

Programación orientada a objetos en swift

Clases

```
class forma{
    var lados = Int?//Atributo. Indicamos que el valor es opcional, si no se asigna valor será nil (Equivalente a NULL en este lenguaje)

    init(lados: Int){//Constructor de la clase
        self.lados = lados//El this es Self en Swift
    }

    func descSimple()->String{//metodo
        return "una forma con \ \(lados)"
    }
}
var formilla = forma()//instanciamos un objeto de la clase forma
formilla.lados = 7 //Le asignamos un valor al atributo lados
print(formilla.descSimple())
```

En caso de querer limpiar memoria tras borrar una clase podemos utiliza deinit

Herencia

Para indicar herencia simplemente ponemos dos puntos y la clase de la que se hereda. Se pueden sobrescribir funciones añadiendo un "override" antes del func. Se puede llamar al constructor del padre con "super.init(Atributo:Valor)":

```
Class Poliedro: forma{

    var nuevoAtributo: Int
    init(lados:Int, nuevoAtributo:Int){
        super.init(lados:lados)//llamamos al constructor del padre
        self.nuevoAtributo = nuevoAtributo
    }

    override func descSimple(){
        print("Hola, soy una función heredada reescrita")
    }
}
```

Precondiciones y postcondiciones

Sirven para asignar acciones que se deben hacer antes y después de una tarea.

Enumeraciones y Estructuras

Ambas cosas son prácticamente los mismo, sirven para pasar datos por valor o por referencia. Los Struct se pasan por valor y los enum por referencia. Suele tener que ver con el rendimiento, cuando algo se usa muy a menudo suele ser una clase, pero cuando es algo más temporal se usan structs y enums.

```
struct cartas{
    var rank: Rank
    var suti: Suit
    func simpleDescription()->String{
        return "the \(rank.simpleDescription()) of \(suit.simpleDescription())"
    }
}

enum suit{
    case spades, hearts, diamonds clubs//Equivaldrían a caso 0,1,2 y 3.
    func simpleDescription()->String{
        switch self{
            case .spades
                return "spades"
            case .hearts
                return "hearts"
            case .diamonds
                return "diamonds"
            case .clubs
                return "clubs"
            default:
                return String(self.rawValue)
        }
    }
}
```

From:

<https://www.knoppia.net/> - Knoppia

Permanent link:

<https://www.knoppia.net/doku.php?id=swift:programacionorientadaobjetos>

Last update: **2023/11/23 11:10**

