Trabajo Práctico #1

Descripción de la situación:
Un banco maneja muchas cuentas bancarias. Cada cuenta está asociada a una sola persona. Las cuentas pueden ser en dólares o en pesos. La distinción es importante, ya que si bien las dos cuentas tienen una persona como titular, un saldo y un CBU asociado, hacen un diferente manejo de las operaciones depositar y retirar fondos. Las personas poseen nombre, apellido, número de documento y una dirección en donde viven. Al banco le interesa poder consultar por las provincias y ciudades donde viven las personas.

Tareas a realizar:
1. Escriba el diagrama de clases para los objetos que considere necesarios.
2. Implemente en Java la siguiente secuencia:
   - creación de un banco
   - creación de 3 cuentas bancarias en pesos
   - creación 2 cuentas bancarias en dólares
   - deposite 1000 pesos en todas las cuentas en pesos
   - deposite 100 dólares en todas las cuentas en dólares
   - extraiga 400 pesos de una cuenta en pesos
   - extraiga 50 dólares de una cuenta en dólares
   - deposite 20 dólares en una cuenta en dólares
   - intente extraer 5000 pesos de una cuenta en pesos
   - transfiera dinero de una cuenta en dólares a una cuenta en pesos
   - transfiera dinero de una cuenta en pesos a una cuenta en dólares
   - obtener todas las cuentas de personas de una provincia
   - obtener todas las cuentas de personas de una ciudad

Consideraciones:
- Las personas deben ser creadas con todos sus datos en el constructor.
- Al ser creadas todas las cuentas tienen un saldo 0.
- El número de CBU se genera automáticamente para cada cuenta.
- Ninguna de las cuentas puede operar al descubierto (extraer más dinero del que tiene).
- El banco permite hacer transferencias entre dos cuentas sin importar el tipo de cuenta, el mismo banco realiza la conversión entre las divisas.

Estrategia de resolución:

El programa tiene tres clases, Banco, Persona y Cuenta. Para que cada cuenta esté asociada a una persona, se utiliza el DNI de la persona y se crea un CBU único que se asigna automáticamente a cada instancia de la clase Cuenta.

La distinción para el tipo de cuenta se hace a través de números enteros. Que una cuenta sea de tipo 1, significa es una cuenta en pesos, mientras que si es de tipo 2 es una cuenta en dólares.

Mi lógica para crear los métodos depositoExtracion y transferencia fue la siguiente:

depositoExtraccion se encarga, cómo su nombre lo indica, de sumar o restar el saldo de las instancias de la Clase cuenta, dependiendo de la operación (incluyendo a las transferencias, además de los depósitos y extracciones). Si bien, al principio pensaba codificar todas las operaciones en este método, me di cuenta que para poder implementar las transferencias iba a tener que utilizar todavía más 'if' anidados y validaciones complejas. Entonces, decidí crear un método a parte para esa lógica. El código está comentado detalladamente, por lo que no creo que sea necesario explicar mucho más en este PDF.

Aclaraciones:

En el diagrama UML, escribí los métodos para obtener y establecer simplemente como 'getters y setters()', porque no me pareció necesario agregar cada uno al diagrama.

Hice el trabajo en este orden: diagrama UML, código, relaciones entre las clases en el diagrama. Al hacer esta última parte, me percaté de que hubiera sido mejor llamar 'Cliente' a la clase 'Persona', ya que suena raro decir que 'Un banco puede tener una o varias personas' en vez de 'clientes'. No sé si ese detalle afecte en algo la nota por el diagrama, pero quería aclararlo por las dudas (lo dejé así porque, sinceramente, no quería editar tanto código por un detalle menor cómo este).