**1.Explain the declarations @IBOutlet and @IBAction? When do we use @IBOutlet and @IBAction?**

**IBAction**is a keyword and return type in method declarations to denote that a particular method is connected to by an action.

**IBOutlet**is a keyword which is added to a variable declaration. It’s an indicator to that. It does not affect the declaration in any way.

This allows the UI constructed in Interface Builder to receive messages from the code. Equally, the IBAction declaration is a place marker for Interface Builder, allowing us to connect calling actions in response to events happening in the UI to a method.

Neither the IBOutlet or IBAction affect how the code is compiled. They are, place markers written to tell Xcode that an object in the code can be connected to a UI component in Interface Builder.

**2. What are the differences between Structs and Classes? Explain with an example?**

|  |  |
| --- | --- |
| Classes | Structs |
| Classes are the reference types. | Structs are the value types. |
| Allocation of large reference type is cheaper than allocation of large value type. | Allocation and de-allocation is cheaper in value type as compare to reference type. |
| All the reference types are allocated on heap memory. | All the value types are allocated on stack memory. |

**Example**: - value type lives inside the class, Reference type lives inside the class as a pointer to somewhere in heap memory where the actual memory lives.

struct ArticleStruct {

let title: String

let url: URL

var readCount: Int = 0

}

class ArticleClass {

let title: String

let url: URL

var readCount: Int = 0

init(title: String, url: URL) {

self.title = title

self.url = url

}

}

**3. What is optional binding in Swift? Explain it with an example?**

**Optional Binding** : Used to detect whether an optional contains a value or not. And no forced unwrapping in the case.  
**Example :**  
  
if let integ = num{

Print(integ)

}else{

Print(“there is no value”)

}  
  
in above example we can clearly observe num is a Int optional, so we are unwraping the value and storing the data in integ using if condition for checking the value is nil or not.

**4. Mention any of the 3 events associated with UITextField object and explain when do they get triggered?**

**Editing Changed**: when you start to entered when the event will be triggered

**Touch Down:** when you click on that field the event will be triggered

**Touch up :** A touch-up event in the control where the finger is inside the bounds of the control.

**5. Do functions can have multiple returns in Swift? If so, demonstrate that using an example?**

Yes, in Functions can have multiple returns in swift

func sumAndAvg(\_ numbers:Double...)- >(sum:Double,avg:Double){  
 var valu:Double=0  
 var age:Double  
   
 for number in numbers {  
 valu+=number  
 }  
 age=valu/Double(numbers.count)  
   
 return (valu,age)  
}  
   
let result=sumAndAvg(1,20.5,3,4.9,10)  
   
print("Sum = \(result.sum)")  
print("Average = \(result.avg)")

This return can access using like tuples

**6. Explain about closures in Swift?**

Closuresare self-contained blocks of functionality that can be passed around and used in your code. Closures in Swift are like blocks in C and Objective-C and to lambdas in other programming languages.

**7. What is the difference between viewDidLoad() and viewDidAppear()?**

The difference between viewDidLoad and viewDidAppear is that **viewDidAppear** is called every time you can land on the screen while the**viewDidLoad** is only called once when the app loads.

**8. Why do we need to use constraints in our app designing?**

By using constraints to represent the relationship between views, Auto-Layout generates these frames for you. It calculates the size and position on your behalf of the rules you set (the constraints) result in one sound outcome.

**9. What are convenience and required initializers in Swift? Explain briefly about them?**

A class can have more than one designated initializer. It is useful to when you want to provide default values or another custom setup. a **convenience initializer** is a secondary initializer that **must call a designated initializer of the same class**. A class does not require convenience initializers.

**10. What are the differences between arrays and dictionaries in Swift?**

Arrays are ordered collections of values. Sets are unordered collections of unique values. Dictionaries are unordered collections of key-value associations. Arrays, sets, and dictionaries in Swift are always clear about the types of values and keys that they can store.