

Tarea 07 – Desarrollo de clases Avanzado

Cuando se crea cada cuenta se almacena en un `ArrayList cuentasClientes` de tipo `CuentaBancaria`, el cual se encarga de instanciarlo la `clase Interface Ioperaciones`.

En el constructor de la clase `CuentaBancaria`, del que heredan los demás tipos de cuenta, se llama al método `almacenarCuenta()` que añade al `ArrayList` la nueva cuenta creada,

Métodos comunes a varias operaciones.

Método `MenuApp.seleccionarCuenta()`

- Se comprueba que hay cuentas introducidas en el sistema: `comprobarExisteCuenta()`
- Se listan todas las cuentas que existen, mediante el método de la `Clase CuentaBancaria` `previewCuentas()`
- Se le pide al usuario que introduzca el ID de la cuenta deseada
- Se llama al método de la `Clase CuentaBancaria` `buscarCuenta()`, que devuelve la posición en el `ArrayList` `cuentasBancarias`, de la cuenta que corresponda con el ID indicado.
- Mediante el método `comprobarIndiceCuenta()` Se asegura que la cuenta existe, si `buscarCuenta()` devuelve -1, la cuenta no existe

Método `MenuApp.introducirImporte()`

- Solicita al usuario que introduzca el importe con el que quiere operar.
- Llama al método `comprobarImporte()` y le pasa por parámetro el importe introducido, para que compruebe que no es un valor negativo

Métodos de clase `CuentaBancaria`.

Los métodos `@Override` se llamara al implementado en una clase u otra dependiendo del tipo de la cuenta que se haya escogido.

- **Método `previewCuenta(): String`**
 - Busca la posición de la cuenta con la que queremos operar.
- **Método `previewCuenta(): String`**
 - Imprime un resumen de los detalles del objeto `CuentaBancaria`
- **Método `toString(): String`**
 - Imprime los detalles del objeto `CuentaBancaria`, dependiendo del tipo de cuenta llamará al método `@Override` del tipo correspondiente
- **Método `detallesCuenta(indice)`**
 - Recibe el índice de la cuenta seleccionada en el método `seleccionarCuenta()` de la `clase MenuApp`
 - Llama al método `toString()` del objeto correspondiente a la posición en el `ArrayList cuentasClientes` pasado mediante el índice devuelto por el método `seleccionarCuenta()`
- **Método `ingresarEfectivo(importe)`**
 - Recibe el importe con el que se quiere realizar la operación
 - Lee el saldo actual de la cuenta y le suma el importe mediante el método `getSaldo()`
 - Actualiza el saldo de la cuenta mediante el método `setSaldo()`
 - Devuelve el nuevo saldo de la cuenta

- **Método retirarEfectivo(importe)**
 - Recibe el importe con el que se quiere realizar la operación
 - Coge el saldo disponible de la cuenta con la que esta trabajando, mediante el **método getSaldo()**
 - Comprueba que el saldo no es inferior al importe que se quiere retirar, mediante el método de la clase **comprobarSaldo()**
 - Si la cuenta seleccionada es de tipo **CuentaBancaria > CuentaCorriente > ctaEmpresa** la comprobación se realiza mediante la implementación de este método en la clase correspondiente
 - Devuelve una Exception si el importe no es correcto para realizar la operación
 - Lee el saldo disponible de la cuenta y le resta el importe
 - Actualiza el saldo de la cuenta mediante el **método setSaldo()**
 - Imprime el nuevo saldo de la cuenta.
- **Método comprobarSaldo(saldo, importe)**
 - Recibe el importe con el que se quiere operar y el saldo actual de la cuenta, y comprueba que el saldo no sea 0 ó que el importe no sea mayor que el saldo disponible
 - Para el tipo de **CuentaBancaria > CuentaCorriente > ctaEmpresa**, además tambien comprueba que no exceda el limite del importe permitido por descubierto.
 - Devuelve false si existe alguna discrepancia o true si esta todo correcto.

Menú de Operaciones

- **1. Operación "Abrir nueva cuenta"**
 - Llama al **método** `appAperturaCuenta()` -lo explico al final de esta sección-
- **2. Operación "Cuentas disponibles"**
 - Llama al **método** `opcion02()` de la **clase** `MenuApp` que hace lo siguiente:
Mediante un for-each va llamando al **método** `listarCuentas` de cada objeto `CuentaBancaria` almacenado en `cuentasClientes`
Dependiendo del tipo de cuenta llamara al método implantado en la clase de la cuenta.
- **3. Operación "Obtener datos de una cuenta"**
 - Se le solicita al usuario que seleccione la cuenta sobre la que quiere realizar la operación: `seleccionarCuenta()`, que nos devuelve la posición de la cuenta, en el `ArrayList`, con la que queremos operar.
 - A través del método `detallesCuenta()` imprime los datos de la cuenta seleccionada.
- **4. Operación "Realizar un ingreso"**
 - Se le solicita al usuario que seleccione la cuenta sobre la que quiere realizar la operación: `seleccionarCuenta()`, que nos devuelve la posición de la cuenta, en el `ArrayList`, con la que queremos operar.
 - Se le pide al usuario que introduzca el importe con el que hay que operar mediante el **método** `introducirImporte()`.
 - Para que realice las operaciones oportunas, llama al **método** `ingresarEfectivo()`, de la **clase** `CuentaBancaria`, correspondiente al tipo de la cuenta seleccionada y le envía el importe introducido.
 - Imprime el nuevo saldo de la cuenta
- **5. Operación "Retirar efectivo"**
 - Se le solicita al usuario que seleccione la cuenta sobre la que quiere realizar la operación: `seleccionarCuenta()`, que nos devuelve la posición de la cuenta, en el `ArrayList`, con la que queremos operar.
 - Se le pide al usuario que introduzca el importe con el que hay que operar mediante el **método** `introducirImporte()`.
 - Para que realice las operaciones oportunas, llama al **método** `retirarEfectivo()`, de la **clase** `CuentaBancaria`, correspondiente al tipo de la cuenta seleccionada y le envía el importe introducido.
- **6. Operación "Consultar Saldo"**
 - Se le solicita al usuario que seleccione la cuenta sobre la que quiere realizar la operación: `seleccionarCuenta()`, que nos devuelve la posición de la cuenta, en el `ArrayList`, con la que queremos operar.
 - Se comprueba que hay cuentas introducidas en el sistema: `comprobarExisteCuenta()`
 - Se listan todas las cuentas que existen, mediante el método de la **Clase** `CuentaBancaria` `previewCuentas()`
 - Se le pide al usuario que introduzca el ID de la cuenta deseada
 - Se llama al método de la **Clase** `CuentaBancaria` `buscarCuenta()`, que devuelve la posición en el `ArrayList` `cuentasBancarias`, de la cuenta que corresponda con el ID indicado.
 - Mediante el método `comprobarIndiceCuenta()` Se asegura que la cuenta existe, si

buscarCuenta() devuelve -1, la cuenta no existe

- Se llama al método **consultarSaldo()** del objeto tipo **CuentaBancaria** seleccionado del ArrayList, mediante los pasos indicados para el método seleccionaCuenta(), que manda a imprimir el saldo disponible.
- **7. Operación "Salir"**
 - Sale del programa

Método **appAperturaCuenta()**

El método lleva al usuario a un nuevo menú en el que se le dá la opción de elegir el tipo de cuenta que quiere abrir, volver al menú anterior o salir de la aplicación.

Dependiendo del tipo de cuenta elegido para crear, el programa instanciará un nuevo objeto de tipo CuentaBancaria pero de una de sus clases heredadas CuentaAhorro o CuentaCorriente, que, de esta última descienden las clases CtaPersonal y CtaEmpresa.

Para esto solicitará al usuario los datos necesarios, según el tipo de cuenta.

En este punto tambien se crean los objetos tipo Persona y tipo CodigoCuenta, mediante los datos introducidos por el usuario.

Los métodos usados para crear la cuenta son:

- **solicitaTitular()**
 - Solicita los atributos "nombre, apellidos y fecha_nac" y construye el objeto Persona titular.
 - Devuelve el objeto titular, necesario para abrir la cuenta
- **solicitaCodigo()**
 - Pide que se introduzca el número de cuenta completo, lo limpia de espacios en blanco y lo manda a validar.
 - Devuelve el objeto codCuentaCliente, necesario para abrir la cuenta

