PROYECTO GESTIONBANCARIA

Tu proyecto debe implementar los siguientes requerimientos:

SALDO: INGRESAR, SACAR, INFO — TOTAL 15 puntos

Realiza las siguientes modificaciones sobre la clase CuentaBancaria:

1. **(3 puntos)** Incluye un **nuevo atributo saldo** para gestionar el dinero que almacena la cuenta. Al crear una cuenta el saldo debe ser 0. Al mostrarlo/recuperarlo deben aparecer solo 2 decimales.
2. **(4 puntos)** Añade método **double ingresar(double)** dinero en la cuenta. Se debe comprobar que la cantidad pasada como parámetro no es negativa e incrementar el atributo saldo. Se debe devolver el nuevo saldo.
3. **(4 puntos)** Añade método **double sacar(double)** dinero de la cuenta. Se debe comprobar que existe el suficiente dinero y que la cantidad pasada como parámetro no es negativa y disminuir el atributo saldo en esa cantidad. Se debe devolver el nuevo saldo.
4. **(4 puntos)** Define un **nuevo método informacionCuenta()** que liste información de la cuenta con el siguiente formato:

   "nº de cuenta – nombre Titular
   Personas autorizadas: [nombre (dni) , nombre (dni) …..] *//Solo debe aparecer si hay personas autorizadas*
   saldo: XX.XXX,00 €"

AUTORIZADOS: EXISTE, AUTORIZAR, DESAUTORIZAR — TOTAL 15 puntos

Realiza las siguientes modificaciones sobre la clase CuentaBancaria:

1. **(5 puntos)** Añade método **Persona existe(String dni)**, que revisa la lista de autorizados buscando una persona con el dni pasado como parámetro y si lo encuentra devuelve esa persona y sino devuelve null. Para saber si dos personas son iguales usa el método igual() de la clase Persona
2. **(5 puntos)** **Mejora el método autorizar** para que no permita almacenar más de una vez la misma persona. Haz ese control usando el método existe() anterior. Debe devolver true si ha autorizado a la persona, sino devolver false.
3. **(5 puntos)** Añade método **boolean desautorizar(String dni)** que debe comprobar que existe ese dni y eliminarlo de la lista de autorizados, en este caso devuelve true, sino devuelve false.

BANCO — TOTAL 40 puntos

(3 puntos) Crea la clase Banco con un atributo para el nombre y dirección del banco, y otro para almacenar todas sus cuentas bancarias (MAP con clave el nº de cuenta y valor el objeto CuentaBancaria). Solamente deberán existir los métodos get/set para los atributos nombre y dirección del banco.

Deberás crear los siguientes métodos que representan a las acciones sobre las cuentas del banco:

1. **(11 puntos) CuentaBancaria crearCuenta(String nif, String nombre)**, crea y almacena una nueva cuenta bancaria con los datos proporcionados. El número de cuenta será asignado consecutivamente empezando en 1000. Regalaremos saldo a nuestros clientes, la cantidad será de forma aleatoria con un máximo de 50€.
2. **(8 puntos) int eliminarCuenta(long numCuenta)**, devuelve 0 si la cuenta ha sido eliminada, -1 si no existe y -2 si tiene saldo.
3. **(8 puntos) Set<CuentaBancaria> listarCuentas(String nif)** genera un conjunto de cuentas bancarias cuyo titular tiene como nif el pasado por parámetro.

PROYECTO GESTIONBANCARIA

4. **(6 puntos) CuentaBancaria localizaCC(long ncuenta),** deberá localizar la cuenta bancaria cuyo nº de cuenta se corresponde con el parámetro ncuenta, sino devolverá null. La búsqueda debe ser eficente, es decir, terminar cuando se haya localizado la cuenta buscada.
5. ****** NUEVO **** (4 puntos) void datosInicio(),** debe crear varias cuentas bancarias con datos inventados para poder hacer pruebas en la aplicación e incluirlas en el atributo cuentas. Utiliza este método desde el main()

MENÚ PRINCIPAL — TOTAL 30 puntos

Crea el **programa Principal** (main) que **gestiona un solo banco y todas sus posibles cuentas**, sus datos y su operativa. Crea un método static para cada una de las opciones del menú, donde solicitarás los datos según la operación solicitada e invocar al método correspondiente de la clase Banco.

OPCIONES DEL MENÚ: Crea un único objeto Banco y pásalo como parámetro a cada uno de esos métodos static:

- **(10 puntos) Crear** cuenta bancaria. Recoge los datos del titular y se los transmite al método crearCuenta() de banco. Informaremos del número de cuenta asignado y del saldo de regalo.
- **(10 puntos) Eliminar** cuenta bancaria. Solicita el número de cuenta a borrar y se lo pasará al eliminarCuenta(). Si no se ha borrado porque tiene saldo, se ofrecerá la posibilidad de transferir ese dinero a otra cuenta y proceder a borrar la cuenta.
- **(4 puntos) Listar** información de las cuentas de un titular. Indicando su saldo total.
- **(6 puntos) Operativa** sobre una ***cuenta bancaria.*** *En primer lugar, solicitaremos el nº de cuenta sobre la que actuaremos, localizaremos esa cuenta mediante localizaCC() y la pasaremos como parámetro a los métodos static. Estas opciones deberán ser otro submenú, donde se mostrará en todo momento el número de cuenta y titular sobre el que se está operando:*
  - Ingresar dinero
  - Sacar dinero
  - Autorizar persona
  - Desautorizar persona
  - Ver información de la cuenta
  - Volver al menú anterior

Debes mantener al usuario informado, en todo momento, de lo que va sucediendo.