SANTO®
TOMÁS

ShopFast

Nombre: Cristian Patricio Villalobos Velasquez

Asignatura: Estructura de datos y algoritmos

Actividad: Examen semana 8

CONTENIDO

DISENO DEL SISTEMA

El sistema de gestión de pedidos de ShopFast está diseñado para automatizar las operaciones de una tienda en línea, utilizando programación orientada a objetos (POO) para modelar las entidades principales del negocio: clientes, productos y pedidos.

DISEÑO Y ESTRUCTURA DEL SISTEMA

El diseño del sistema se basó en la separación de responsabilidades, dividiendo el código en archivos modulares para facilitar su comprensión, mantenimiento y reutilización:

- **cliente.py**: Contiene la clase Cliente, que modela a los clientes de la tienda con atributos como RUT, nombre, correo y teléfono.
- **producto.py**: Incluye la clase Producto, que gestiona los productos con su código, nombre, precio y stock disponible. Esta clase también tiene un método para descontar el stock cuando se realiza una compra.
- **pedido.py**: Define la clase Pedido, que maneja la lógica de las compras. Utiliza un diccionario para almacenar los productos y sus cantidades. Implementa métodos para agregar, eliminar productos, y calcular el total de la compra, asegurando que se apliquen las validaciones de stock necesarias.
- **app.py**: Actúa como el archivo principal que orquesta todas las operaciones. Importa las clases de los otros archivos y contiene la lógica principal para interactuar con el usuario, solicitar datos del cliente, mostrar productos, crear pedidos y presentar un resumen final.

FUNCIONALIDADES Y CONTROL DEL FLUJO

El sistema está diseñado para ser interactivo y fácil de usar, guiando al usuario a través del proceso de compra:

- **Gestión de Datos**: Utiliza listas para almacenar clientes y diccionarios para los productos y los ítems del pedido, lo que permite una organización eficiente y un acceso rápido a la información.
- **Estructuras de Control**: Aplica estructuras de decisión (if) e iteración (for, while) para validar el stock disponible, recorrer elementos y gestionar el flujo de la aplicación. Por ejemplo, antes de agregar un producto al pedido, se verifica que haya stock suficiente para la cantidad solicitada.

- **Interacción con el Cliente**: Después de que el usuario selecciona los productos, se le presenta un resumen del pedido y se le da la opción de modificar o eliminar productos antes de finalizar la compra, lo que añade flexibilidad al proceso. Finalmente, se muestra el total a pagar y el stock actualizado.

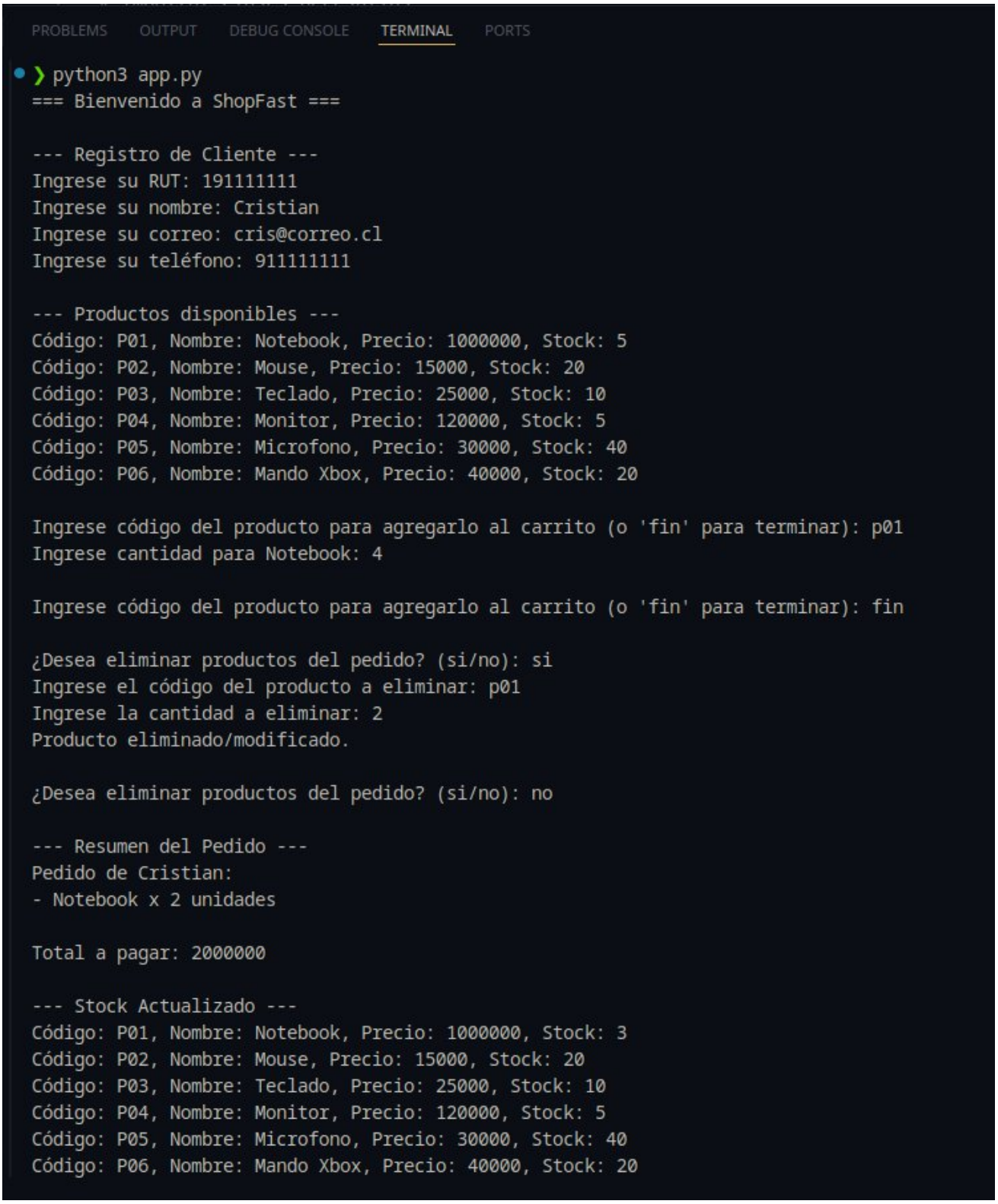
El código incluye comentarios detallados para facilitar su entendimiento y cumple con los requisitos del examen final, que solicita la implementación de clases, el uso de estructuras de control y la elaboración de un reporte de *testing* para verificar su correcto funcionamiento

EJECUCION DE LA APP

Basándonos en la ejecución proporcionada, se puede observar lo siguiente:

PROCESO DE COMPRA

1. **Registro de Cliente**: El usuario, identificado como "Cristian" con RUT "191111111", inicia la aplicación y registra sus datos personales.
2. **Selección de Producto**: Se muestra la lista de productos disponibles. El cliente agrega el producto "Notebook" con el código "p01" (que transforma en mayusculas con .upper) y una cantidad de 4 unidades a su carrito.
3. **Modificación del Pedido**: El cliente decide modificar su pedido. Ingresa "si" para eliminar productos y reduce la cantidad de "Notebook" en 2 unidades, resultando en un pedido final de 2 unidades de este producto.
4. **Resumen y Finalización**: El sistema muestra un resumen del pedido final. Se confirma que el pedido es de "Cristian", con "Notebook x 2 unidades", y el total a pagar es de 2,000,000.
5. **Actualización de Stock**: Finalmente, se presenta el stock actualizado de los productos. El stock de "Notebook" ha disminuido de 5 a 3 unidades, lo que refleja correctamente la cantidad de 2 unidades vendidas.



Ejecucion de la aplicacion

RESULTADO DEL TESTING

- Caso de Prueba 1: Adición de Productos y Cálculo de Total: El programa agregó productos al pedido y calculó el total correctamente. El resultado obtenido ($1,030,000) coincidió con el resultado esperado, confirmando que la funcionalidad de compra básica es precisa.
- Caso de Prueba 3: Eliminación de Producto y Devolución de Stock: Al eliminar un producto del pedido, el sistema devolvió la cantidad correspondiente al inventario. El stock final obtenido (5 unidades) fue igual al stock final esperado, verificando que la lógica de eliminación y actualización de stock es correcta.
- Caso de Prueba 2: Validación de Stock: Se probó el escenario donde no hay suficiente stock. El sistema impidió la venta, mostró un mensaje de error y no modificó el inventario. El resultado obtenido (el producto no se agregó) y el stock final (5 unidades) coincidieron con lo esperado, demostrando que el control de stock funciona de manera adecuada.

```
--- Ejecutando Caso de Prueba 1: Agregar productos y calcular total ---
Resultado obtenido: 1030000
Resultado esperado: 1030000
Test 1: OK
.
--- Ejecutando Caso de Prueba 3: Eliminar producto y devolver stock ---
Resultado obtenido (stock final): 5
Resultado esperado (stock final): 5
Test 3: OK
.
--- Ejecutando Caso de Prueba 2: Validación de stock ---
No hay stock suficiente para Notebook
Resultado obtenido (agregado): False
Resultado esperado (agregado): False
Resultado obtenido (stock final): 5
Resultado esperado (stock final): 5
Test 2: OK
.
----------------------------------------------------------------------
Ran 3 tests in 0.000s

OK
```

Ejecucion de Unitest

CONCLUSIÓN REFLEXIVA

Con base en la implementación y las pruebas realizadas, se puede concluir que el sistema de gestión de pedidos de ShopFast cumple eficazmente con los objetivos planteados en el caso de estudio. La separación del código en clases y archivos distintos demuestra una sólida aplicación de la **Programación Orientada a Objetos** (POO). Esta estructura no solo facilita la comprensión y el mantenimiento del sistema, sino que también permite la reutilización de código y la modularidad.

El uso de **estructuras de control** (if, for, while) e iteración resultó ser fundamental para validar el stock y gestionar los datos del pedido. Esto se complementa con el uso de listas y diccionarios , que organizan de manera eficiente a los clientes y productos, optimizando la gestión de la información.

A pesar de los desafíos iniciales, como los errores de entrada de datos, el sistema demostró ser robusto y confiable, como se confirmó en el reporte de testing. La capacidad de agregar, eliminar y modificar productos, junto con la actualización automática del inventario y el cálculo del total, valida el correcto funcionamiento del programa. Este ejercicio no solo permitió aplicar los conocimientos teóricos de la asignatura, sino que también brindó una experiencia práctica valiosa en el desarrollo de un sistema funcional.

___ **Codigo fuente en: Github Codigo fuente Examen S8** ___