Actividad collections

Actividad evaluable con Listas

Se desea crear un programa para administrar una cafetería, en la que tengamos unas mesas establecidas y a estas les podamos agregar productos, que tendrán nombre y precio. Podremos añadir productos a una mesa. También podemos cobrar una mesa, lo que implica obtener la cuenta y vaciar la lista de productos añadidos en la mesa.
Se puede hacer un recuento de caja, que indica el total de dinero cobrado a las mesas hasta el momento.

El menú ofrecido por la aplicación es:

1. Consultar mesas.
2. Consultar el recuento de caja actual.
3. Añadir producto a una mesa.
4. Cobrar mesa.
5. Abrir mesa.

Detalles del problema:

Mesa

Una mesa tiene un número que la identifica. Hay varias mesas, cuyo número de especifica al inicio del programa (mediante un prompt como el siguiente):

```
Número de mesas abiertas:
```

Después, sepueden abrir nuevas mesas si es necesario mediante la opción 5.

Al cobrar una mesa, se vacía la lista de productos asociada y se agrega la cantidad a la caja.

Productos

Tienen un nombre y un precio. Los productos serán 4 y se añaden al inicio de la aplicación.

1. Churro (0,50€)
2. Café con leche (1€)
3. Tostada (1,5€)
4. Zumo de naranja (2€)

Consultar mesas

Se mostrará un listado de mesas y los productos que contiene.

Consultar el recuento de caja actual

Se mostrará un mensaje con el total, que debe ajustarse a lo pagado hasta el momento.

```
Total: 35,5 €
```

Añadir producto a mesa

Para añdir un producto a la mesa, se mostrará un submenú como el siguiente:

```
    Número de mesa [1-5]: 4
    ---------------------
            Mesa 4
    ---------------------

    Número de producto [0 -> terminar. -1 -> borrar cuenta]: 2
    Cantidad: 3
    Número de producto [0 -> terminar. -1 -> borrar cuenta]: 1
    Cantidad: 1
    Número de producto [0 -> terminar. -1 -> borrar cuenta]: 3
    Cantidad: 2
    Número de producto [0 -> terminar. -1 -> borrar cuenta]: 0

    3 x Cafe con leche
    1 x Churro
    2 x Tostada

    ---------------
```

> Esto añadiría 3 cafés con leche, 1 churro y 2 tostadas a la mesa 4.

Cobrar mesa

Para cobrar una mesa, se preguntará el número de mesa, se mostrará la cuenta, antes de vaciar la lista de productos y sumar la cantidad a la caja.

```
Número de mesa: 4
Cuenta:
        - 3 x Café con leche
        - 1 x Churro
        - 2 x Tostada
        --------------------
        4,50€
```

Abrir mesa

Abrir una mesa sólo pide confirmación antes de abrirla. Cada nueva mesa utiliza el siguiente número.

```
¿Abrir nueva mesa [s]/n? s

Mesa número 6 abierta
```

Arquitectura

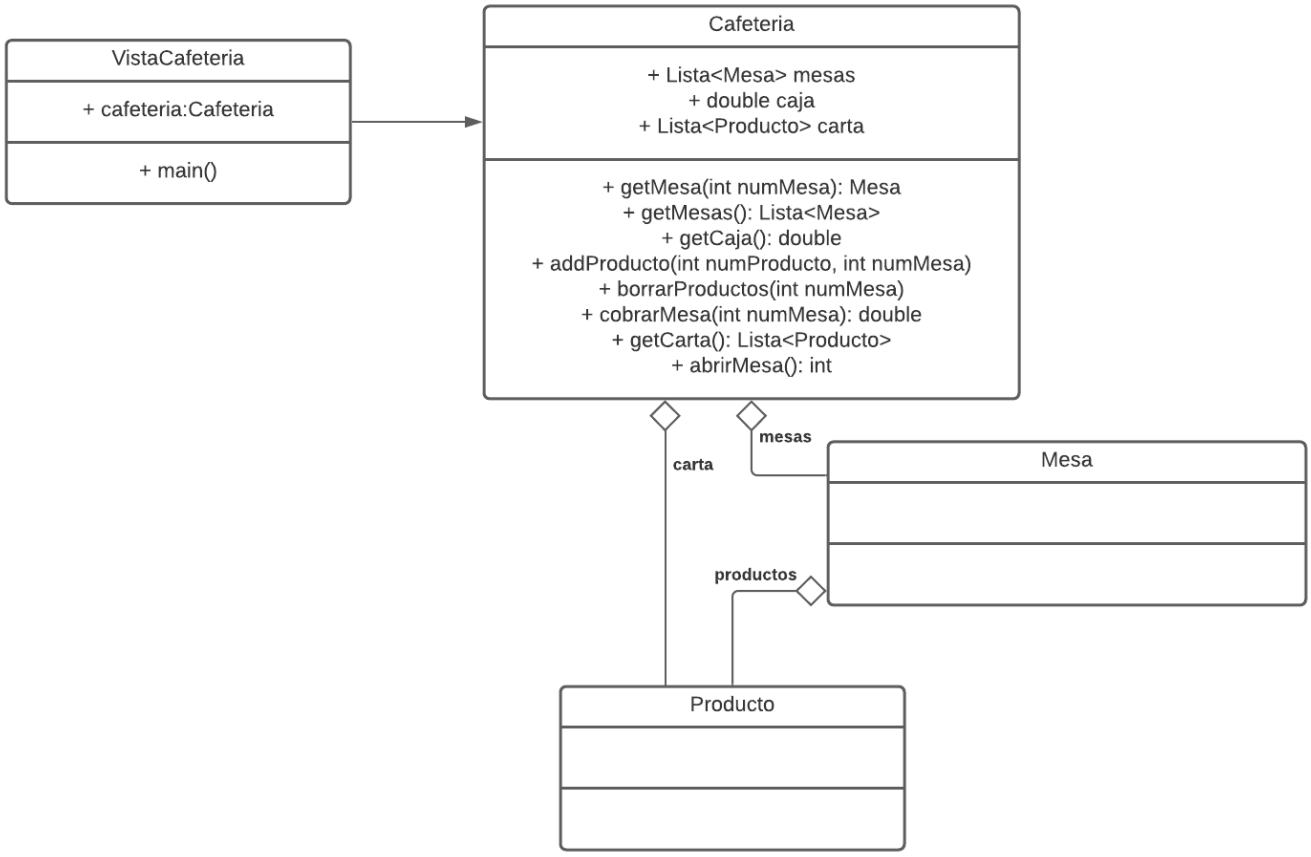
Las aplicaciones suelen organizarse arquitectónicamente buscando la seperación de responsabilidades. Así, por ejemplo, una arquitectura habitual se basa en la siguiente idea:

- Vista: incluye todo lo necesario para la interacción del usuario. Es el conjunto de clases que contiene ventanas, menús, etc. También incluye referencias a la lógica de negocio, en forma de atributos privados. Esta es la parte más externa de la aplicación.
- Lógica de negocio: incluye las clases que modelan el problema. Contiene el código necesario para llevar a cabo los procesos necesarios (como por ejemplo, cobrar una mesa y pasar el total a caja). Estas clases tienen una interfaz (métodos públicos) que son utilizados por la vista. Es común crear una clase *fachada* que incluya los métodos necesarios en la vista, y que redirija las llamadas a las clases de la lógica de negocio. Por ejemplo, en este problema, esa clase podría llamarse *Cafeteria*.

> NOTA: Esta forma de separar la vista y la lógica de negocio, permite que la vista sea intercambiable. Por ejemplo, podríamos añadir otro tipo de vista, porque la lógica de negocio funciona igual la use quien quien la use.

- Capa de Datos: la información de una aplicación suele almacenarse en una capa de persistencia, como una base de datos. Todo lo relativo a la interacción con la capa de persistencia, es llamado capa de datos. En nuestro caso aun no tenemos esa necesidad, pero surgirá en próximas clases.

Se propone la siguiente arquitectura:



> El diagrama está incompleto, y sólo está definida una parte.

Se pide:

- Escribir una aplicación para resolver el problema
- Respetar las indicaciones sobre menús de usuario y salidas
- Respetar la interfaz (métodos) de la clase *Cafeteria*