Prueba Escrita – PE (Supuesto Práctico)

UD05 - PROGRAMACIÓN ORIENTADA A OBJETOS EN PYTHON

Ejercicio 1: Gestión de Personas (2,5P)

Contexto del ejercicio

Vas a desarrollar una pequeña aplicación en Python utilizando **Programación Orientada a Objetos**. El programa debe permitir gestionar una persona mediante un **menú principal**.

Puntuación del ejercicio: 2,5 puntos

Enunciado

Crea una clase llamada **`Persona`** que permita almacenar y gestionar información básica.

El programa debe mostrar el siguiente **menú principal**:

1. Crear persona
2. Mostrar datos de la persona
3. Mostrar edad en años futuros (suma una cantidad de años a la edad de la persona
4. Salir

Requisitos técnicos

La clase `Persona` debe tener:

- Atributos:
  - `nombre`
  - `edad`
- Métodos:
  - `mostrar_datos()` → devuelve una cadena con el nombre y la edad
  - `calcular_edad_futura(anios)` → devuelve la edad dentro de X años
  - generar métodos setters y getters

Debes completar los métodos indicados. No cambies los nombres.

APARTADOS EVALUABLES

- **A)** Definición correcta de la clase y atributos (**0,8 puntos**)
- **B)** Implementación correcta de los métodos (**1,0 punto**)
- **C)** Funcionamiento del menú e interacción , simulación (**0,7 puntos**)

Ejercicio 2: Cuenta Bancaria (Encapsulación) (2,5P)

Contexto del ejercicio

En este ejercicio trabajarás el concepto de **encapsulación** mediante una clase que gestione una cuenta bancaria. Deberás completar los métodos indicados y utilizar un **menú principal** para interactuar con el objeto.

Puntuación máxima del ejercicio: 2,5 punto

Enunciado

Crea una clase llamada **`CuentaBancaria`** que permita gestionar el saldo de una cuenta.

El programa debe mostrar el siguiente **menú principal**:

1. Mostrar saldo
2. Ingresar dinero
3. Retirar dinero
4. Mostrar historial (se muestra el historial de operaciones en cuenta)
5. Salir

Requisitos técnicos

La clase **`CuentaBancaria`** debe tener:

- Atributo **privado**:
  - `saldo (lista que se inserta una colección`
  - nombre_cuenta (nombre de la cuenta)
- Atributo
  - historial (es una lista donde se insertan por cada ingreso o retirada una colección con los datos siguientes):
    - tipo: "ingreso" o "retirada"
    - cantidad: cantidad (la cantidad a ingresar o retirar)
    - saldo_actual: saldo (el saldo actual)
- Métodos:
  - `mostrar_saldo()` → devuelve el saldo actual
  - `ingresar(cantidad)` → suma la cantidad al saldo
  - `retirar(cantidad)` → resta la cantidad si hay saldo suficiente
  - mostrar_info() -> muestra información de la cuenta (nombre, saldo, movimientos, etc. Todo bien formateado)
  - mostrar_historial() -> mostramos el historial de operaciones de la cuenta
  - Generar métodos setters y getters

Atención: Controlar saldos negativos o extraer dinero de cuentas sin saldo.

Debes completar los métodos indicados respetando los nombres.

APARTADOS EVALUABLES

- **A)** Uso correcto de encapsulación (atributos públicos y privados) e implementación de las clases correspondientes (**1,0 punto**)
- **B)** Implementación de métodos de gestión del saldo descritos (**0,8 puntos**)
- **C)** Funcionamiento del menú , interacción y simulación (**0,7 puntos**)

Ejercicio 3: Animales (Herencia y Polimorfismo) (2,5P)

VERSIÓN ALUMNO

Contexto del ejercicio

En este ejercicio trabajarás los conceptos de **herencia** y **polimorfismo**. A partir de una clase base, crearás clases hijas que redefinen un mismo método. La aplicación deberá funcionar mediante un **menú principal**.

Puntuación máxima del ejercicio: 2,5 puntos

Enunciado

Crea una clase base llamada **`Animal`** y dos clases hijas llamadas **`Perro`** y **`Gato`**.

El programa debe mostrar el siguiente **menú principal**:

1. Crear perro
2. Crear gato
3. Hacer hablar al animal
4. Salir

Requisitos técnicos

- Atributo
  - nombre (atributo privado)
  - edad (atributo privado)
- La clase `Animal` debe tener un método:
  - `hablar()`
  - show_me() (muestra información del animal)
  - generar los getters y setters
- Las clases `Perro` y `Gato` deben **heredar de `Animal`** y **redefinir el método `hablar()`** con un comportamiento diferente.

Debes completar los métodos indicados y respetar los nombres de las clases y métodos.

APARTADOS EVALUABLES

- **A)** Uso correcto de herencia (0,8 puntos)
- **B)** Implementación del polimorfismo (1,0 punto)
- **C)** Funcionamiento del menú , interacción y simulación (0,7 puntos)

Ejercicio 4: Biblioteca (Métodos especiales) – (2,5P)

Contexto del ejercicio

En este ejercicio trabajarás con **métodos especiales de Python**, que permiten personalizar el comportamiento de los objetos cuando se usan funciones internas del lenguaje. Deberás implementar estos métodos dentro de una clase y utilizar un **menú principal** para interactuar con los objetos.

Puntuación máxima del ejercicio: 2,5 puntos

Enunciado

Crea una clase llamada **`Libro`** que represente un libro de una biblioteca.

El programa debe mostrar el siguiente **menú principal**:

1. Crear libro
2. Mostrar libro
3. Mostrar número de páginas
4. Salir

Requisitos técnicos

La clase **`Libro`** debe tener:

- Atributos:
  - `titulo`
  - `paginas (atributo privado)`
- Métodos especiales:
  - `__str__()` → devuelve el título del libro
  - `__len__()` → devuelve el número de páginas
  - Generar los setters y getters

Debes completar los métodos indicados respetando los nombres.

APARTADOS EVALUABLES

- **A)** Implementación de métodos especiales (**1,0 punto**)
- **B)** Gestión correcta del objeto y clase (**0,8 puntos**)
- **C)** Funcionamiento del menú, interacción y simulación (**0,7 puntos**)

INSTRUCCIONES GENERALES PARA EL ALUMNADO

- El examen consta de **4 ejercicios prácticos independientes**.
- Todos los ejercicios deben resolverse utilizando **Programación Orientada a Objetos**.
- **NO se pueden cambiar los nombres** de clases ni métodos indicados.
- Cada ejercicio incluye una **plantilla de código** que debes completar.
- Todos los ejercicios deben incluir un **menú principal funcional**.
- Se valorará el uso correcto de clases, métodos, encapsulación, herencia, polimorfismo y métodos especiales.

Entrega **un archivo ejercicioX.py** con cada uno de los ejercicios correctamente identificados.

Contenidos evaluados

- Ejercicio 1 → Clases, atributos y métodos
- Ejercicio 2 → Encapsulación
- Ejercicio 3 → Herencia y polimorfismo
- Ejercicio 4 → Métodos especiales (`__str__`, `__len__`)

Nota: (para todos los ejercicios)

Responder exactamente a lo que se pregunta y como se pregunta.
No usar IA. No usar Internet.
Ejercicio obligatorio e individual de cada alumno.
No se puede usar ningún material no suministrado por el profesor.
El código debe de estar libre de errores.
Realizar comentarios y documentar tu solución.
Formatear el código para lectura clara e indentación correcta que facilite la lectura del código.
Generar y justificar tu solución aportada.
Prohibido el uso del móvil en el aula y durante el examen.
Permanecer en silencio durante el examen (no se habla con el compañero).
Entregar dicho ejercicio en la plataforma siguiendo instrucciones del profesor, tiene hora de entrega (si se entrega fuera de hora se considera no entregado).
Leer atentamente todas las preguntas.