DOCUMENTO TÉCNICO

# Requisitos

## Especificación

#RESUMEN DEL PROBLEMA (opcional)

Implementar mediante clases, métodos y atributos una tienda con productos personalizables

### Entrada:

### La entrada constará del ID del cliente y una serie de productos los cuales se irán añadiendo a un carro de compras.

### Salida:

# Mostrar cuales y el precio que debe pagar el usuario a todos los productos que pidió.

# Diseño

## Estrategia

### Para modelar la tienda se hizo uso de dos clases. La primera iba a contener las características del producto donde en este caso fueron diseño, cantidad y fondo que fueron los datos pedidos al cliente. También contiene un objeto mas el cual es el precio, pero este no se pide ya que el precio se tiene en una base de datos de la tienda (diccionario).

### Posteriormente se tiene la segunda clase la cual tiene 3 objetos, el ID de la persona que esta comprando, el carrito ya con todos los productos y el precio final de la compra. Cabe aclarar que el carrito va a tener todos los productos que va pidiendo el cliente en la primera clase y el precio final va tener la suma de todos los precios de todos los productos.

### Al final simplemente se muestra el carrito completo y el precio final que se calcula accediendo a la posición precio de cada producto por aparte e irla sumando así obteniendo el precio final de la compra. Además, al precio final se le impuso el IVA el cual es del 19%

### #Estructuras de datos

### Diccionarios, listas

#ALGORITMO

Programa adjunto

#INVARIANTE

Texto

Descripción generada automáticamente

* Iniciación: False, 0
* Estabilidad: Por cada recursión, va analizando si encuentra el elemento en el carrito ya existente
* Terminación: Actualiza el elemento si esta y si no esta lo añade al carrito

Texto

Descripción generada automáticamente

* Iniciación: False, 0
* Estabilidad: Por cada recursión, va analizando si encuentra el elemento en el carrito ya existente
* Terminación: Borra el elemento si esta y si no está imprime error.

## Casos de prueba

|  |  |  |
| --- | --- | --- |
| **Entrada** | **Justificación** | **Salida** |
|  |  |  |
| Eliminar un elemento | (Probar la funcionalidad del método eliminar) | Carrito sin ese elemento |
| Añadir un elemento | (Probar la funcionalidad del método actualizar) | El carrito con ese elemento añadido |
| Añadir un elemento ya existente | (Probar la funcionalidad del método actualizar) | El carrito con ese elemento, pero con el precio y la cantidad actualizado |

# Análisis

# Texto Descripción generada automáticamenteTexto Descripción generada automáticamenteTexto Descripción generada automáticamenteTexto Descripción generada automáticamenteTexto Descripción generada automáticamenteTexto Descripción generada automáticamente

## Temporal

La complejidad de este código es aproximadamente de n, ya que en el proceso mas complejo que es el de actualizar un elemento o eliminarlo, si tomamos como el peor de los casos que recorra todo el carrito de compras, al final va a estar recorriendo toda la lista, es decir una complejidad de n. Los demás procesos tienen complejidad constante ya que son crear clases, verificar datos y operaciones matemáticas que se consideran que en Python son eficientes.

# Código

*Al igual que el punto anterior pueden guiarse con este ejemplo*

## Documentación

Dentro del código.

## Fuentes

*Las fuentes son el código sobre el cual escribieron el documento*

/arena\_1\_a