Práctica 3

UGR

Desarrollo de Software

Autores:

- Alba Ayala Carmona
- María Ramos Martínez
- Sofía González Uceda
- Natalia Serrano Cerceda

Índice

1. Introducción

Esta práctica consiste en realizar el diseño e implementación software de un sistema bancario a partir de unos requisitos que se nos proporcionan. Además, debemos realizar una serie de test para comprobar el correcto funcionamiento de este sistema.

Para realizarla, hemos tomado de base el UML que se nos proporcionó, al que le hemos realizado algunos cambios que hemos considerado oportunos.

1.1. Cambios realizados en el UML

En primer lugar, hemos añadido una clase `User`, que representa al cliente de nuestro sistema, en la cual tenemos su nombre y una lista de todas sus cuentas. Además, hemos añadido los métodos necesarios para añadir cuentas a estos usuarios.

En todas las clases que necesitan un `id` único (`Transaccion` y `Cuenta`), hemos implemetado un método para generarlos (`generateId()`). Para ello, hemos usando el paquete `uuid`. También hemos añadido los `getters` y `setters` que hemos visto convenientes.

Para implementar la clase `TransferTransaction` nos hemos decantado por la opción de pasarle al constructor por parámetro una cuenta (la cuenta *to*) y al método `apply` pasarle la otra cuenta (*from*). De esta forma mantenemos la herencia planteada, en la que las clases que heredan de `Transaction` tienen un método `apply` al que se le pasa un único parámetro.

Por último, en la clase `BankService` hemos añadido más operaciones que han sido necesarias para comprobar los test y poder interactuar con la interfaz gráfica correctamente. Además, hemos cambiado el `map<String, Account>` por un `set<User>`, para tener agrupados las cuentas con sus usuarios.

2. Interfaz Gráfica

Hemos realizado una interfaz gráfica que nos permite crear cuentas indicándole el nombre de usuario. Si tenemos ya ese usuario creará una cuenta para dicho usuario y si el usuario no existía antes, lo creará y después le creará sus cuentas.

Luego tenemos otro apartado en el que realizamos las operaciones. Para ello, primero tenemos dos filas con dos desplegables cada uno: la primera fila se usa en todas las operaciones, en cambio, la segunda fila solo se usará para la operación de *transferir*; en la tercera fila, volvemos a tener otro deplegable para elegir la transacción que se va a realizar. Por último, hemos añadido un *textField*, para obtener la cantidad necesaria para la operación.

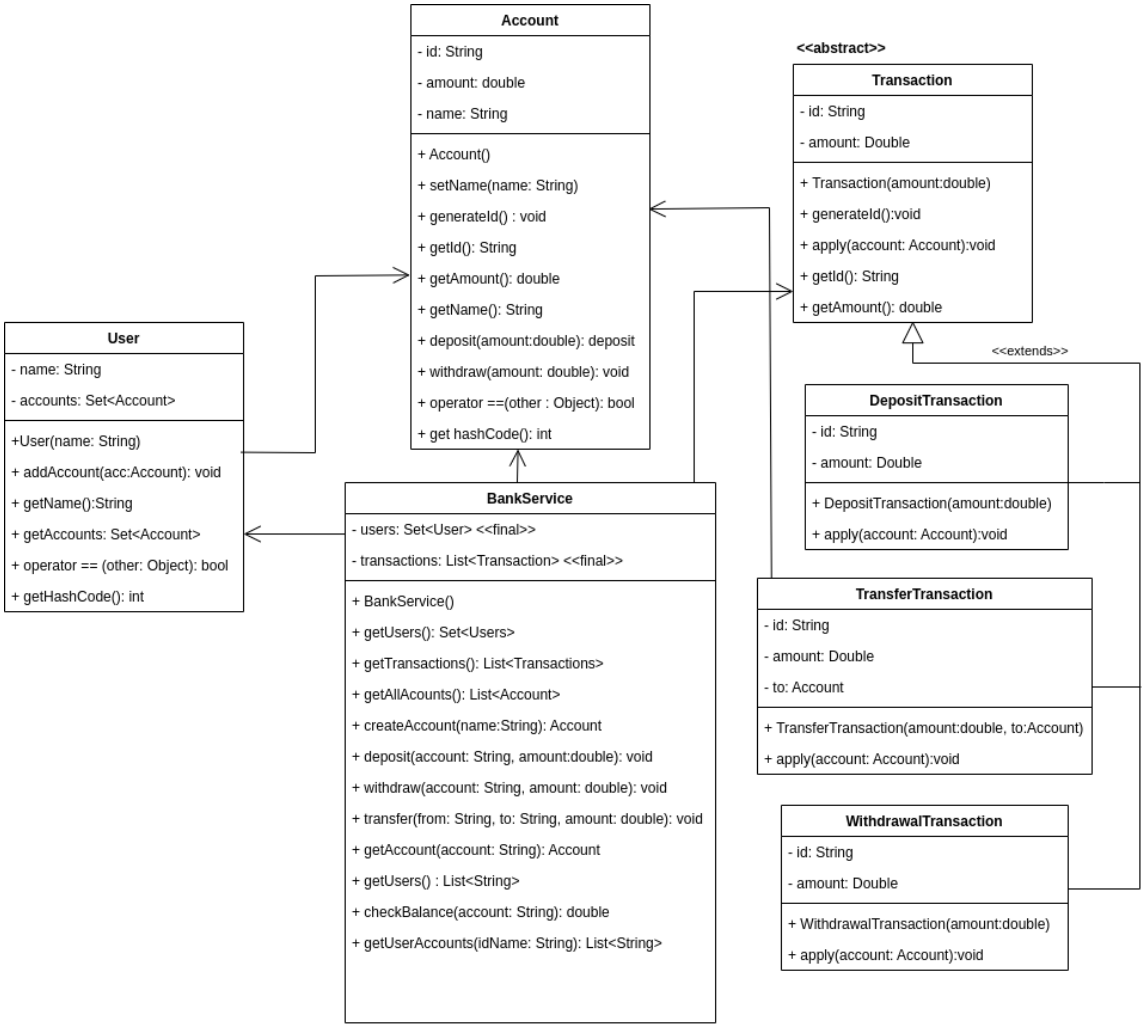
Finalmente, hemos añadido un mensaje que indica tanto si se ha podido realizar con éxito la operación (mostrando también el resultado, si corresponde) como si no se ha podido llegar a completar.

3. Explicación test

Una vez implementado todo el código de nuestro sistema, hemos realizado los test que se pedían en el guion y comprobado que se pasen todos.

Además, para la clase `Usuario` hemos incluido otros test para comprobar que un usuario se crea sin cuentas asociadas y que cuando se cree una se le asocie correctamente.

4. UML



(a) Nuevo patrón