



Cómputo Móvil

Nombre del Alumno: Cabrera Garibaldi Hernán Galileo
Medel Sánchez Berenice

Nombre del Profesor: Germán Santos Jaimes

Nombre de la materia: Cómputo Móvil

Clave: 674 (Optativa)

Grupo: 1

Nombre de la Actividad: Actividad N° 3

Fecha de entrega: 1 de febrero de 2021

Semestre: 2021-1

Proyecto 3

Planteamiento del Problema

En una taquería de la CDMX se desea saber los gastos aunados a diversos complementos que lleva cada taco como lo es el cilantro, los limones, la salsa, etc. Dicha taquería tiene una gama enorme de tacos de toda la República Mexicana que ofrece a sus comensales (clientes).

Un cliente puede pedir un número N de tacos, sin embargo no todos los tacos llevan los mismos ingredientes, y tampoco es el mismo costo por cada ingrediente (ej. salsa, cilantro, chiles, limones, aguacate, etc, etc) Desarrollar el conjunto de clases, métodos, propiedades, extensiones y/o protocolos que resuelvan las siguientes preguntas. El cliente Pedro, comió 2 tacos de cecina, 1 taco de pastor, 1 taco de suadero, en todas las órdenes pidió solo cebolla y varios limones (6).

¿Cuánto fue el costo total de la cuenta? ¿Cuánto se gastó en cebolla?, suponiendo que cada orden de cebolla costo 2 pesos, ¿cuánto se gastó en limones? tomando en cuenta que cada limón cuesta 3 pesos.

Una pareja de recién casados piden una orden de 6 tacos de carnitas, 1 orden de salsa roja, 8 limones, cilantro y cebolla.

Se deben contestar las mismas preguntas incluyendo ahora el cilantro y la salsa. En cada orden o comanda existen al menos entre 3 o 4 tipos de bebidas diferentes.

Los costes de describen a continuación

- 1 limón = 3 pesos
- 1 orden de cebolla por taco = 2 pesos
- 1 orden de cilantro por taco = 2 pesos
- 1 orden de salsa = 10 pesos
- 1 taco de carnitas = 15 pesos
- 1 taco de suadero = 12 pesos
- 1 taco de pastor = 12 pesos

Resolución del problema

Para el desarrollo de los ejercicios se creó una “command line application” para tener acceso a la lectura de datos desde consola. Se modelaron las siguientes clases:

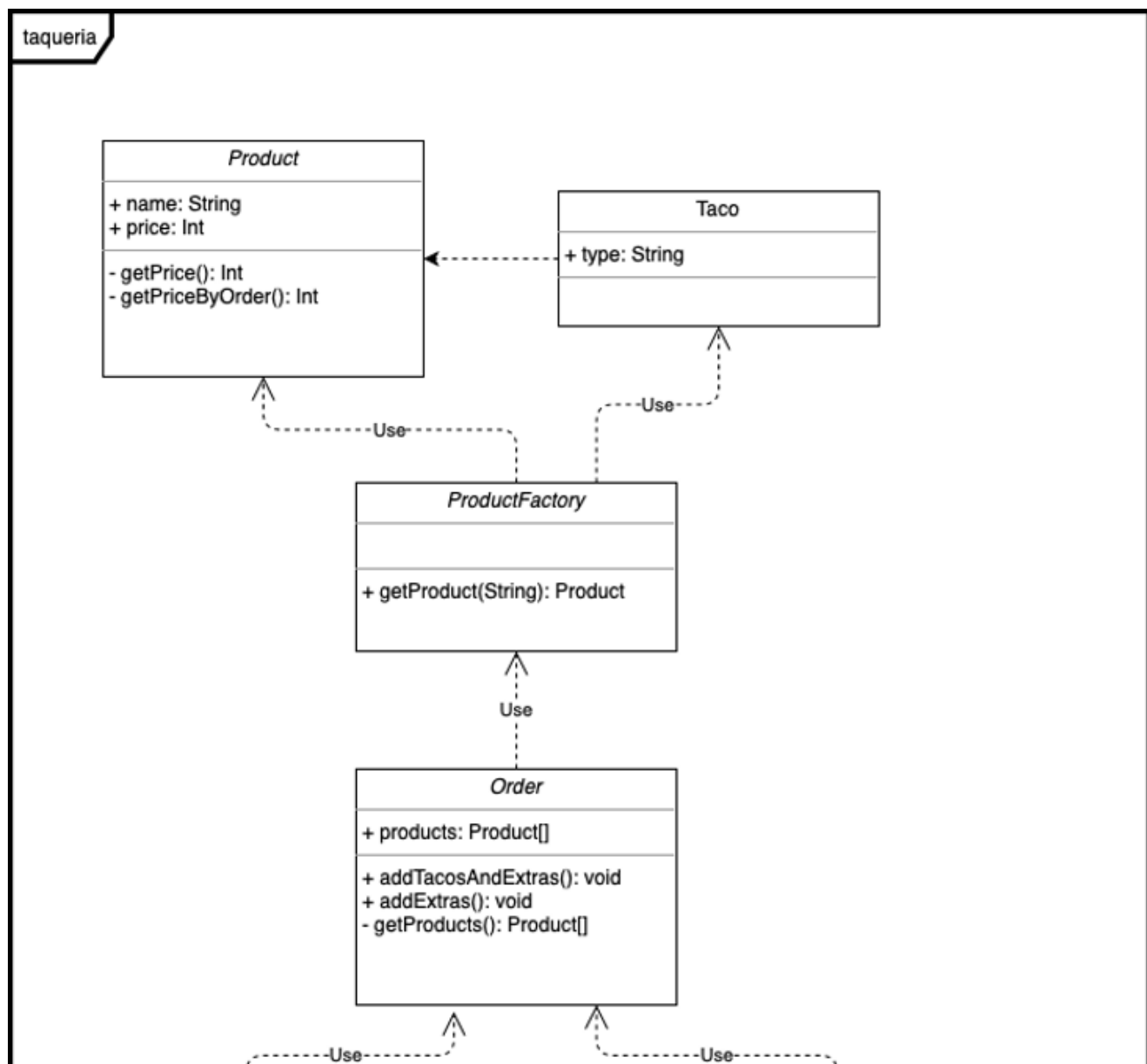
- **Clase Producto:** Contiene el nombre y su precio, para el modelado de algo más específico (como son los tacos), se modeló la clase Taco. Por otro lado, se implementó la clase

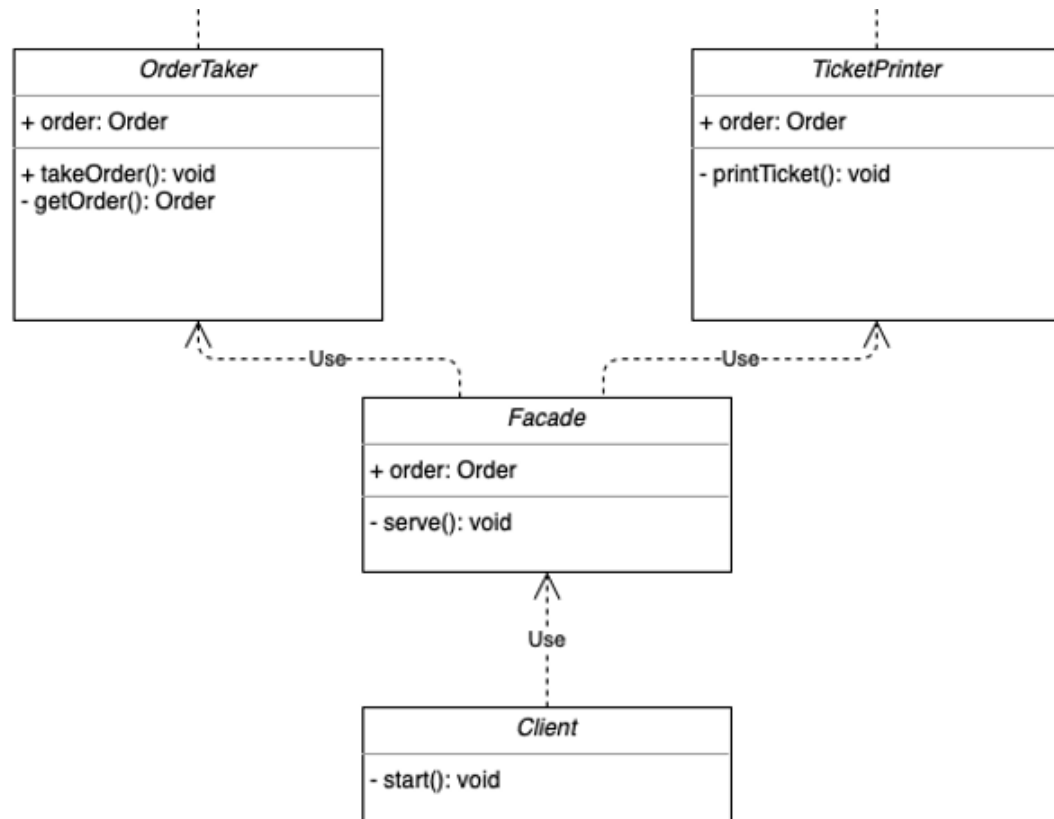
ProductFactory, encargada de la creación de los productos de la taquería, tanto complementos como tacos.

- **Clase Order:** Agrupa los productos que contiene un cliente compra, además contiene una lista de productos que se va modificando cuando un cliente pide más alimentos
- **Order Taker(Subsistema):** mediante el cual se le pide y se registra la orden a un cliente, es un ciclo iterativo donde se le pregunta al cliente su orden hasta que ya no quiera añadir más tacos.
- **TicketPrinter(Subsistema):** Encargada de imprimir los costos y cantidades de los productos que un cliente pidió, extrayendo los objetos Producto de una Orden.
- **Clase Facade:** Para encapsular el funcionamiento de la taquería y la unión de los subsistemas OrderTaker y TicketPrinter. Así como la clase cliente, que es la que consume el servicio de la taquería

Diagrama UML

A continuación se presenta el diagrama UML modelado para este problema





Código

Clase Producto

```

import Foundation

class Product {
    let name: String
    let price: Int
    init(name: String, price: Int){
        self.name = name
        self.price = price
    }
    func getPPriceByOrder(num: Int) -> Int{
        return self.price * num
    }
    func getPrice()->Int{
        return self.price
    }
}

```

Clase Taco

```
class Taco: Product{
    let type: String
    init(type: String, price:Int){
        self.type = type
        super.init(name: "Taco de \$(type)", price:price)
    }
}
```

Clase ProductFactory

Clase en la que se implementa el patrón de diseño *Factory*, por medio de esta clase se pueden crear productos o tacos, sólo se necesita pasar el nombre del producto como parámetro

```
import Foundation

class ProductFactory{
    static func getProduct(type: String)-> Product{
        switch type{
            case "suadero":
                return Taco(type: "Suadero", price: 12)
            case "longaniza":
                return Taco(type: "longaniza", price: 10)
            case "pastor":
                return Taco(type: "pastor", price: 12)
            case "carnitas":
                return Taco(type: "carnitas", price: 20)
            case "cebolla":
                return Product(name: "cebolla", price: 3)
            case "limon":
                return Product(name: "limon", price: 2)
            case "cilantro":
                return Product(name: "cilantro", price: 2)
            case "salsa":
                return Product(name: "salsa", price: 10)
            case "queso":
                return Product(name: "queso", price: 3)
            default:
                return Product(name: "invalid", price: 0)
        }
    }
}
```

Clase Order

Esta clase es la encargada de agrupar los productos del cliente, tiene métodos de crear y agregar productos a la orden. Ya sean tacos y extras, tacossolos o solo extras.

```
import Foundation

class Order{
    var products: [Product]
    init(){
        self.products = []
    }
    func addTacosAndExtras(number: Int, type:String, extras:[String]){
        let tacos = 1...number
        for _ in tacos{
            self.products.append(ProductFactory.getProduct(type:type))
            for extra in extras{
                if extra != "salsa" && extra != "limon"{
                    self.products.append(ProductFactory.getProduct(type:extra))
                }
            }
        }
    }
    func addExtras(number:Int, type:String){
        let count = 1...number
        for _ in count{
            self.products.append(ProductFactory.getProduct(type:type))
        }
    }
    func getProducts() -> [Product]{
        self.products
    }
}
```

Clase Order Taker

Esta clase simula es un subsistema que toma las órdenes de los clientes, es la encargada de crear y llenar la orden de un cliente

```
class OrderTaker{
    var order: Order
    init(){
        order = Order()
    }
    func takeOrder(){
```

```

    let tacosTypes: [Int:String] = [1: "suadero", 2: "longaniza", 3: "pastor", 4:
"carnitas"]
    let tacosComplements: I[Int:String] = [1: "cilantro", 2: "cebolla", 3:
"salsa", 4: "queso", 5:"limon"]
    var isDone: Bool = false
    repeat{
        var tacosNumber = 0
        print("Cuantos Tacos desea?")
        tacosNumber = Int(readLine() ?? "0")!
        if tacosNumber != 0{
            print("De que son sus Tacos?\n1.-Suadero\n2.-Longaniza\n3.-
Pastor\nCarnitas")
            let tacoType = Int(readLine() ?? "1")!
            print("Con que van sus tacos? Sin nada(enter)\n1.-Cilantro\n2.-
Cebolla\n3.- Salsa\n4.-Queso\n5.-Limon")
            if let extrasInput = readLine(){
                let extras = extrasInput.lowercased().split(separator:
",").map({tacosComplements[Intr($0)!]!})
                self.order.addTacosAndExtras(number: tacosNumber,
type:TacosTypes[tacoType]!, extras: extras)
                if extras.contains["salsa"]{
                    self.order.addExtras(number:1, type: "salsa")
                }
                if extras.contains["limon"]{
                    print("Cuantos Limones necesitas?")
                    let lemonsNumber = Int(readLine() ?? "1")!
                    self.order.addExtras(number: lemonsNumber, type: "limon")
                }
            }else{
                self.order.addTacosAndExtras(number:tacosNumber, type:
tacosTypes[tacoType]!, extras[])
            }
        }
        print("Desea algo mas?")
        isDone = Bool(readLine() ?? "true")!
    }while(isDone)
}
func getOrder() -> Order{
    self.order
}
}

```

Clase TickerPrinter

Clase que simula la impresión de Tickers. Imprime las cantidades de productos, así como los totales del pedido y sus costos

[illegible]

Clase Facade

Encargada de agrupar dos subsistemas y llevar a cabo la ejecución de las responsabilidades que el cliente desea consumir. Es decir, tomar sus órdenes y darle el total de su compra y productos.

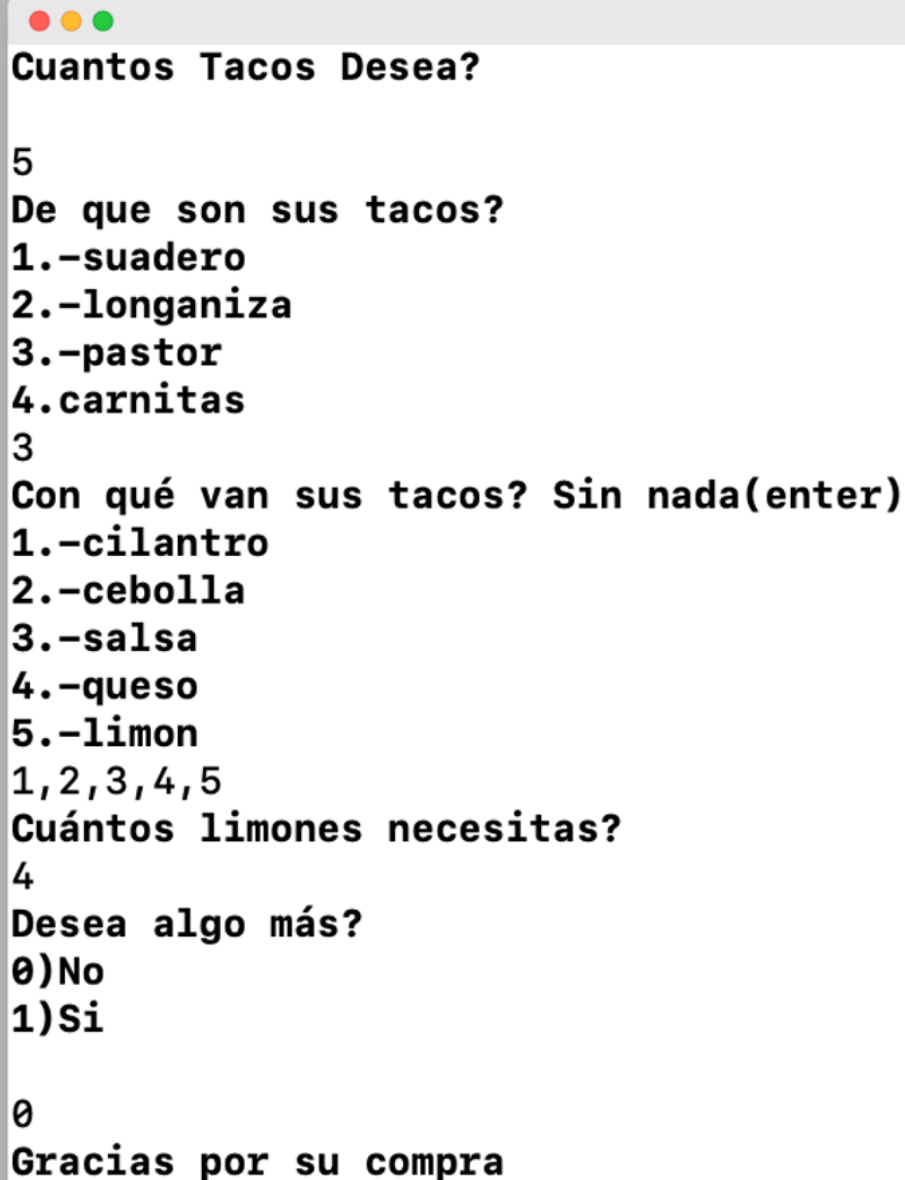
Implementación del patrón de diseño Facade.

```
class Facade{
    static func serve(){
        let orderT = OrderTaker()
        orderT = OrderTaker()
        let order = OrderT.getOrder()
        TicketPrinter.printTicket(order: order)
    }
}
```


Clase Cliente

```
class Client{  
    static func start(){  
        Facade.serve  
    }  
}  
Client.start()
```

Capturas de Pantalla



Cuantos Tacos Desea?

5

De que son sus tacos?

1.-suadero
2.-longaniza
3.-pastor
4.carnitas

3

Con qué van sus tacos? Sin nada(enter)

1.-cilantro
2.-cebolla
3.-salsa
4.-queso
5.-limon

1,2,3,4,5

Cuántos limones necesitas?

4

Desea algo más?

0)No
1)Si

0

Gracias por su compra

```
Producto:      cantidad      total
queso          5             $15
limon          4             $12
Taco de pastor 5             $60
cilantro              5             $10
salsa          1             $10
cebolla        5             $10

Total:                      $117
Program ended with exit code: 0|
```

Conclusiones

Para este proyecto ya tenía mayor dominio y facilidad en el lenguaje de programación Swift, por lo que fue mas sencillo realizar estos 5 programas, sólo tuve algunos tropiezos con mi lógica propia

[Repositorio de Github](#)