Persistencia de los Objetos en la Base de Datos

Historial de cambios

Fecha	Versión	Descripción
23/09/2024	1.0	Documento inicial
06/10/2024	2.0	Aclaración en 1. Introducción y mención a nuevo atributo **volatilidad**. Actualización del script de creación de tablas en la BD en 2. Base de Datos Se eliminó la frase: “Esta información sólo debe mantenerla en memoria.” en 3. Funcionalidades del sistema > Generar Stock

Tabla de Contenido

1. Introducción

En el Entregable 1 se trabajó en el modelo de clases de una Billetera Virtual. Generalmente una vez que el modelo de clases está acordado, suele trabajarse en el resto de la aplicación, donde se sumarán nuevas clases con otras responsabilidades. Recuerde que cuando trabajamos con programación orientada a objetos, es clave la delegación de responsabilidades.

El objetivo de esta práctica es realizar una prueba de concepto (POC) que consiste en implementar una pequeña sección de ese modelo de clases que haciendo uso de alguna estrategia de conversión/mapeo nos permita persistir la información contenida en los objetos en una Base de Datos. El subconjunto de clases que usaremos en esta entrega corresponde a las Monedas y Transacciones. Adicionalmente para simplificar esta POC, no tenemos cuenta el usuario asociado.

En este Entregable surge un nuevo atributo asociado a las Criptomonedas: la **volatilidad**. El atributo **volatilidad** en el contexto de criptomonedas generalmente se mide como un valor numérico que indica la magnitud de las fluctuaciones en el precio de un activo. Los valores válidos para la volatilidad son normalmente de 0 a 100 (con posibles decimales). No necesita modificar su diseño del Entregable 1, pero agregue el atributo en la clase para esta entrega.

2. Base de Datos

Para realizar este entregable, usaremos SQLite, una base de datos de distribución simple, que no necesita instalación, sólo necesita incorporar una librería (**archivo .jar**) dentro de su proyecto. Puede descargarlo desde la siguiente URL en la sección **Files**

https://mvnrepository.com/artifact/org.xerial/sqlite-jdbc/3.46.1.2

Si ya tiene instalado un motor de base de datos como MySQL o MariaDB también puede usarlo.

A continuación se presenta una sección de código que necesitará cuando escriba su solución. Esta sección de código corresponde a la creación de las tablas (estructura interna) de la base de datos:

```
/**
        * Este método se encarga de la creación de las tablas donde se
       * almacenará la
        * información de los objetos. Una vez establecida una conexión, debería
        * ser lo próximo a ser ejecutado.
        *
        * @param connection objeto conexion a la base de datos SQLite
        * @throws SQLException
        */
```

```
private static void creaciónDeTablasEnBD(Connection connection) throws
                                                                SQLException {
            Statement stmt;
            stmt = connection.createStatement();
            String sql = "CREATE TABLE MONEDA "
                        + "("
                        + " TIPO       VARCHAR(1)    NOT NULL, "
                        + " NOMBRE       VARCHAR(50)    NOT NULL, "
                      + " NOMENCLATURA VARCHAR(10)  PRIMARY KEY   NOT NULL, "
                        + " VALOR_DOLAR     REAL     NOT NULL, "
                        + " VOLATILIDAD     REAL     NULL, "
                        + " STOCK    REAL     NULL "  + ")";
            stmt.executeUpdate(sql);
            sql = "CREATE TABLE ACTIVO_CRIPTO"
                  + "("
                  + " NOMENCLATURA VARCHAR(10)  PRIMARY KEY     NOT NULL, "
                   + " CANTIDAD REAL    NOT NULL " + ")";
            stmt.executeUpdate(sql);
            sql = "CREATE TABLE ACTIVO_FIAT"
                  + "("
                  + " NOMENCLATURA VARCHAR(10)  PRIMARY KEY     NOT NULL, "
                   + " CANTIDAD REAL    NOT NULL " + ")";
            stmt.executeUpdate(sql);
            sql = "CREATE TABLE TRANSACCION"
                  + "("
                  + " RESUMEN VARCHAR(1000)   NOT NULL, "
                   + " FECHA_HORA            DATETIME  NOT NULL " + ")";
            stmt.executeUpdate(sql);
            stmt.close();
      }
```

3. Funcionalidades del Sistema

Ud. deberá entregar una aplicación Java que se ejecute por línea de comando y que permita realizar las siguientes operaciones:

1. **Crear Monedas**.
   - Se solicita al usuario que ingrese: tipo (Cripto o FIAT), nombre, nomenclatura, valor en dólar y el stock disponible. Una vez ingresados los datos, si el usuario confirma se guardan en la Base de Datos.
   - Para el tipo de moneda el usuario no debería poder ingresar cualquier valor, debería estar acotado a las 2 posibles opciones.
   - Si el usuario confirma, se guarda en la base de datos.

2. **Listar Monedas**
   - Muestra en pantalla información de las monedas disponibles.

- Si bien las monedas se mostrarán ordenadas por valorEnDolar, debe ser capaz de ordenarse por nomenclatura. Se espera que use alguno de los mecanismos de interface vistos en teoría.

3. **Generar Stock**
   - De manera aleatoria genera una cantidad de monedas disponibles para todos los usuarios de la Billetera.

4. **Listar Stock**
   - Muestra en pantalla información del stock disponible.
   - Si bien el Stock se mostrará ordenado por cantidad de manera descendente, debe ser capaz de ordenarse por nomenclatura. Se espera que use alguno de los mecanismos de interfaces vistos en teoría.

5. **Generar Mis Activos** (o como ud. haya nombrado a la cantidad de una determinada moneda en posesión del usuario)
   - Permitir ingresar la cantidad y la nomenclatura.
   - Si el usuario confirma, se guarda en la base de datos. La nomenclatura ingresada debe existir entre las criptomonedas creadas en el punto anterior para guardarse en la base de datos, caso contrario el programa informa un error.

6. **Listar Mis Activos**
   - Muestra en pantalla información de los activos disponibles.
   - Si bien los Activos se mostrarán ordenados por cantidad de manera descendente, debe ser capaz de ordenarse por nomenclatura de la moneda. Se espera que use alguno de los mecanismos de interfaz vistos en teoría.

7. **Simular una COMPRA de Criptomoneda.** Al seleccionar esta opción se ingresará una criptomoneda a comprar, una moneda FIAT y la cantidad disponible en esa moneda FIAT. A modo de ejemplo, si se ingresa: BTC, ARS, 63081.39 el sistema calcula el equivalente en ARS y devuelve 0.001 BTC.
   - Si la criptomoneda a comprar no es aún un activo, se crea el nuevo activo.
   - Si la criptomoneda a comprar ya existe, simplemente se incrementa la cantidad y se decrementa la cantidad en FIAT.
   - Si el usuario confirma la operación se realiza la actualización de los activos y se guarda **sólo una descripción de la transacció**n en la Base de Datos. Si la cantidad de FIAT ingresado no es suficiente, se devuelve un error.
   - Se espera que verifique el Stock disponible.
   - Se espera que haga uso de las clases de su modelo que le permiten representar “una compra” antes de guardarlo en la Base De Datos

8. **Simular un SWAP.** Es decir: al seleccionar esta opción se ingresará una criptomoneda a convertir, una cantidad y la criptomoneda esperada. A modo de ejemplo, si se ingresa: BTC, 5, ETH y el sistema calcula el equivalente en ETH de 5 BTC.
   - Tanto la criptomoneda a convertir como la criptomoneda esperada deben encontrarse entre **Mis Activos**.
   - Si el usuario confirma la operación se realiza la actualización de los activos.
   - Se guarda **una descripción** de la transacción en la base de datos.

- Se espera que verifique el Stock disponible.
- Se espera que haga uso de las clases de su modelo que le permiten representar “un swap” antes de guardarlo en la Base De Datos

4. De la Implementación

A. Se espera que implemente una aplicación de escritorio, donde la interacción se realiza a través de la línea de comando.
B. No se requiere que implemente todo su modelo de clases, **sólo un conjunto mínimo de clases** que permita realizar las operaciones requeridas en la sección anterior.
C. Debe implementar el patrón DAO visto en teoría.

5. Contenido de la Entrega

La entrega incluye un archivo ZIP que contiene lo siguiente:

A. Un archivo .jar ejecutable con la aplicación
B. Un archivo readme con cualquier particularidad o información que considere necesario tener en cuenta para ejecutar su aplicación.

6. Información de Referencia

Uso de SQLite
https://www.tutorialspoint.com/sqlite/sqlite_java.htm

Statements:
https://docs.oracle.com/javase/tutorial/jdbc/basics/processingsqlstatements.html

Prepared Statements:
https://docs.oracle.com/javase/tutorial/jdbc/basics/prepared.html

Recuperando datos de Result Set:
https://docs.oracle.com/javase/tutorial/jdbc/basics/retrieving.html