Trabajo Práctico: Patrones de Diseño – Sistema de Facturación

Objetivo

Implementar un sistema simple de facturación usando patrones de diseño. El objetivo es separar la lógica de cálculo de descuentos e impuestos de manera que el sistema sea modular y sea fácil agregar o cambiar funcionalidades sin modificar el código principal.

Descripción del problema

La entrada del sistema serán archivos JSON con el siguiente formato:

```
None
{ "items": [
  { "id": 1, "product_category": "category_a", "price": 10.0 },
  { "id": 2, "product_category": "category_b", "price": 20.0 }
], "discount": "student" }
```

Donde:

- `discount`: indica el descuento a aplicar.
  - (campo vacío): Sin descuento.
  - `student`: Descuento de estudiante (10%).
  - `black_friday`: Descuento de Black Friday (30%).
- `id`: Identificador de cada ítem.
- `product_category`: Categoría del producto, que puede ser:
  - food_item
  - cellphone
  - computer
  - car
  - imported_car
- `price`: Precio de cada ítem (a criterio del alumno).

La salida tendrá este formato:

```
None
{
  "subtotal": 0.0,
  "applied_discount": 0.0,
  "item_taxes": [
    { "id": 1, "tax": 0.0 },
    { "id": 2, "tax": 0.0 }
  ],
  "total_taxes": 0.0,
  "final_total": 0.0
}
```

Donde:

- `subtotal`: Suma total antes de aplicar descuentos e impuestos.
- `applied_discount`: Monto de descuento aplicado.
- `item_taxes`: Lista de impuestos aplicados a cada ítem.
- `total_taxes`: Suma total de los impuestos.
- `final_total`: Total final después de aplicar descuentos e impuestos.

Relación entre impuestos y categorías

Los productos pagan impuestos según su categoría:

- **IVA (20%)**: Aplica a todos los productos **excepto** los de tipo `food_item`.
- **Derechos de importación (30%)**: Se aplican **solo** a productos importados como `imported_car`, `cellphone` y `computer` se consideran importados.

De este modo, los impuestos se determinan de forma modular según la categoría, lo que permite ampliar fácilmente el sistema si se agregan nuevas categorías o tipos de gravámenes.

Para mantener la solución modular se recomienda implementar el cálculo de impuestos y descuentos mediante el **patrón Strategy**. También puede considerarse el uso del **patrón Chain of Responsability** para aplicar impuestos y descuentos item por item. Pensar bien en los pros y contras de cada alternativa.

Referencias:

- Ejemplos de implementacion en python: https://refactoring.guru/design-patterns/python

TODO para v2:

- Agregar test cases
- Aclarar que se pueden usar tools de IA (valien bien el resultado)
- Aclarar que tiene que hacerse codigo funcional
- Pedir que armen el diagrama de clases para explicar como funciona el solucion