ACTIVIDAD 6. Herencia – Gastos Personales

CFGS Técnico Superior en DAX

Módulo 03: Programación

INFORMACIÓN IMPORTANTE

Requisitos que deben cumplirse en vuestros trabajos:

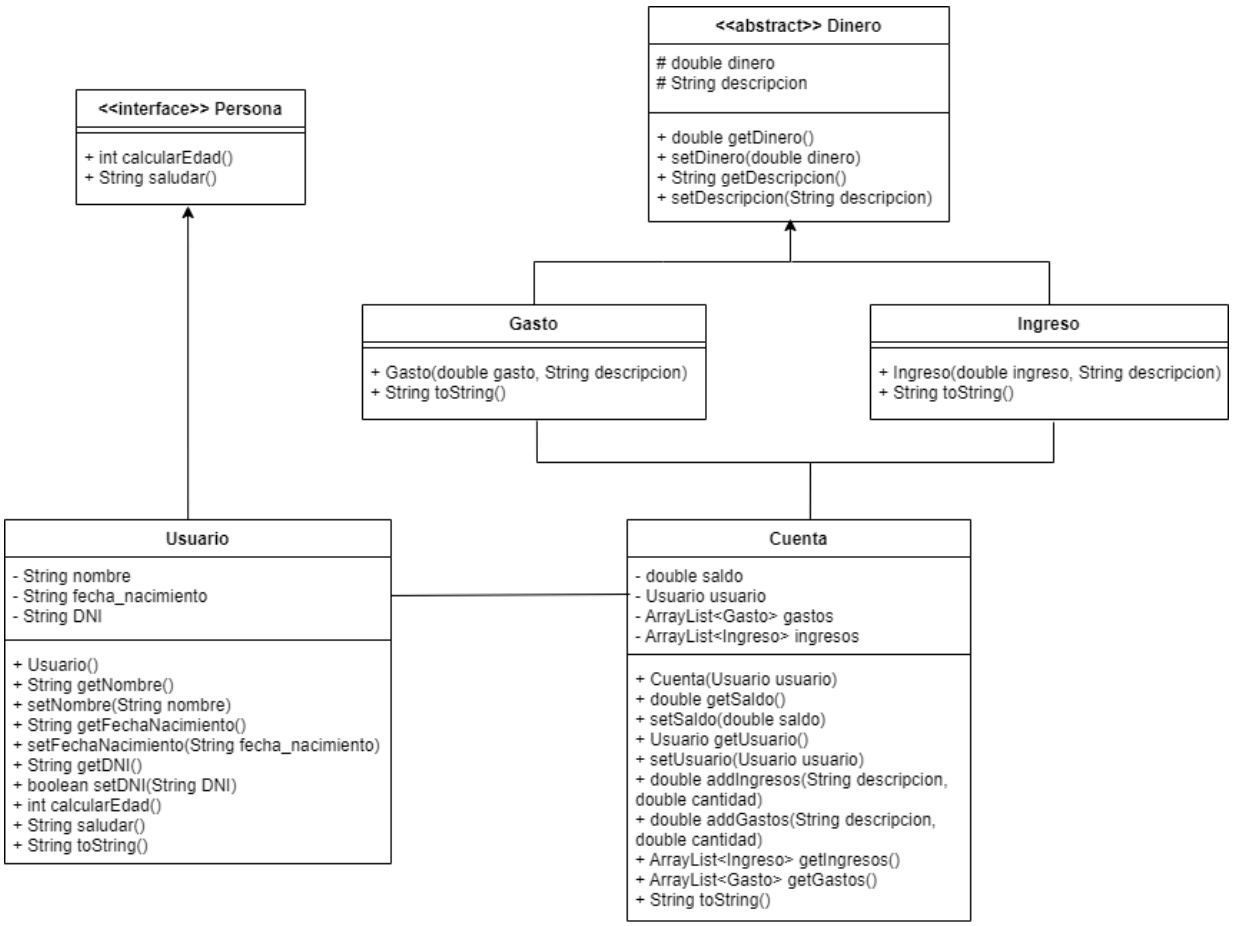
- Las actividades se enviarán únicamente a través de la plataforma dentro de los plazos de entrega establecidos. En caso de no cumplir dichos plazos, NO se podrán enviar de forma posterior.
- Siempre que utilicéis información de Internet para responder / resolver alguna pregunta, tenéis que citar la fuente (la página web) de dónde habéis sacado esta información.
- No se aceptarán copias literales de Internet. Podéis utilizar Internet para localizar información, pero el redactado de las respuestas debe ser de elaboración propia.
- Las respuestas deben estar debidamente argumentadas. No se admiten respuestas escuetas.
- Las actividades deben entregarse siempre en formato ZIP.
- El fichero en formato .zip debe llevar la siguiente nomenclatura: **DAW_PROG_Act6_ApellidosNombre.zip**
- Es responsabilidad del alumno comprobar que el archivo subido en la plataforma es el correcto, ya que en ningún caso el profesor revisará el documento antes del periodo de corrección.
- Si no se entrega una actividad la calificación equivaldrá a un 0.
- Si se detecta que dos alumnos presentan dos actividades iguales la nota se dividirá entre dos, aspirando cada alumno a un 50% de la nota como máximo.

Herencia e Interfaces – Gastos personales

En esta práctica vas a crear una aplicación para gestionar los gastos personales. A través de un menú interactivo por consola, se introducirán ingresos y gastos para así poder llevar un pequeño control de nuestra economía.

El siguiente diagrama de clases representa la estructura que debes realizar:

Diagrama de clases – Gastos personales



Recuerda: es obligatorio que todas las funciones y los métodos tengan el mismo nombre y parámetros que se indican en el diagrama de clases anterior.

Interfaz Persona

Define los métodos **calcularEdad()** y **saludar()**

Clase Usuario

La clase **Usuario** implementa la Interfaz **Persona**.

Esta clase será la encargada de gestionar un único usuario. Éste se creará al inicio del programa leyendo los datos por el teclado.

Ejemplo de datos correctos:

Nombre: Alberto

Fecha nacimiento: 23-12-2009

El DNI deberá tener un formato concreto, está comprobación se realizará en la función setter, la cual devolverá un booleano conforme es correcto o no. Si el DNI es correcto quedará asignado.

Formato correcto:

- Los primeros 8 caracteres solo podrán ser numéricos.
- El último carácter deberá ser una letra entre la A y la Z.
- El guion entre los números y la letra es opcional admitiendo ambas posibilidades.

  DNI: 78844112L

  DNI: 78844112-L

Método **calcularEdad():** a partir de la fecha de nacimiento de tipo String, se deberá descomponer el dato en día, mes y año y calcular la edad de la persona. Devolverá la edad.

Método **saludar():** el programa saludará al usuario, mostrando el siguiente mensaje:

"Bienvenido al programa de gestión de gastos personales *NOMBRE_USUARIO*".

Tendrá una función **toString()** con la que devolver su contenido.

Clases Gasto e Ingreso

Las clases **Gasto** e **Ingreso** heredarán de **Dinero** y tendrán un único constructor en el que se inicializarán los valores recibidos por parámetros.

Además, tendrán una función **toString()** con la que devolver su contenido.

Clase Cuenta

Clase donde se gestionarán todos los movimientos de dinero tanto ingresos como gastos.
Inicialmente (en el constructor) se recibirá el usuario que es dueño de la cuenta y el saldo inicial será de 0€.

Al añadir un nuevo **ingreso**, se sumará al saldo de la cuenta teniendo en esta variable nuestro dinero real, la función devolverá el saldo de la cuenta.

Al añadir un nuevo **gasto** se debe comprobar si se dispone de saldo suficiente, en caso contrario se deberá mostrar un aviso de que no se dispone de saldo para realizar este gasto, pero el programa no debe finalizar.

Si se dispone de saldo suficiente, se restará el importe del gasto y se devolverá el saldo actual de la cuenta.

Las funciones **getGastos()** y **getIngresos()** nos devolverán todos los movimientos de un tipo u otro.

Tendrá una función **toString()** con la que devolverá el usuario y su saldo.

Main

La clase **main** será la que se ejecute al iniciar el programa y seguirá unos pasos definidos:

1. Creación del usuario y sus datos, el DNI no se establecerá hasta que se introduzca uno correcto, el orden de los datos será:
    a. Nombre
    b. Fecha de nacimiento
    c. DNI

Una vez creado el usuario, se mostrará en pantalla el siguiente mensaje, utilizando para ello la función **calcularEdad():**
"Acceso autorizado a la aplicación. Tu edad es *XX*".

2. Creación de la cuenta. Al finalizar la creación de la cuenta, debes llamar a la función **saludar()** del usuario asignado a la cuenta para darle la bienvenida al programa.

3. Visualización del menú con las instrucciones tal y como se muestra en la siguiente figura:

```
Realiza una nueva acción:
1 Introduce un nuevo gasto
2 Introduce un nuevo ingreso
3 Mostrar los gastos
4 Mostrar ingresos
5 Mostrar saldo
0 Salir
```

4. Cada acción realizará una operación donde se deberán solicitar los datos si los necesitase.

5. Al finalizar la aplicación se deberá mostrar el mensaje (**importante que sea igual al que se indica**):

    Fin del programa.
    Gracias por utilizar la aplicación.

RECUERDA: Crea un proyecto nuevo en tu *Workspace* de Eclipse para crear tu nuevo programa. Para hacer la entrega, dale a exportar el proyecto en un archivo **ZIP**.

Comenta el código para que se entienda qué se hace en aquellas partes más complejas. Organiza bien tu código con funciones, reutilizando código, dejando espacios, tabulando bien cada parte, etc.

El archivo que entregues debe tener el siguiente nombre:

DAW_PROG_Act6_ApellidosNombre.zip

Rúbrica de evaluación

Criterios	Calificaciones			Pt
Compilación	0.1 pt El programa compila y no tiene errores		0 pt El programa no compila y no tiene errores	0.1 pt
Estructura y organización	0.2 pt El programa presenta una estructura correcta y se reutiliza código cuando conviene (funciones, espacios, llaves, tabulaciones, …)	0.1 pt El programa presenta una estructura organizada en algunas partes, y algo desordenada en otras. Se reutiliza código y funciones, pero no siempre.	0 pt El programa no está estructurado correctamente, se repite código, las variables no tienen nombres lógicos, …	0.2 pt
Solución	0.6 pt El programa da solución perfectamente a lo pedido en el enunciado	0.3 pt El programa contiene una solución parcial de lo propuesto en el enunciado	0 pt El programa no da una solución correcta	0.6 pt
Comentarios	0.1 pt El código incluye comentarios adecuados al contenido		0 pt El código carece de comentarios	0.1 pt

Puntos totales: 1 pt

¡Buen trabajo!