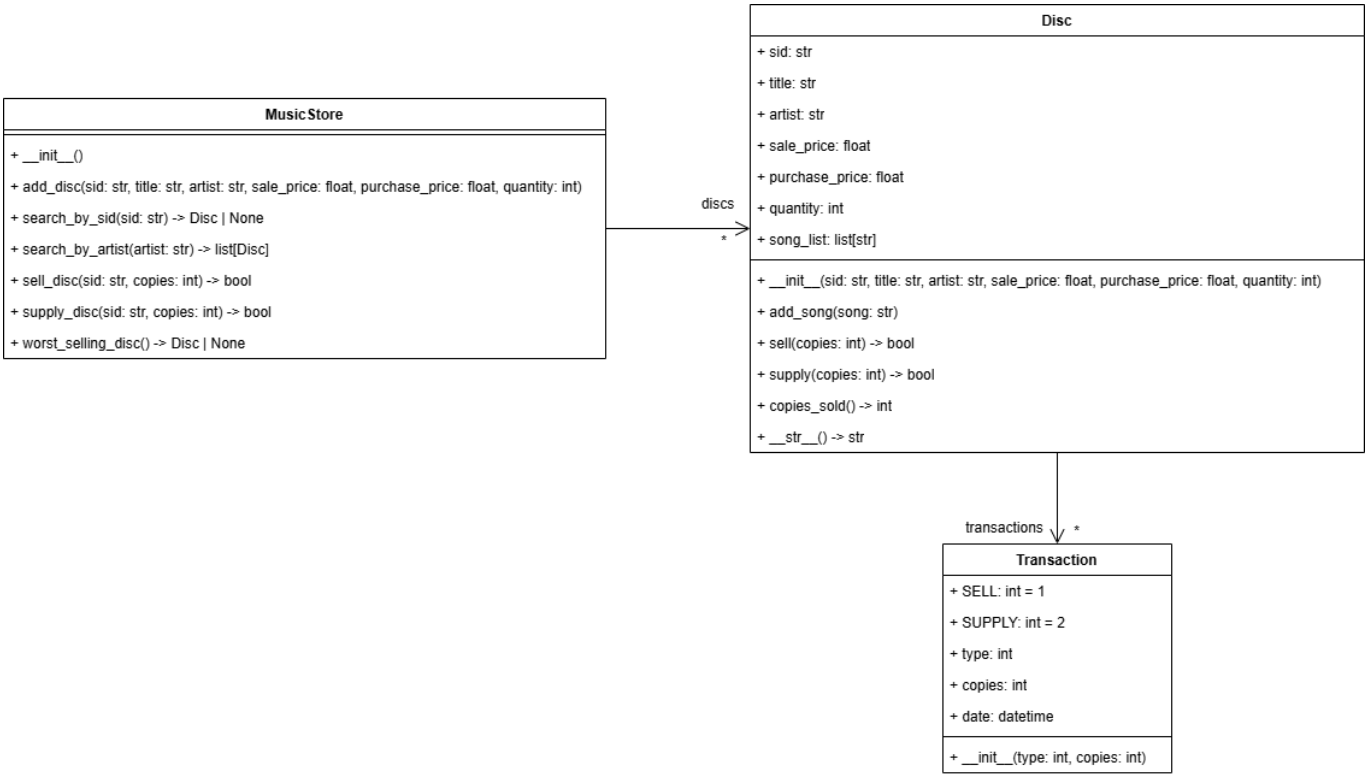
Tienda de música

La tienda de música es una aplicación utilizada para evaluar el conocimiento de los conceptos de POO en Python. La aplicación es una simple tienda de música que permite a los usuarios agregar, listar y buscar discos de música. La aplicación está implementada utilizando clases y objetos en Python.

El modelo de la aplicación es el siguiente:



El código de la aplicación está incompleto, la idea es completarlo teniendo en cuenta los siguientes pasos.

1. Completa la clase `Transaction` teniendo en cuenta los siguientes requisitos:
    - La clase debe tener una constante `SELL` de tipo `int` con valor `1`.
    - La clase debe tener una constante `SUPPLY` de tipo `int` con valor `2`.
    - La clase debe tener un método `__init__` que reciba los siguientes parámetros:
        - `type` de tipo `int`.
        - `copies` de tipo `int`.

      En el método `__init__`, la clase inicializa los atributos `type` y `copies` con los valores recibidos como parámetros.
    - La clase debe tener un atributo `date` de tipo `datetime` que debe inicializarse con la fecha y hora actual

      > **Sugerencia:** puedes usar la función `datetime.now()` para obtener la fecha y hora actual.

2. Completa la clase `Disc` teniendo en cuenta los siguientes requisitos:

- La clase debe tener un método `__init__` que reciba los siguientes parámetros:
  - `sid` de tipo `str`.
  - `title` de tipo `str`.
  - `artist` de tipo `str`.
  - `sale_price` de tipo `float`.
  - `purchase_price` de tipo `float`.
  - `quantity` de tipo `int`.

  En el método `__init__`, la clase debe inicializar los atributos `sid`, `title`, `artist`, `sale_price`, `purchase_price` y `quantity` con los valores recibidos como parámetros.

- La clase debe tener los atributos `transactions` de tipo `list[Transaction]` y `song_list` de tipo `list[str]`. Ambos atributos deben inicializarse como listas vacías.

- La clase debe tener un método de instancia `add_song` que reciba un parámetro `song` de tipo `str` y agregue la canción a la lista `song_list`.

- La clase debe tener un método de instancia `sell` que reciba un parámetro `copies` de tipo `int` y haga lo siguiente:
  - Si el parámetro `copies` es mayor que el atributo `quantity` del disco, el método debe devolver `False`.
  - De lo contrario, el método disminuye el atributo `quantity` del disco en el valor del parámetro `copies` y agrega un nuevo objeto `Transaction` a la lista `transactions` con el tipo `Transaction.SELL` y el número de `copies` vendidas.
  - El método debe devolver `True`.

- La clase debe tener un método de instancia `supply` que reciba un parámetro `copies` de tipo `int` y haga lo siguiente:
  - Aumenta el atributo `quantity` del disco en el valor del parámetro `copies`.
  - Agrega un nuevo objeto `Transaction` a la lista `transactions` con el tipo `Transaction.SUPPLY` y el número de `copies` suministradas.

- La clase debe tener un método de instancia `copies_sold` que devuelva un `int` con el número total de copias vendidas.

  > **Sugerencia:** puedes sumar el número de copias de cada transacción de tipo `Transaction.SELL`.

- La clase debe tener un método de instancia `__str__` que devuelva un `str` con el formato:

```
SID: {sid}
Title: {title}
Artist: {artist}
Song List: {song_list}
```

Donde `{sid}`, `{title}` y `{artist}` deben ser reemplazados por los valores de los atributos del disco. El `{song_list}` debe ser reemplazado por la lista de canciones del disco separadas por una coma y un espacio.

> **Sugerencia:** Usa un f-string (`f""`) para formatear la cadena y `\n` dentro de la cadena para una nueva línea.

3. Completa la clase `MusicStore` teniendo en cuenta los siguientes requisitos:

   - Debe tener un método `__init__` que inicialice el atributo `discs` de tipo `dict[str, Disc]` como un diccionario vacío.

   - La clase debe tener un método de instancia `add_disc` que reciba los parámetros `sid` de tipo `str`, `title` de tipo `str`, `artist` de tipo `str`, `sale_price` de tipo `float`, `purchase_price` de tipo `float` y `quantity` de tipo `int` y haga lo siguiente:

     - Verifica si el `sid` no está en el diccionario `discs`.
     - Si el `sid` no está en el diccionario `discs`, el método crea un nuevo objeto `Disc` con los parámetros recibidos y lo agrega al diccionario `discs` utilizando el `sid` como clave.

   - La clase debe tener un método de instancia `search_by_sid` que reciba el parámetro `sid` de tipo `str` y devuelva `Disc | None`. El método debe devolver el disco con el `sid` recibido como parámetro o `None` si el disco no se encuentra.

   - La clase debe tener un método de instancia `search_by_artist` que reciba el parámetro `artist` de tipo `str` y devuelva `list[Disc]`. El método debe devolver una lista con todos los discos que tengan el `artist` recibido como parámetro.

   - La clase debe tener los métodos de instancia `sell_disc` y `supply_disc` que reciban los parámetros `sid` de tipo `str` y `copies` de tipo `int`. Copia el siguiente código en la clase `MusicStore` para completar los métodos:

     ```
     def sell_disc(self, sid: str, copies: int) -> bool:
         disc = self.search_by_sid(sid)
         if disc is None:
             return False

         return disc.sell(copies)

     def supply_disc(self, sid: str, copies: int) -> bool:
         disc = self.search_by_sid(sid)
         if disc is None:
             return False

         disc.supply(copies)
         return True
     ```

   - La clase debe tener un método de instancia `worst_selling_disc` que devuelva `Disc | None` con el disco que ha vendido el menor número de copias o `None` si no hay discos en la tienda.