



HABILITACIÓN – PROGRAMACIÓN BÁSICA

VARIANTE 1

En el área metropolitana se está consolidando una nueva entidad bancaria; los corporativos de esta entidad son conscientes que, para el éxito de este negocio, es importante estar a la altura de la competencia, sin embargo, para lograr este cometido, se construirá la aplicación que permite integrar los distintos servicios en varias fases; la primera fase permitirá a los clientes del banco realizar las siguientes acciones:

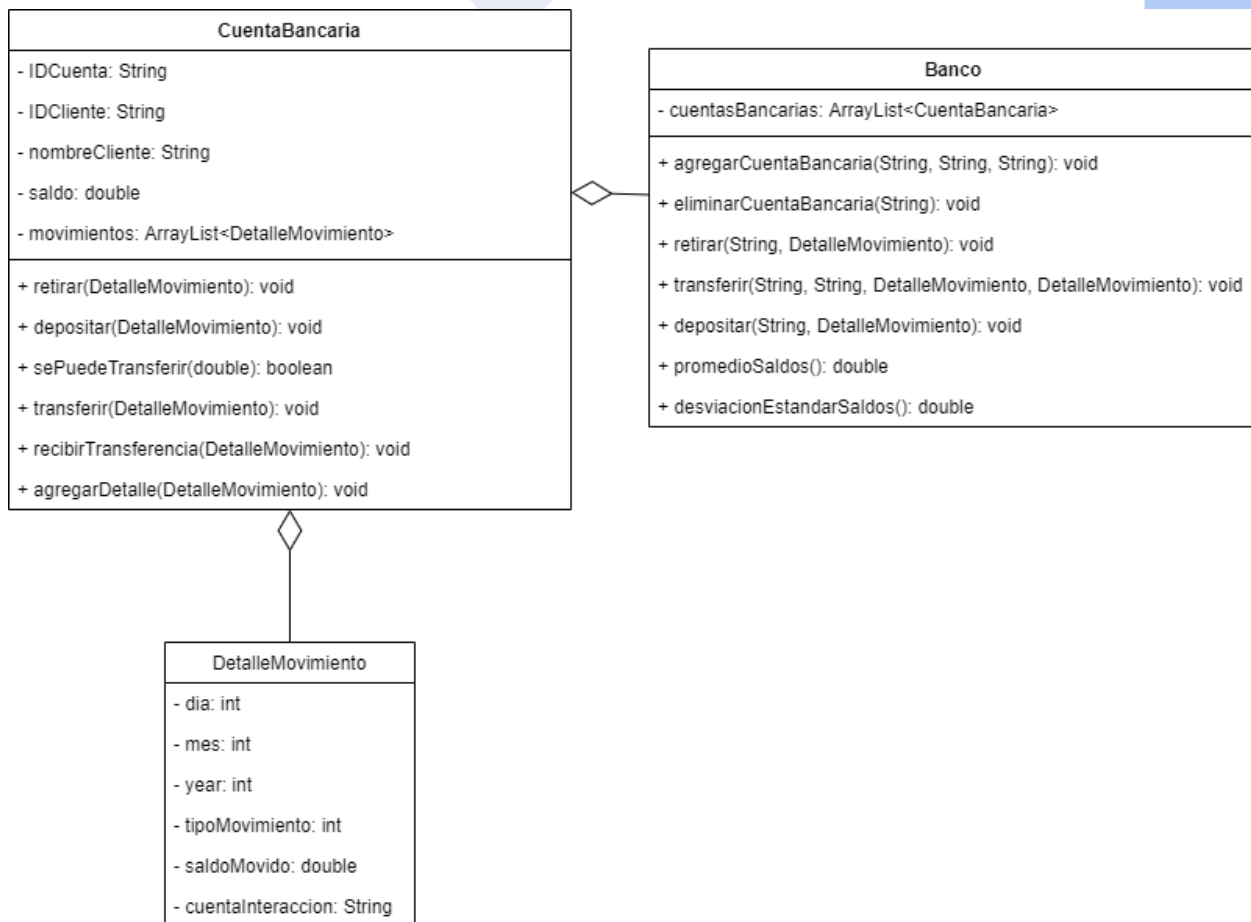
1. Transacciones bancarias entre cuentas del mismo banco.
2. Retiros (Cuando un cliente saca dinero de su cuenta bancaria en efectivo).
3. Depósitos (Cuando un cliente guarda más dinero en su cuenta bancaria).

Usted ha sido contratado como Java Expert Developer, porque ha logrado demostrar habilidades de desarrollo en este lenguaje de programación y se le ha concedido implementar las clases correspondientes a Banco, CuentaBancaria y DetalleTransferencia.

- Clase Banco: Contiene la información las cuentas bancarias de todos los clientes.
- Clase CuentaBancaria: Contiene la información de una cuenta bancaria de un cliente.
- DetalleTransferencia: Contiene los detalles de una transferencia, es decir, si se hizo un retiro, un depósito o una transferencia.

Para facilitar la implementación de estas tres clases, el departamento de Ingeniería de software le hace entrega del diagrama de clases (Recuerde que los métodos relacionados a los *getters* y *setters*, así como también el método constructor son obviados en el diagrama de clases, pero deberán ser incluidos en el código, estos métodos deberán ser creados con el estándar camel case: Por ejemplo, si el atributo se llama `saldo`, sus métodos correspondientes a *get* y *set* serían `getSaldo` y `setSaldo`).





Además del diagrama, el departamento de Ingeniería le entrega esta documentación para comprender mejor los elementos del diagrama:

Clase DetalleMovimiento

Atributos

NOMBRE	TIPO DATO	CONCEPTO	INICIALIZACIÓN
dia	int	Guarda el día en que se hizo el movimiento, es un valor entre 1 y 31	En el método constructor
mes	int	Guarda el mes en que se hizo el movimiento, es un valor entre 1 y 12.	En el método constructor





year	int	Guarda el año en que se hizo el movimiento, es un valor entre 1900 y 3000.	En el método constructor
tipoMovimiento	int	Estos son los valores que puede tomar: <ul style="list-style-type: none">• 1: Si fue envío de dinero a otra cuenta bancaria• 2: Si fue recepción de dinero desde otra cuenta bancaria<ul style="list-style-type: none">• 3: Si fue retiro• 4: Si fue depósito	En el método constructor
saldoMovido	double	Cantidad de dinero que jugó en el movimiento, es un valor mayor o igual que cero (Si se retiró \$35000, el saldo movido fue la cantidad retirada, si se hizo una transferencia de \$23000, el saldo movido fue la cantidad transferida y si se hizo un depósito de \$670000, el saldo movido fue el que se recibió).	En el método constructor
cuentaInteraccion	String	Guarda el ID de la cuenta con la que se tuvo una transacción (Independientemente si envió o recibió), en caso tal de que el cliente haya hecho un retiro o un depósito, se guardará la cadena de texto "NONE" (Ya que en estos dos casos la cuenta del cliente no interactuó con otra cuenta bancaria)	En el método constructor





Clase CuentaBancaria

Atributos

NOMBRE	TIPO DATO	CONCEPTO	INICIALIZACIÓN
IDCuenta	String	Guarda el número de la cuenta bancaria del cliente	En el método constructor
IDCliente	String	Guarda el número del documento de identidad del cliente	En el método constructor
nombreCliente	String	Guarda el nombre del cliente	En el método constructor
saldo	double	Guarda la cantidad de dinero que tiene el cliente en la cuenta bancaria (Este es un valor mayor o igual que cero).	0
movimiento	ArrayList<DetalleMovimiento>	Guarda los diferentes movimientos realizados por el cliente en su cuenta bancaria	Debe ser un ArrayList vacío



Métodos

NOMBRE	TIPO RETORNO	PARÁMETROS	CONCEPTO
retirar	void	DetalleMovimiento d: Es una instancia de la clase DetalleMovimiento	Descuenta <code>d.getSaldoMovido()</code> a saldo y agrega d al <code>ArrayList</code> movimientos (Siempre y cuando saldo quede mayor o igual que cero después de hacer el descuento, si no, NO SE HACE EL RETIRO Y NO SE AGREGA NINGÚN DETALLE)
depositar	void	DetalleMovimiento d: Es una instancia de la clase DetalleMovimiento	Suma <code>d.getSaldoMovido()</code> a saldo, y agrega el detalle al <code>ArrayList</code> movimientos
sePuedeTransferir	boolean	double d: Cantidad de dinero que se desea transferir	true en caso tal de que $saldo - d \geq 0$ y false en caso de que $saldo - d < 0$
transferir	void	DetalleMovimiento d: Es una instancia de la clase DetalleMovimiento	Si se puede realizar la transferencia descuenta <code>d.getSaldoMovido()</code> a saldo y agrega d al <code>ArrayList</code> movimientos
recibirTransferencia	void	DetalleMovimiento d: Es una instancia de la clase DetalleMovimiento	Si se puedo realizar la transferencia suma <code>d.getSaldoMovido()</code> a saldo y agrega d al <code>ArrayList</code> movimientos
agregarDetalle	void	DetalleMovimiento d: Es una instancia de la clase DetalleMovimiento	Agrega d al <code>ArrayList</code> movimientos





Clase Banco

Atributos

NOMBRE	TIPO DATO	CONCEPTO	INICIALIZACIÓN
cuentasBancarias	ArrayList <CuentaBancaria>	Guarda las cuentas bancarias de todos los clientes	Debe ser un ArrayList vacío





Métodos

NOMBRE	TIPO RETORNO	PARÁMETROS	CONCEPTO
agregarCuentaBancaria	void	<ul style="list-style-type: none">String IDCuenta: Número de cuenta de la nueva cuenta bancariaString IDCliente: Número de documento de identidad del dueño de la nueva cuenta bancariaString nombre: Nombre del dueño de la nueva cuenta bancaria	Agrega una nueva cuenta bancaria al <i>ArrayList</i> cuentasBancarias
eliminarCuentaBancaria	void	<ul style="list-style-type: none">String IDCuenta: Número de la cuenta bancaria a eliminar del <i>ArrayList</i> cuentasBancarias	Elimina la cuenta bancaria con número de cuenta igual al recibido por parámetro (IDCuenta)
retirar	void	<ul style="list-style-type: none">String c: Número de la cuenta de la que se desea realizar el retiroDetalleMovimiento d: Es una instancia de la clase DetalleMovimiento	Busca en el <i>ArrayList</i> cuentasBancarias el objeto cuyo IDCuenta sea igual a c, y le invoca el método retirar, pasándole como parámetro d
transferir	void	<ul style="list-style-type: none">String c1: Número de la cuenta que pasará el dineroString c2: Número de la cuenta que recibirá el dineroDetalleMovimiento d1: Es una instancia de la clase DetalleMovimiento que son los detalles asociados al movimiento realizado por c1.DetalleMovimiento d2: Es una instancia de la clase DetalleMovimiento que	No olvidar verificar si de c1 se puede retirar la cantidad especificada en d1.getSaldo()





		son los detalles asociados al movimiento realizado por <code>c2</code> .	
depositar	void	<ul style="list-style-type: none">• <code>String c</code>: Número de la cuenta en la que se guardará el dinero• <code>DetalleMovimiento d</code>: Es una instancia de la clase <code>DetalleMovimiento</code> que son los detalles asociados al movimiento realizado por <code>c</code>.	Busca en el <code>ArrayList</code> <code>cuentasBancarias</code> el objeto cuyo <code>IDCuenta</code> sea igual a <code>c</code> , y le invoca el método <code>depositar</code> , pasándole como parámetro <code>d</code>
promedioSal dos	double	No recibe	Retorna el promedio de los saldos de todas las cuentas bancarias
desviacion EstandarSa ldos	double	No recibe	Retorna la desviación estándar de los saldos de todas las cuentas bancarias





PRECISIONES

1. No hay métodos estáticos.
2. Deben existir *getters* y *setters* de todos los atributos de cada clase, estos deben ser escritos en la forma estándar camel case.
3. La fórmula de la desviación estándar a usar es: $\sigma = \sqrt{\frac{1}{n} \sum_{i=1}^n (x_i - \mu)^2}$, donde n es el número de muestras a considerar en el cálculo (Número de cuentas bancarias que hay en el ArrayList), x_i es el saldo de cada cuenta bancaria y μ es el promedio de los saldos del conjunto de cuentas bancarias que hay en el ArrayList.
4. El calificador asumirá un error del 1% por aproximaciones que su código pueda hacer en el cálculo del promedio y la desviación estándar.

TAREAS

- En el archivo preconstruido en la plataforma Moodle, implementar las clases especificadas en el diagrama de clases, teniendo en cuenta las precisiones dadas por el equipo de Ingeniería de software.
- Los nombres de los métodos y atributos **DEBEN** ser nombrados tal y como aparecen en el diagrama de clases.
- Usted **NO** debe solicitar datos por teclado, ni programar un método `main`, tampoco use `Java Source Package`, usted está solamente encargado de la construcción de la clase.
- Los parámetros de los métodos constructores de cada atributo deben aparecer en el orden en que fueron enunciados en la tabla de la documentación (En caso de que sea un atributo que se debe inicializar en el método constructor), es decir, el método constructor de la clase `DetalleMovimiento` es:

```
public DetalleMovimiento(int día, int mes, int year, int  
tipoMovimiento, double saldoMovido, String  
cuentaInteraccion)
```





NOTA ACLARATORIA

Usted podrá desarrollar la clase requerida en un IDE como NetBeans, y al final copiar y pegar el código en la herramienta VPL, pero **NO** deberá subir archivos, es decir:

Modo incorrecto:

Esta imagen muestra la interfaz de entrega de una tarea en el sistema VPL. En la parte superior, se ven los logos de 'Misión TIC 2022', 'Ingeniería' y 'UNIVERSIDAD DE ANTIOQUIA'. El título de la tarea es 'SEM-Programacion básica ciclo 2'. A la derecha, un mensaje en rojo dice 'NO SUBIR NINGÚN ARCHIVO'. En el centro, hay un campo de texto con el placeholder 'Entrega'. Una flecha verde apunta a este campo con el texto 'COPIA TU CÓDIGO ACÁ (Como en el Ciclo 1)'. Debajo, hay un botón 'Selecciona un archivo...' y una advertencia 'Tamaño máximo para archivos nuevos: 5MB'. En la parte inferior, hay un área de arrastrar y soltar con el texto 'Puede arrastrar y soltar archivos aquí para añadirlos' y botones 'Enviar' y 'Cancelar'.

Modo correcto:

Esta imagen muestra la interfaz de entrega de una tarea en el sistema VPL, pero en el modo correcto. El título de la tarea es 'SEM-Programacion básica ciclo 2'. En el centro, un mensaje en verde dice 'LUGAR CORRECTO'. Debajo, hay un editor de código con el archivo 'Personaje.java' abierto. El código muestra una clase 'Personaje' con comentarios para insertar atributos, método constructor, métodos (NO LOS GETTER Y SETTERS) y setters y getters. A la derecha, hay un botón rojo que dice 'CLIC AQUÍ PARA ACCEDER AL ENUNCIADO'.

