Ejercicio cuentas bancarias – clases y métodos abstactos

Desarrollar en Python un software que implemente un modelo de gestión bancaria de acuerdo a los siguientes requerimientos:

Se debe crear una **clase abstracta** llamada **CuentaBancaria,** una clase **CajaDeAhorro** y una clase **CuentaCorriente,** ambas que hereden de CuentaBancaria. Una clase **Cliente** que entre sus atributos tenga una **lista de elementos de tipo CuentaBancaria**, y una clase **Banco** que entre sus atributos tenga una **lista de elementos de tipo Cliente**.

La clase **CuentaBancaria**, debe contener:

- **Atributos de instancia**:
  - **nro_cuenta** de tipo string.
  - **cbu** de tipo string.
  - **alias** de tipo string.
  - **saldo** de tipo float.
  - **movimientos** de tipo list.
- **Método constructor:**
  - **__init__ (**self, *args**)** para inicializar los atributos de instancia nro_cuenta, cbu, alias y saldo con los valores recibidos como parámetros; y el atributo movimientos como una nueva lista.
- **Métodos de instancia**:
  - **consultar_saldo ()**, que retorna el saldo de la cuenta.
  - **depositar (**monto_a_depositar de tipo float**)**, que retorna un booleano indicando si la operación se realizó correctamente o no. En caso que el depósito se haya realizado correctamente, y luego de haber actualizado el saldo, debe agregar un nuevo elemento a la lista movimientos, donde elemento será una tupla que tendrá los siguientes elementos: fecha de tipo date, "depósito", monto depositado y saldo.
- **Métodos abstractos**:
  - **extraer (**monto_a_extraer de tipo float**)**, que retorna un valor booleano. En caso que la extracción se haya realizado correctamente, y luego de haber actualizado el saldo, debe agregar un nuevo elemento a la lista movimientos, donde elemento será una tupla que tendrá los siguientes elementos: fecha de tipo date, "extracción", monto extraído y saldo.
  - **transferir (**monto_a_transferir de tipo float, cuenta_destino de tipo CuentaBancaria**)**, que retorna un valor booleano. En caso que la transferencia se haya realizado correctamente, y luego de haber actualizado el saldo, debe agregar un nuevo elemento a la lista movimientos, donde elemento será una tupla que tendrá los siguientes elementos: fecha de tipo date, "transferencia", monto transferido y saldo.

La clase **CajaDeAhorro** debe contener:

- **Atributos de instancia**:
  - **monto_limite_extracciones** de tipo float.
  - **monto_limite_transferencias** de tipo float.
  - **cant_extracciones_disponibles** de tipo int.
  - **cant_transferencias_disponibles** de tipo int.

- **Método constructor:**
  - **__init__(self, args *)** para inicializar los atributos de instancia nro_cuenta, alias, cbu, saldo, monto_limite_extracciones, monto_limite_transferencias, cant_extracciones_disponibles y cant_transferencias_disponibles con los valores recibidos como parámetros.
- **Métodos de instancia**:
  - **extraer (**monto_a_extraer de tipo float**)**, que retorna un booleano indicando si la operación se realizó correctamente o no. La extracción se considerará como correcta si monto_a_extraer es mayor a cero, si monto_a_extraer no es superior al saldo, si monto_a_extraer no es superior al monto_limite_extracciones y si la cant_extracciones_disponibles es mayor a cero, en cuyo caso se deberá actualizar el saldo y cant_extracciones_disponibles; e invocar al método heredado para registrar el movimiento.
  - **transferir (**monto_a_transferir de tipo float, cuenta_destino de tipo CuentaBancaria**)**, que retorna un booleano indicando si la operación se realizó correctamente o no. La transferencia se considerará como correcta si monto_a_transferir es mayor a cero, si monto_a_transferir no es superior al saldo, si monto_a_transferir no es superior al monto_limite_transferencias y si la cant_transferencias_disponibles es mayor a cero, en cuyo caso se deberá actualizar el saldo y cant_transferencias_disponibles de la cuenta origen y el saldo de la cuenta destino; e invocar al método heredado para registrar el movimiento.

La clase **CuentaCorriente** debe contener:

- **Atributos de instancia**:
  - **monto_maximo_descubierto** de tipo float.
- **Método constructor:**
  - **__init__(**self, *args**)** para inicializar los atributos de instancia nro_cuenta, alias, cbu, saldo y monto_maximo_descubierto con los valores recibidos como parámetros.
- **Métodos de instancia**:
  - **extraer (**monto_a_extraer de tipo float**)**, que retorna un booleano indicando si la operación se realizó correctamente o no. La extracción se considerará como correcta si monto_a_extraer es mayor a cero, si monto_a_extraer no es superior al saldo más el monto_maximo_descubierto, en cuyo caso se deberá actualizar el saldo; e invocar al método heredado para registrar el movimiento.
  - **transferir (**monto_a_transferir de tipo float, cuenta_destino de tipo CuentaBancaria**)**, que retorna un booleano indicando si la operación se realizó correctamente o no. La transferencia se considerará como correcta si monto_a_transferir es mayor a cero, si monto_a_transferir no es superior al saldo más el monto_maximo_descubierto, en cuyo caso se deberá actualizar el saldo de la cuenta origen y el saldo de la cuenta destino; e invocar al método heredado para registrar el movimiento.

La clase **Cliente** debe contener:

- **Atributos de instancia**:
  - **razon_social** de tipo string.
  - **cuit** de tipo string.
  - **tipo_persona** de tipo string (física o jurídica).
  - **domicilio** de tipo string.
  - **cuentas_bancarias** de tipo list.

- **Método constructor:**
  - **__init__(**self, *args**)** para inicializar los atributos de instancia razon_social, cuit, tipo_persona y domicilio con los valores recibidos como parámetros, y el atributo cuentas_bancarias como una nueva lista.
- **Métodos de instancia**:
  - **crear_nueva_cuenta_bancaria ()**, que retorna un booleano indicando si la operación se realizó correctamente. Para crear una nueva instancia de CuentaBancaria se debe indicar el tipo de cuenta (CajaDeAhorro o CuentaCorriente) e ingresar nro_cuenta, alias, cbu, saldo y demás valores necesarios según el tipo de cuenta. El nuevo objeto CuentaBancaria debe agregarse a la lista cuentas_bancarias.

La clase **Banco** debe contener:

- **Atributos de instancia**:
  - **nombre** de tipo string.
  - **domicilio** de tipo string.
  - **clientes** de tipo list.
- **Método constructor:**
  - **__init__(**self, *args**)** para inicializar los atributos de instancia nombre y domicilio con los valores recibidos como parámetros, y el atributo clientes como una nueva lista.
- **Métodos de instancia**:
  - **crear_nuevo_cliente ()**, que retorna un booleano indicando si la operación se realizó correctamente. Para crear una nueva instancia de Cliente se debe ingresar razon_social, cuit, tipo_persona y domicilio. El nuevo objeto Cliente debe agregarse a la lista clientes.

Para validar el modelo de gestión bancaria implementado, incluya una función **main()** con las instrucciones necesarias para crear **un objeto Banco**, **tres instancias de Clientes** (como mínimo) cada uno de ellos con **dos objetos CuentaBancaria** (como mínimo), uno de tipo **CajaDeAhorro** y otro de tipo **CuentaCorriente**.
Luego simule varias **operaciones de depósito, extracción y transferencias** entre las cuentas.
Finalmente, muestre por pantalla los datos de los clientes del banco, con los saldos de sus cuentas y el registro de los movimientos de las mismas.

Nota:
Todos los **atributos de instancia** declarados en las clases **deberán ser privados** y accesibles mediante **propiedades públicas** de tipo getters y setters (usando el **decorador @property**).
Para trabajar con **fechas** debe importar el **módulo datetime**.