IES TORREVIGÍA TORREVIEJA

UNIDAD 5.
INTRODUCCIÓN A LA PROGRAMACIÓN ORIENTADA A OBJETOS

CASO PRÁCTICO

PROGRAMACIÓN

CFGS DAW

Trinitario Gómez López

2022/2023

1. CASO PRÁCTICO: TRABAJADOR

Implementar una **clase Trabajador** con los siguientes atributos y métodos:

- **Atributos privados**:
  - **Nombre** de tipo String.
  - **Edad** de tipo entero.
  - **Categoría** de tipo entero.
  - **Antigüedad** de tipo entero.
- **Métodos**:
  - ***Públicos***
    - **Constructor por defecto** vacío.
    - **Constructores sobrecargados**:
      - Un constructor sobrecargado que se le pasa por parámetro el nombre y la edad. Si la edad es un número menor a 18 años o, el nombre es vacío o null, la aplicación lanzará una excepción. *Será necesario emplear los métodos privados explicados a continuación.*
      - Un constructor sobrecargado que se le pasa por parámetros: el nombre, edad, categoría y antigüedad. Los valores de categoría y antigüedad deberán estar entre 0 y 2, en caso contrario, se lanzará una excepción. *Será necesario emplear los métodos privados explicados a continuación.*
    - **Constructor de copia**.
    - **Setters** y **Getters** para cada unos de los atributos. Se deberá validar los cuatro atributos de la siguiente forma:
      - El nombre nunca podrá ser vacío o null.
      - La edad sebe ser un número mayor o igual a 18.
      - La categoría y la antigüedad sebe ser número entero 0 y 2.

      En caso que no se cumpla alguna condición se lanzarán las respectivas excepciones. *Será necesario emplear los métodos privados explicados a continuación.*
    - Método **CalcularSueldo()**. Devuelve el sueldo del empleado calculado a partir de su antigüedad (Novato = 0, Maduro = 1, Experto = 2) y categoría profesional (Empleado = 0, Encargado = 1, Directivo = 2). La forma de calcular el sueldo del empleado será de acuerdo a la siguiente tabla:

Sueldo base	607 €
Empleado **Encargado** **Directivo**	+15% sueldo base +35% sueldo base +60% sueldo base
Novato **Maduro** **Experto**	+150 € +300 € +600 €

- ***Privados***:
  - Método **ComprobarCategoria(int categoria)**, devolverá un valor booleano indicando si la categoría pasada por parámetro es válida o no.
  - Método **ComprobarAntiguedad(int antigüedad)**, devolverá un valor booleano indicando si la antigüedad pasada por parámetro es válida o no.
  - Método **ComprobarNombre(String nombre),** devolverá un valor booleano indicando si el nombre pasado por parámetro es válido o no.
  - Método **ComprobarEdad(int edad),** devolverá un valor booleano indicando si la edad pasada por parámetro es válido o no.

Probar la clase Trabajador en el Main, creando objetos de tipo trabajador e invocando a sus métodos, así como validando los valores introducidos.

Para gestionar las validaciones desarrolladas referentes a los atributos: nombre, edad, antigüedad y categoría, será necesario lanzar **excepciones** y manejarlas con los bloques **try-catch**.

2. CASO PRÁCTICO: DAWBANK

La empresa *LibreCoders* te ha contratado para desarrollar un software de gestión de una cuenta bancaria para la cooperativa de banca ética y sostenible *DawBank*. Se trata de una aplicación Java formada por una **clase principal DawBank** y otra llamada **CuentaBancaria**.

El programa pedirá los datos necesarios para crear una cuenta bancaria. Si son válidos, creará la cuenta y mostrará el menú principal para permitir actuar sobre la cuenta. Tras cada acción se volverá a mostrar el menú:

1. **Datos de la cuenta**. Muestra la información de la cuenta.
2. **Ingreso**. Pedirá la cantidad a ingresar y realizará el ingreso si es posible.
3. **Retirada**. Pedirá la cantidad a retirar y realizará la retirada si es posible.
4. **Salir**. Finaliza el programa

Clase CuentaBancaria

Una cuenta bancaria tiene como datos asociados el **iban** (international bank acount number, formado por dos letras y 22 números, por ejemplo ES6621000418401234567891), el **titular** (un nombre completo), el **saldo** (dinero en euros).

Cuando se crea una cuenta es obligatorio que tenga un iban y un titular (*que no podrán cambiar nunca*). El saldo inicial será de 0 euros sino se indica una cantidad específica.

El saldo solo puede variar cuando se produce un ingreso (entra dinero en la cuenta) o una retirada (sale dinero de la cuenta). Los ingresos y retiradas solo pueden ser valores inferiores a cero, en caso contrario se lanzarán excepciones.

El saldo de una cuenta nunca podrá ser inferior a -50 euros:

- Si se produce un movimiento que deje la cuenta con un saldo negativo (no inferior a -50) habrá que mostrar el mensaje “AVISO: Saldo negativo”.
- Si se produce una retira que pueda dejar un saldo inferior a -50 euros, se lanzará una excepción y no se producirá la retirada de dinero.

Por el contrario, si se produce un ingreso superior a 3.000 euros se mostrará el mensaje “AVISO: Notificar a hacienda”.

Teniendo en cuenta los aspectos anteriores, el diseño de la clase se ajustará al siguiente modelo:

- Los **atributos** serán privados.
- Los **métodos públicos** a implementar serán los siguientes:
  - **Constructor por defecto,** será privado o no se implementará.
  - **Constructores sobrecargados**:
    - Un constructor sobrecargado que reciba el iban y el titular.
    - Un constructor sobrecargado que reciba todos los atributos de clase.

    Se debe validar que tanto el **nombre** como el **iban** no sean valores nulos ni cadenas vacías, y el **saldo** sera un valor mayor o igual a cero, en caso contrario, se lanzarán las excepciones oportunas.
  - **Constructor de copia.**
  - Métodos **Getters** necesarios.
  - Método **Ingresar.** Ingresará el dinero al saldo de la cuenta. El valor debe ser positivo, en caso contrario lanzará una excepción.
  - Método **Retirar**. Restará la cantidad pasada por parámetro del saldo de la cuenta.
  - Método **DatosCuenta**. Mostrará toda la información de la cuenta del usuario, en concreto: titular, saldo, código del banco, código de la sucursal, e iniciales del país. Emplea los métodos privados que consideres oportunos.
- Los **método privados** a implementar serán los siguientes:
  - Método **ObtenerPais**. Devuelve las iniciales del país perteneciente de la cuenta.
  - Método **CodigoBanco**. Devuelve el código del banco indicado en el iban.
  - Método **CodigoSurcusal**. Devuelve el código de la sucursal indicado en el iban.
  - Método **ComprobarNombre(String nombre),** producirá una excepción si el nombre pasado por parámetro es válido o no.
  - Método **ComprobarIBAN(String iban),** provocará una excepción si el iban pasado por parámetro es válido o no.

Clase DawBank

Clase principal con función main. Encargada de interactuar con el usuario, mostrar el menú principal, dar feedback y/o mensajes de error, etc. Utilizará la clase CuentaBancaria. Puedes implementar las funciones que consideres oportunas.

NOTA: para el control de errores será necesario emplear **Excepciones** junto a la estructura **try-catch.**

Formato de número IBAN: