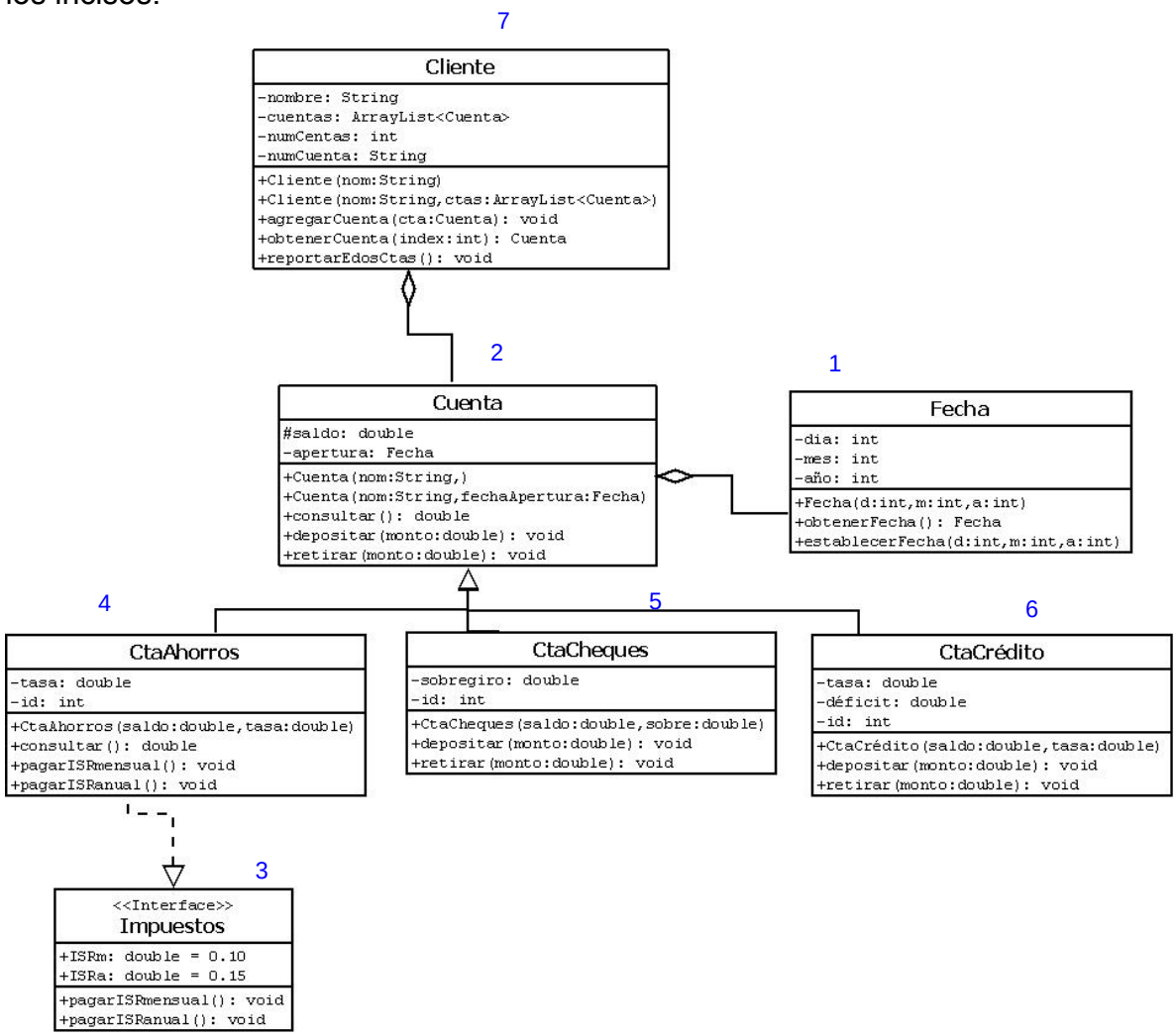
Programación Orientada a Objetos

Práctica 3: Agregación de clases, herencia de clases, interfaces

Objetivo: Realizar la implementación de los conceptos de Agregación de clases y Herencia de clases, así como de interfaces, mediante una aplicación en Java

Desarrollo: Con base en el diagrama de clases mostrado a continuación resuelve los incisos.



a) Implementar la clase **`CtaAhorros`**:

- La cuenta de ahorros deberá manejar una **tasa de interés** (atributo **`tasa`**).
- Cada vez que se invoque el método **`consultar()`** se debe revisar la fecha de apertura (para determinar la fecha de corte), y calcular si ha transcurrido un mes. Si ha transcurrido un mes (30 días) entonces se agregará el interés devengado al saldo; de lo contrario el saldo permanecerá sin cambio.

agregar el interes

un cliente puede tener varias cuentas

- El atributo **`id`** será usado para identificar, mediante un valor consecutivo iniciando con 1, cada una de las cuentas de ahorro que tenga un cliente.
- Esta clase debe implementar la interfaz **`Impuestos`**: Al completar un mes a partir de la fecha de apertura, se hace el cálculo (**`pagarISRmensual`**) sobre el saldo que tiene el cliente y este será el pago el cual se descontará al mismo valor del saldo (siempre y cuando el saldo sea mayor que 10,000 pesos); al completar un año a partir de la fecha de apertura, se hace el cálculo (**`pagarISRanual`**) sobre el saldo que tiene el cliente y este será el pago el cual se descontará al mismo valor del saldo (siempre y cuando el saldo sea mayor que 50,000 pesos);

b) Implementar la clase **`CtaCheques`**.

dinero que te da el banco de inicio

- La cuenta de cheques deberá manejar un **`sobregiro`** (atributo **`sobregiro`**).

si monto es mayor que el saldo echar mano

- El método **`retirar()`** debe en primer lugar, verificar si el **`monto`** a ser retirado puede ser cubierto con el **`saldo`**. De ser el caso, entonces se resta el **`monto`** al **`saldo`** y la operación termina. Si el **`monto`** es mayor que el **`saldo`**, entonces se debe echar mano tanto del **`saldo`** como del **`sobregiro`**: primero se tomarán los recursos del **`saldo`** y después se completará con los recursos del **`sobregiro`**. Si el **`monto`** no se cubre con la suma del **`saldo`** y el **`sobregiro`**, entonces se debe enviar un mensaje indicando que no se cuenta con los recursos suficientes para realizar la operación de retiro, y no se realiza la operación.

Si con el sobregiro no alcanza no se hace la operacion

- Los cheques caducan al año de la fecha de apertura de la cuenta: no podrá entonces llevarse a cabo ningún retiro de la cuenta de cheques.
- El método **`depositar`** deberá cubrir en primer lugar el valor del **`sobregiro`** que se adeude al banco, y luego entonces se hará el depósito en el atributo **`saldo`**.
- El atributo **`id`** será usado para identificar, mediante un valor consecutivo iniciando con 1, cada una de las cuentas de cheques que tenga un cliente.

c) Implementar la clase **`CtaCrédito`**.

- La cuenta de crédito deberá manejar una tasa de interés (atributo **`tasa`**) que será la cantidad, por concepto de intereses, que mensualmente debe pagar el cliente.
- Cuando el usuario retira una cantidad mayor a su **`saldo`** el atributo **`déficit`** comenzará a crecer. Cada mes será aplicada la tasa de interés al atributo **`déficit`**.
- El método **`depositar`** deberá cubrir en primer lugar el valor del **`déficit`** que se adeude al banco, y una vez cubierto entonces el depósito se aplicará al atributo **`saldo`**.
- El atributo **`id`** será usado para identificar, mediante un valor consecutivo iniciando con 1, cada una de las cuentas de crédito que tenga un cliente.

d) Implementar la clase **`Cliente`**.

- Esta clase deberá usar un **`ArrayList`** de objetos de la clase **`Cuenta`** los cuales pueden ser objetos de la clase **`CtaAhorros`**, **`CtaCrédito`**, o **`CuentaCheques`**; no hay un orden preciso en los tipos de cuenta, tampoco una cantidad prestablecida, sin embargo es importante usar todos los tipos de cuenta.
- El atributo **`numCtas`** indica el número de cuentas que tiene asignadas el cliente. Mediante los constructores inicializamos objetos de la clase **`Cliente`** con base en un nombre, o con base en un nombre y un **`ArrayList`** de objetos **`Cuenta`** los cuales deben ser creados previamente.
- El método **`reportarEdosCtas`** desplegará en pantalla el listado de cada una de las cuentas del cliente. Esto es, se reportará: el tipo de cuenta (ahorros, cheques, crédito), el identificador (id) de cada cuenta, todos los movimientos realizados en cada cuenta (depósito y retiros) con sus respectivas fechas.

e) Implementar la clase usuaria de la clase **Cliente**, por ejemplo la clase **Banco**; hacer lo siguiente:

- Crear instancias (al menos tres) de la clase **Cliente** y en cada una de las instancias hacer depósitos, retiros y consultas de saldo. Por ejemplo, se puede crear el cliente "Quijote" y asignarle dos cuentas de ahorro (**`CtaAhorro`**), una cuenta de cheques (**`CtaCheques`**), y una cuenta de crédito (**`CtaCrédito`**) mediante el método **`agregarCuenta()`**.El ArrayList **cuentas**, de tipo **Cuenta**, de la clase **Cliente**, admite cualquier tipo de cuenta (**`CtaAhorro`**, **`CtaCheques`**, y **`CtaCrédito`**, son todas ellas de tipo **`Cuenta`**, y esta es una característica de la herencia de clases).
- Usar la referencia al cliente "Quijote" para hacer consultas, retiros, y depósitos tanto a las cuentas de ahorro como a la de cheques y la de crédito.

Nota: Para este inciso es requisito usar los métodos del diagrama UML, no se requiere agregar más métodos.

- Un cliente puede tener cualquier cantidad de cuentas asignadas, tanto de ahorro como de cheques o de crédito.
- El método **`reportarEdosCtas`** de la clase **Cliente** debe reportar los movimientos realizados en cada cuenta (de ahorro, de cheques, y de crédito) del cliente, reportando los tipos de movimientos, las fechas de realización, y los montos correspondientes.

Por ejemplo, para el cliente "Quijote" con número de cuenta "QX1600", que tiene dos cuentas de ahorro (ambas con tasa del 10%), una de cheques, y una de crédito (con tasa 30%), podríamos generar el siguiente reporte:

Nombre del cliente: Quijote
Número de cuenta: QX400
Apertura: 01-03-2015 (día-mes-año)

Cuenta de Ahorros id=1:

Movimiento	Fecha	Detalle
Consulta	01-03-2015	saldo= 1000.0
Consulta	02-04-2015	saldo = 1100.0 (El saldo aumentó 10% por pago de intereses)
Retiro	04-04-2015	monto = 500.0
Consulta	07-04-2015	saldo = 600.0
Depósito	08-04-2015	monto = 300.0
Consulta	09-04-2015	saldo = 900.0
Consulta	02-05-2015	saldo = 9900.0 (El saldo aumentó 10% por pago de intereses)

Cuenta de Ahorros id=2:

Movimiento	Fecha	Detalle
Consulta	01-02-2015	saldo= 3000
Consulta	02-03-2015	saldo = 3300 (El saldo aumentó 10% por pago de intereses)
Retiro	15-03-2015	monto = 1200
Consulta	16-03-2015	saldo = 1100
Depósito	17-03-2015	monto = 500
Consulta	20-03-2015	saldo = 1600
Consulta	02-04-2015	saldo = 1760 (El saldo aumentó 10% por pago de intereses)

Cuenta de Cheques id=1:

Movimiento	Fecha	Detalle
Consulta	10-02-2015	saldo= 2000 sobregiro = 1000
Consulta	05-03-2015	saldo = 2000 sobregiro = 1000
Retiro	08-03-2015	monto = 1200
Consulta	09-03-2015	saldo = 800 sobregiro = 1000
Retiro	10-03-2015	monto = 1300
Consulta	11-03-2015	saldo = 0 sobregiro = 500
Consulta	02-04-2015	saldo = 0 sobregiro = 500
Depósito	10-05-2015	monto = 1900
// Con el monto de 1900 se cubren los 500 faltantes del sobregiro, los		
// restantes 1400 se van al saldo del cliente		
Consulta	15-05-2015	saldo = 1400 sobregiro = 1000
//La cuenta de cheques es cancelada después de un año de la apertura		
Retiro	11-02-2016	monto = 1000
Transacción no permitida: cuenta cancelada		

Cuenta de Crédito id=1:

Movimiento	Fecha	Detalle
Consulta	10-02-2015	saldo= 1500 déficit = 0
Consulta	05-03-2015	saldo = 1500 déficit = 1000
Retiro	08-03-2015	monto = 1200
Consulta	09-03-2015	saldo = 300 déficit = 0
Retiro	10-03-2015	monto = 1300
Consulta	11-03-2015	saldo = 0 déficit = 1000

Consulta	12-04-2015	saldo = 0	déficit = 1150 (15% de interés fue aplicado)
Depósito	15-04-2015	monto = 1000	
Consulta	12-04-2015	saldo = 0	déficit = 150

Nota 1. En la clase **Cliente** y en la clase **Cuenta** se tendrán que agregar los atributos y métodos necesarios para implementar este reporte. **También es factible agregar una nueva clase, siempre y cuando se preserve el diseño original de la jerarquía de clases del diagrama UML**.

Presentación de la práctica:

- Presentar el programa en ejecución.
- **Presentar un escenario de prueba para cada aspecto a evaluar, de forma que se cubran todos los requerimientos de esta práctica. No se revisará la práctica si no se tiene el escenario de prueba.**
- Sustentar un breve examen oral acerca del código presentado y del programa en ejecución.
- Presentar individualmente o en equipo (máximo dos integrantes).
- Presentar diagrama UML en caso de haber hecho modificaciones al diagrama original.

Tabla de Evaluación:

Concepto	Puntos
Diseño de la jerarquía de clases **(Presentar diagrama UML usado en esta práctica, esto es, todas las interfaces y clases junto con sus miembros)**	**1**
Clase Cliente **Constructores y métodos** **Método reportarEdosCtas**	**1** **2**
Clase Cuenta **Constructores y métodos**	**1**
Clase CtaAhorros **Constructores y métodos**	**1**
Clase CtaCheques **Constructores y métodos**	**1**
Clase CtaCrédito **Constructores y métodos**	**1**
Clase Fecha **Constructores y métodos**	**1**
Interfaz Impuestos **Declaración, implementación (por otra clase), uso.**	**1**

Total	**10**

Fecha de entrega:

Del lunes 18 al viernes 22 de 2020. Se debe agendar día y hora de entrega, de forma virtual (Google Meet).