Informe de consideraciones sobre tarea PROG06.

La aplicación permite al usuario gestionar varios tipos de cuentas bancarias, que se guardaran en una clase Banco capaz de gestionar, como máximo, 100 cuentas.

El programa está compuesto por las clases:

· **Principal**: Controla la interacción con el usuario mediante un menú.

· **Banco**: Representa un banco, con sus atributos y métodos.

· **Persona**: Representa al titular de una cuenta, con sus atributos.

· **CuentaBancaria**: Representa un tipo de cuenta genérico.

· **CuentaAhorro**: Representa un tipo de cuenta más específico que CuentaBancaria. Hereda de ésta sus métodos y atributos y especifica nuevos atributos.

· **CuentaCorriente**: Representa un tipo de cuenta más específico que CuentaBancaria. Hereda de ésta sus métodos y atributos y especifica nuevos atributos.

· **CuentaCorrientePersonal**: Representa un tipo de cuenta todavía más específico que CuentaCorriente. Hereda de ésta sus métodos y atributos y especifica nuevos atributos.

· **CuentaCorrienteEmpresa**: Representa un tipo de cuenta todavía más específico que CuentaCorriente. Hereda de ésta sus métodos y atributos y especifica nuevos atributos.

Y la interfaz:

· **Imprimible**: Define un método con el que cada una de las clases que lo implementen podrán mostrar toda su información.

La clase "**Principal**" contiene el método "**main**", que ejecuta el menú de opciones para el usuario utilizando un bucle **while** manejado por la variable booleana "exit" y un **switch** dentro del bucle. La variable se inicializa en false y mientras no cambie a true al seleccionar la opción 7 continuaremos en el programa.

Esta clase muestra el menú, instancia un objeto "banco" y permite **crear** distintos tipos de cuentas asociados a su titular si ingresamos por teclado los datos requeridos correctamente (utiliza ciertas validaciones para comprobar que los datos sean del tipo y formato correctos), llama a los métodos del objeto banco que hayamos creado para **listar** las cuentas creadas, permite la **búsqueda de una cuenta** en concreto, hacer un **ingreso** o **retirar** una cantidad de la cuenta que especifiquemos mediante el IBAN y **obtener el saldo** de una cuenta en concreto.

La interfaz "**Imprimible**" declara el método **devolverInfoString()**, para ser usado por las clases que la implementen. Este método devolverá la información de una cuenta como cadena de caracteres.

La clase "**Banco**" define los atributos y métodos del objeto banco.

· El método **Banco()** es el constructor por defecto, al ser llamado instanciará el ArrayList que contendrá las cuentas.

· El método **abrirCuenta()** recibe un objeto CuentaBancaria y la inserta en el banco, si el banco está lleno o la cuenta ya existe, devuelve false, si se crea correctamente, devuelve true.

· El método **listadoCuentas()** devuelve un ArrayList, donde cada posición será un String que representatra la información de cada una de las cuentas.

Se utiliza un .forEach que recorre el ArrayList de tipo CuentaBancaria, que contiene todas las cuentas del banco.

Para cada cuenta comprueba si es una instancia de CuentaAhorro, CuentaCorrientePersonal o CuentaCorrienteEmpresa (todas pueden guardarse en el ArrayList, ya que por herencia todas son CuentaBancaria, sean del tipo que sean).

Una vez que encuentra la coincidencia, se hace un casting del tipo de dato de la cuenta (que es CuentaAhorro por el ArrayList, al tipo del que es instancia, para poder usar sus métodos correctamente.

Una vez hecho el casting se llama el método .devolverInfoString(), y se guarda el string obtenido en el Arraylist "listado".

· El método **informacionCuenta()** recibe por parámetros un IBAN, si existe devolverá un String con los datos de la cuenta. Sino no existe devolverá null.

Se utiliza un bucle for para iterar el ArrayList que contiene todas las cuentas hasta encontrar el IBAN coincidente, y una vez encontrado funciona igual que en el caso anterior.

Para cada cuenta comprueba si es una instancia de CuentaAhorro, CuentaCorrientePersonal o CuentaCorrienteEmpresa (todas pueden guardarse en el ArrayList, ya que por herencia todas son CuentaBancaria, sean del tipo que sean).

Una vez que encuentra la coincidencia, se hace un casting del tipo de dato de la cuenta (que es CuentaAhorro por el ArrayList, al tipo del que es instancia, para poder usar sus métodos correctamente.

Una vez hecho el casting se llama el método .devolverInfoString(), y obtiene el string con los datos de la cuenta.

· El método **ingresoCuenta()** recibe por parámetros un IBAN y una cantidad a ingresar, si existe, ingresará la cantidad en la cuenta y devolverá true, sino devolverá false.

· El método **retiradaCuenta()** recibe por parámetros un IBAN y una cantidad a retirar, si la cuenta existe y el importe a retirar es menor o igual al importe que tenga la cuenta, retirará la cantidad en la cuenta y devolverá true, sino devolverá false.

· El método **obtenerSaldo()** recibe un IBAN por parámetro y, si existe, devuelve el un String con el saldo de la cuenta, si no existe devuelve -1.

La clase "**Persona**" define los atributos y métodos del objeto persona, que contendrá la información del titular de una cuenta.

· Contendrá los datos del titular y además implementará la interfaz Imprimible, por lo que especificará el método **devolverInfoString()**, para que al ser llamado devuelva la información del titular.

La clase abstracta "**CuentaBancaria**" define los a tributos y métodos del objeto cuentaBancaria, y además implementa la interfaz Imprimible, de forma que todos estos atributos y métodos (tanto de la clase, como de la interfaz) estarán disponibles para todas sus subclases.

· El método **CuentaBancaria()**, es el constructor, recibe un objeto del tipo persona con la información del titular, seguido del IBAN y el saldo para poder inicializar una cuenta.

· El método **getIban()**, permite consultar el IBAN de la cuenta.

La clase "**CuentaAhorro**" hereda de "**CuentaBancaria**", por lo que comparte los métodos y atributos de ésta última. Además de añadir sus atributos y métodos propios.

· El método **CuentaAhorro()**, es el constructor, igual que en CuentaBancaria, debe recibir un objeto del tipo persona con la información del titular, seguido del IBAN y el saldo (y pasárselo al constructor de la clase padre) además del "tipoInteresAnual", propio de la CuentaAhorro, para inicializar la cuenta.

· El método **devolverInfoString()** implementado en la clase padre y heredado a ésta, será redefinido aquí. Devolviendo un string con la información de la cuenta.

La clase "**CuentaCorriente**" hereda de "**CuentaBancaria**", por lo que comparte los métodos y atributos de ésta última. Además de añadir sus atributos y métodos propios.

· El método **CuentaCorriente()**, es el constructor, igual que en CuentaBancaria, debe recibir un objeto del tipo persona con la información del titular, seguido del IBAN y el saldo (y pasárselo al constructor de la clase padre) además de "listaEntidades", propio de la CuentaCorriente, para inicializar la cuenta.

La clase "**CuentaCorrientePersonal**" hereda de "**CuentaCorriente**", por lo que comparte los métodos y atributos de ésta última. Además de añadir sus atributos y métodos propios.

· El método **CuentaCorrientePersonal()**, es el constructor, igual que en CuentaCorriente, debe recibir un objeto del tipo persona con la información del titular, seguido del IBAN y el saldo (y pasárselo al constructor de la clase padre) además de "comisionMantenimiento", propio de la CuentaCorrientePersonal, para inicializar la cuenta.

· El método **devolverInfoString()**, heredado a ésta clase desde la clase CuentaBancaria, será redefinido aquí. Devolviendo un string con la información de la cuenta.

La clase "**CuentaCorrienteEmpresa**" hereda de "**CuentaCorriente**", por lo que comparte los métodos y atributos de ésta última. Además de añadir sus atributos y métodos propios.

· El método **CuentaCorrienteEmpresa()**, es el constructor, igual que en CuentaCorriente, debe recibir un objeto del tipo persona con la información del titular, seguido del IBAN y el saldo (y pasárselo al constructor de la clase padre) además de "tipoInteresDescubierto", "maxDescubierto" y "comisionDescubierto", propio de la CuentaCorrienteEmpresa, para inicializar la cuenta.

· El método **devolverInfoString()**, heredado a ésta clase desde la clase CuentaBancaria, será redefinido aquí. Devolviendo un string con la información de la cuenta.