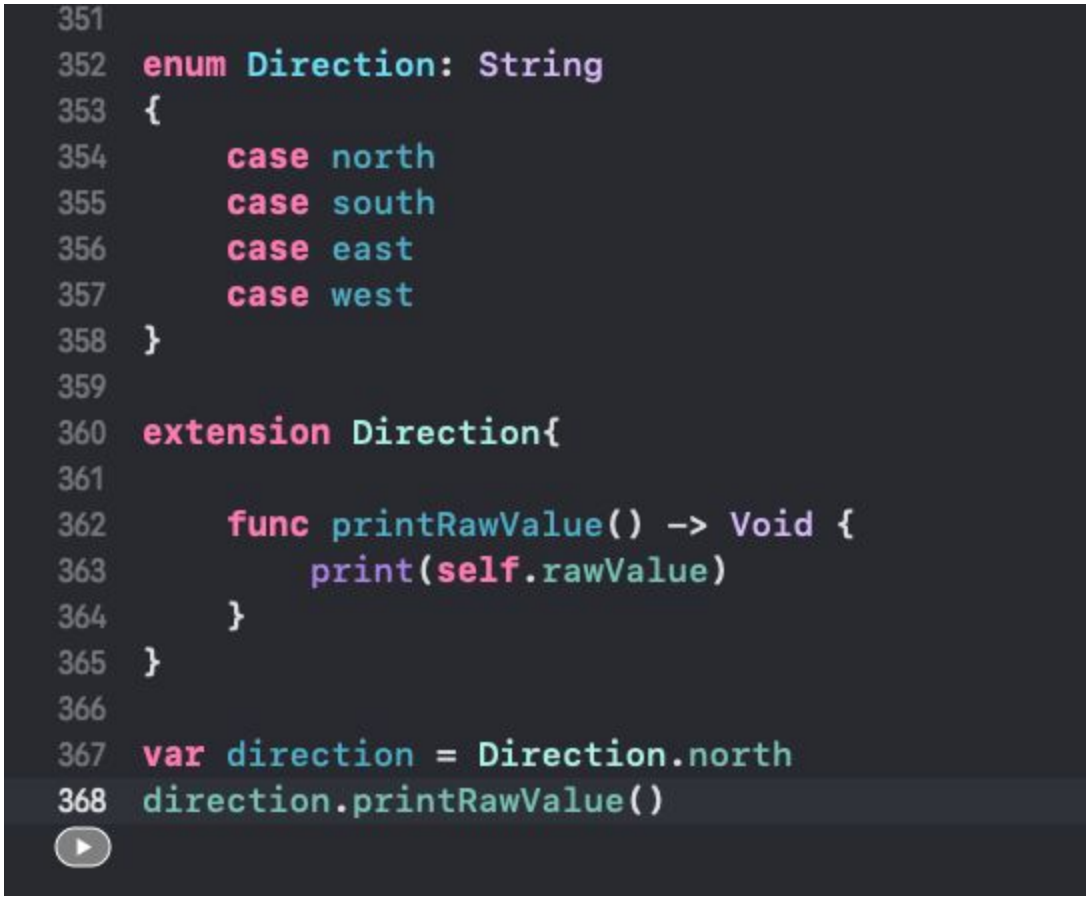
2. Create a class and write the delegate of UITextField in extension of that class.

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.



```
351
352 enum Direction: String
353 {
354     case north
355     case south
356     case east
357     case west
358 }
359
360 extension Direction{
361
362     func printRawValue() -> Void {
363         print(self.rawValue)
364     }
365 }
366
367 var direction = Direction.north
368 direction.printRawValue()
```

5. What is Generic?

Generic allows us to write flexible and re-usable code that can be used with any type of defined by the user.

6. Explain generic with an example?

```swift
func swapToInt(_ value1: inout Int, _ value2: inout Int)
{
    let temp = value1
    value1 = value2
    value2 = temp
}

func swapToString(_ value1: inout String, _ value2: inout String)
{
    let temp = value1
    value1 = value2
    value2 = temp
}

func swapToValue<T>(_ value1: inout T, _ value2: inout T)
{
    let temp = value1
    value1 = value2
    value2 = temp

}
```

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

map() : it is able to return a different type from the one that was originally used. So, this will convert our integer array to a string array

CompactMap(): it is working with optionals can be annoying, but compactMap() can make life much easier. It perform a transformation, but then unwraps all the optional and discards any that are nil

```swift
var arrTemp = [1,2,3,nil,5]

print(arrTemp.map{$0})
print(arrTemp.compactMap{$0})
```

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

```
399
400 var arrTemp = [1,2,4,6,8,5]
401
402 var newArr = arrTemp.reduce(1000, {$0+$1})
403 print(newArr)
```

9.

```
406
407 struct Person {
408     var name : String
409     var age : Int
410 }
411
412 let person1 = Person(name: "Sam", age: 23)
413 let person2 = Person(name: "John", age: 30)
414 let person3 = Person(name: "Rob", age: 27)
415 let person4 = Person(name: "Luke", age: 20)
416
417 let personArray = [person1, person2, person3, person4]
418
419 func findAge(_ arr: [Person]) -> [Person]
420 {
421     let newArr = arr.filter{$0.age>25}
422     return newArr
423 }
424
425 print(findAge(personArray))
```