Boletín POO

3013 - Vector 4D (sobrecarga del operador)

Definir una clase **Vector4D** que representa un vector en 4 dimensiones con números con números como atributos: **u**, **v**, **x** y **y**. La clase Vectro4D debe contener el constructor y los siguientes métodos:

- **__init__(self, …)**: constructor que permite inicializar los atributos.
- **__add__(self, vector)**: nos permite sumar dos vectores.
- **__sub__(self, vector)**: nos permite restar dos vectores.
- **__mul__(self, vector)**: nos permite multiplicar dos vectores.
- **__truediv__(self, escalar)**: nos permite dividir un vector por un escalar.

3015 - Empleado (constructor y destructor)

Definir una clase **Empleado** que contiene dos atributos: nombre y edad. La clase contiene un constructor y los siguientes métodos:

- **__init__(self, …)**: constructor que permite inicializar los atributos de la clase Empleado. Además el constructor debe mostrar una cadena indicando que el nuevo empleado ha sido creado.
- **__del__(self)**: destructor que permite eliminar todas las referencias de un objeto. Además el destructor debe mostrar una cadena indicando que el objeto ha sido eliminado.

3016 - ManipuladoresArchivos (gestión de archivos)

Definir una clase **ManipuladoresArchivos** que contiene dos atributos: nombre del archivo y objeto retornado después de abrir el archivo en modo lectura y escritura.

La clase contiene un constructor, un destructor y el siguiente método:

- **__init__(self, …)**: constructor que permite inicializar los atributos de la clase y mostrar una cadena en consola.

- **__del__(self)**: destructor que permite cerrar el archivo y eliminar las referencias del objeto de la instancia y luego mostrar una cadena en la consola.
- **escribir_archivo(self, frase)**: escribe la frase en el archivo y muestra una cadena en la consola.

3017 - Micadena

Definir la clase **MiCadena** que contiene el atributo cadena. La clase contiene un constructor y los siguientes métodos:

- **__init__(self, variable_str)**: constructor que permite inicializar el atributo de la clase.
- **__add__(self, cadena)**: concatena la cadena, el carácter de separación es '|'
- **__len__(self)**: retorna el número de caracteres de la cadena, sin contar ( )
- **__str__(self)**: retorna la representación del objeto de la clase en forma de cadena
- **__contains__(self, subcadena)**: retorna True si la subcadena está contenida en el objeto de la clase.

3020 - Trabajador (Polimorfismo)

Definir 3 clases:

- **Trabajador**: sus atributos son **nombre**, **salario** y **edad**. Sus métodos son **__init__(self,...)** constructor que inicia los atributos, **mostrar_función(self)** que muestra "Soy un trabajador" y **mostrar_info(self)** que muestra información sobre el trabajador.
- **Director** hereda de Trabajador: sus atributos son **nombre, salario, edad y prima**. Sus métodos son **__init__(self,...)** constructor que inicia los atributos, **mostrar_función(self)** que muestra "Soy director de la empresa" y **mostrar_info(self)** que muestra información sobre el Director.
- **Ingeniero** hereda de Trabajador: sus atributos son **nombre, salario, edad y especialidad**. Sus métodos son **__init__(self,...)** constructor que inicia los atributos, **mostrar_función(self)** que muestra "Soy un ingeniero" y **mostrar_info(self)** que muestra información sobre el ingeniero.

Boletín POO

3021 - Calculadora (métodos estáticos)

Definir una clase llamada **Calculadora** que contiene un constructor y los siguientes métodos estáticos:

- **__init__(self, variable)**: constructor que permite inicializar el atributo **variable**.
- **suma(x,y)**: este método estático permite sumar los valores x e y pasados como parámetros.
- **multiplicacion(x,y)**: este método estático permite multiplicar los valores x e y.
- **división(x,y)**: este método estático permite dividir.

3022 - Ordenador (métodos estáticos, herencia, polimorfismo)

Definir una clase llamada **PC** que contiene un atributos **marca, modelo y precio**, y con los métodos **__init__(self, …)** y **listar_marcas()** que es un método estático que muestra un listado de las principales marcas tales como "HP", "Dell", "Lenovo" y "Apple"
Definir también las clases que heredan de Ordenador:

1. **Desktop**, que hereda de **PC** y tiene los atributos **marca, modelo, precio y tamaño** de la UCP (en cms) y con los métodos **__init__(self, …)** y **mostrar_info()** que muestra información del objeto.
2. **Laptop**, que hereda de **PC** y tiene los atributos **marca, modelo, precio y tamañoPantalla** (en pulgadas) y con los métodos **__init__(self, …)** y **mostrar_info()** que muestra información del objeto.

3023 - Usuario (métodos de clase)

Definir una clase llamada **Usuario** que tiene una **variable de clase** que es en **números de usuarios activos** y tres **variables de instancia** que son el **nombre**, el **apellido** y la **edad** del usuario. Tenemos que implementar también los siguientes métodos:

- **__init__(self, …)**: constructor que permite iniciar las variables de instancia con valores pasados como parámetros.

- **extraer_info(cls, cadena)**: este método de clase toma como parámetros una clase **cls** y una cadena en el formato **“nombre,apellido,edad”**, y crea una instancia a partir de esa cadena.
- **Mostrar_usuarios_activos(cls)**: este método de clase toma como parámetros una clase **cls** y retorna una cadena que indica el número de usuarios activos.
- **__del__(self)**: borra el usuario y decrementa el contador de usuario activos, además muestra información del usuario que se desconectó.

3024 - CuentaBancaria (métodos de clase)

Definir una clase llamada **CuentaBancaria**. Esta clase tiene 2 variables de clase que son el **tipo** de la cuenta (cc,c ahorros, etc) y la **tasa de interés** aplicada. Además tiene 4 atributos: **nombre**, **apellidos**, **numéro de cuenta** y **saldo**.

Métodos a implementar:

- **__init__(self, …)**: constructor que permite iniciar los atributos de la clase.
- **modificar_tasa_interes(cls, nueva_tasa)**: método de clase que permite cambiar la tasa de interé s.
- **modificar_tipo(cls, nuevo_tipo)**: método de clase que permite cambiar el tipo de cuenta.
- **ingreso(self,cantidad)**: añade cantidad al saldo de la cuenta.
- **retiro(self,cantidad)**: resta cantidad al saldo de la cuenta.
- **aplica_tasa_interes(self)**: calcula el interés que nos cobra el banco y se lo resta al saldo.
- **__str__(self)**: permite personalizar la visualización del objeto de la clase.

3026 - GestorArchivos (método estático y herencia)

Definir 2 clases:

1. **Gestorarchivos**: es para manejar archivos en un directorio específico. Tiene un atributo que es la **ruta del directorio**.

2. **Gestorarchivosaudio**: es para manejar espcificamente los archivos de audio.

La clase **GestorArchivos** implementa los siguientes métodos:

- **__init__(self, …)**: inicia atributo.
- **listar(self)**: muestra una lista con los nombre de los archivos del directorio.
- **crear(self, archivo)**: añade archivo a la lista de archivos del directorio.
- **eliminar(self, archivo)**: borra archivo de la lista de archivos del directorio.
- **renombrar(self, nombre_viejo, nombre_nuevo)**: renombra archivo de la lista de archivos del directorio.
- **extension(archivo)**: retorna la extensión del archivo.

La clase **GestorArchivosAudio** implementa los siguientes métodos:

- **__init__(self, …)**: inicia atributo.
- **listar(self)**: muestra una lista con los nombre de los archivos de audio del directorio. Distinguiremos los **mp3**, **wav** o **flac**.