EXAMEN UNIDAD 3 y 4 – Clases, objetos, atributos, métodos, herencia, polimorfismo y especialización (Python)

Profesor:	Emmanuel Gómez	Alumno:	
Materia:	Programación Orientada a Objetos	Fecha:	
Grupo:	538	Periodo:	2025-2

INSTRUCCIONES GENERALES

Este examen tiene como objetivo que pongas en práctica los conceptos de **clases, atributos, métodos, objetos, herencia, especialización, agregación, sobrecarga de atributos/metodos y polimorfismo** en Python creando una **Pokédex** simple.
La Pokédex te permitirá almacenar información y funcionalidades de diferentes Pokémon, así como realizar acciones como entrenamiento, combate, y evolución.

- Sigue cada paso cuidadosamente y asegúrate de probar el código en cada sección para verificar que todo funcione correctamente. Quizá no son fanáticos o no conocen del tema, pero para ello deben de investigar. La justificación para este tipo de proyecto está en que este videojuego ya existente desde sus inicios aplica todo y más de lo que estamos viendo en este semestre.

PROYECTO: CREACIÓN DE UNA POKÉDEX

Contexto
Crearás una estructura de clases en Python que simula una Pokédex, es decir, un registro donde podrás visualizar y gestionar Pokémon con sus habilidades. Al final del examen, deberás tener un Pokémon que puedas entrenar, evolucionar, y usar en un combate.

1. CLASE BASE POKEMON

Crea la clase principal (clase padre) Pokemon: Esta será la clase base de todos los Pokémon, y tendrá los siguientes atributos con valores por defecto:

- nombre (str): El nombre del Pokémon, valor predeterminado "Sin Pokémon".
- descripcion (str): Breve descripción del Pokémon, valor predeterminado "No descripción".
- ataque (int): Valor de ataque entre 1 y 1000, valor predeterminado 0.
- defensa (int): Valor de defensa entre 1 y 1000, valor predeterminado 0.
- vida (int): Valor de vida entre 1 y 1000, valor predeterminado 0.
- nivel (int): Valor entre 1 y 100 que representa el nivel del Pokémon. → Cada vez que el nivel llega a 100, se reinicia a 0 y aumenta en 1 el atributo evolucion.
- evolucion (int): Número de evoluciones que ha alcanzado el Pokémon (inicia en 1).
- atrapado (bool): Indica si el Pokémon ha sido atrapado (valor predeterminado False).

Nota: Asegúrate de que al llegar el atributo nivel a >= 100, el Pokémon "evoluciona", lo que significa que el atributo evolucion aumenta en +1 y debe de tener como "tope" el nivel máximo de evolución que tiene el pokemon (investigar el pokemon y sus evoluciones).

Ejemplo de evolución: Siempre debe de iniciar el programa eligiendo a un Pokémon en su estado inicial (evolución = 1) y de allí irá aumentando. Supongamos que el programador decide tener como Pokémon inicial al Pokémon llamado "Pichu" y su atributo nivel llega a >=100, su atributo evolución aumenta en +1, éste pasa a evolucion = 2, convirtiéndose en "Pikachu". Si el nivel vuelve a llegar a >=100, evolución pasa a ser evolucion = 3, evolucionando a "Raichu". Recordar que cuando el nivel sea >= 100 el atributo nivel debe reiniciarse de nuevo a cero. Si el Pokémon solo tiene 2 evoluciones que su límite sea el número de evolución y no avance más. Cuando el Pokémon evolucione debe de aparecer bajo su nuevo nombre, lo demás se mantiene igual. Esto quiere decir que en sus detalles como Pokémon tendrá el nombre nuevo de su evolución y cuando se elija la opción hablar Pokémon deberá de hacer su sonido típico que usualmente es su nombre (según la evolución).

2. MÉTODOS DE LA CLASE POKEMON

Crea los siguientes métodos en la clase Pokemon para realizar distintas acciones.

Parte 1: Métodos básicos

- detallesPokemon(): Muestra en pantalla todos los detalles del Pokémon (nombre, descripción, ataque, defensa, vida, nivel, evolución, y si ha sido atrapado).
- hablar(): Imprime un sonido típico del Pokémon.
- entrenar(): Aumenta el ataque, defensa y vida del Pokémon en +10 y su nivel en +20. Si el nivel llega a 100, imprime a manera de aviso: "¡El Pokémon ha evolucionado!" y reinicia el nivel a 0, incrementando la evolución en +1.

Parte 2: Métodos de mejora (selección individual)

- subirAtaque(): Aumenta solamente el atributo ataque usando un boost_ataque de +20.
- subirDefensa(): Aumenta solamente el atributo defensa usando un boost_defensa de +20.
- subirVida(): Aumenta solamente el atributo vida usando un boost_vida de +20.

Nota: Al usar estos métodos en el programa, se deben llamar individualmente, poder seleccionar solo uno a la vez (referencia en la sección de entrenamientos/Entrenar Pokémon).

Parte 3: Método de actualización completa

- actualizar(): Este método actualiza todos los atributos al mismo tiempo (solo atributos ataque, defensa, y vida) sumando el valor del boost_ataque, boost_defensa, y boost_vida al mismo tiempo. A diferencia de los métodos individuales de la Parte 2, aquí se incrementan todos los valores simultáneamente.

3. CREAR SUBCLASES DE POKEMON

Crea cuatro subclases de Pokemon para representar las categorías de los Pokémon (esto es para cuando va a iniciar el juego nos pide escoger 1 de 4 pokemones iniciales). Cada subclase deberá tener un **atributo especial** y su propio método actualizar().

- **Categorías de Pokémon**:
  - Agua
  - Fuego
  - Eléctrico
  - Hierba

3.1. **Atributo Especial**: En cada subclase, añade el atributo especial llamado: **ataque_especial** para definir el movimiento especial del Pokémon (por ejemplo, "Hidrobomba" para un Pokémon de Agua). El ataque especial debe de ser un poco mayor que el ataque normal, por eso se llama especial.

3.2. **Método actualizar() en subclases**: Sobrescribe el método actualizar() para que puedas modificar los atributos específicos del Pokémon de acuerdo a su categoría.

---

4. COMBATE Y ATRAPADO

Ahora que tienes tu Pokémon y sus habilidades, crea las siguientes funcionalidades.

4.1. **Crear un Pokémon Enemigo**: Crea 4 Pokémon enemigo con los valores de los atributos de tu preferencia. La idea es que el Pokémon que elijas deberá enfrentarse a este Pokémon enemigo en un combate. Al entrar en combate, el programa elegirá aleatoriamente a alguno de los 4 Pokémon enemigos (puedes usar la librería random o usarlos mediante una lista).

Posteriormente habrá en el menú una opción para crear nuestros propios enemigos dándole los valores de los atributos que queramos mediante input. Al crear nuestro pokemon enemigo deberá de estar dentro de la lista de los pokemones enemigos previamente ya creados (estos 4 enemigos ya pueden estar creados/instanciados en nuestro programa)

4.2. **Simular (Atención a lo que se pide)**:

- El combate se basa en **RESTAR** el ataque de un Pokémon contra la defensa del otro, una vez que se le termine los puntos de defensa pasará a atacar ahora la vida del oponente reduciéndolo hasta que llegue a cero. No debe de tener valores negativos.
- Al entrar a combatir se deben de mostrar los STATS (detalles de Pokémon) de mi Pokémon y el Pokémon enemigo.
- Dentro del combate deben de estar las siguientes opciones:
    - 1. Pasar Turno, 2. Ataque normal, 3. Ataque especial, 4. Huir
- La manera de combatir será por turnos, nosotros por medio de las opciones anteriores y el Pokémon enemigo debe de escoger de manera aleatoria si quiere: ataque normal, ataque especial o pasar turno.
- Se deberán de hacer los cálculos pertinentes para poder restar el valor del ataque contra la defensa y de allí pasar a restar la vida del Pokémon hasta llegar a cero.

  Ejemplo:
  -Mi Pokémon elegido tiene **ataque:20, defensa:50 y vida:100** **(el valor del ataque es del atributo ataque)**
  -El Pokémon enemigo tiene **ataque:60, defensa 100 y vida 150**
  -Si me ataca el Pokémon enemigo entonces su **ataque:60** se restará contra mi **defensa:50**; pero no dejará mi defensa en defensa: -10, sino que dejará mi defensa en 0 y el restante pasará contra mi vida quedando de la siguiente manera:
  -Mi Pokémon elegido ahora tendrá **ataque:20, defensa:0 y vida:90**
  De esta manera se logran los cálculos pertinentes, así debe de ser todo el combate hasta que alguno de los 2 tenga su defensa y vida en cero (no valores negativos).

- Si la vida del Pokémon enemigo llega a 0, podrás atraparlo (el atributo atrapado se convierte en True).
- Como es un Pokémon atrapado nos debe de aparecer en la lista de la opción: Pokemones atrapados. Ustedes deciden si quieren que sea un Pokémon como opción jugable, esto es opcional.
- Recuerda que por cada movimiento debemos de ver ambos detalles de los pokemones para ver cómo van en el combate; es decir, por cada movimiento realizado nos debe de traer sus stats: ataque, defensa y vida.

4.3. **Método verPokemonsAtrapados()**: Este método muestra una lista de los Pokémon que has atrapado junto con sus detalles.

5. INTERACCIONES CON EL USUARIO

Haz que el programa sea interactivo para que el usuario pueda seleccionar opciones y realizar acciones con el Pokémon. Usa input() para guiar al usuario durante la ejecución del programa y asegúrate de que el programa se mantenga en ejecución hasta que el usuario decida finalizar (que sea elegir las opciones de menú mediante números).

Requerimientos de la Interfaz de Usuario:

- Al iniciar, la Pokédex debe:
    - Dar la bienvenida al usuario (pídele su nombre).
    - Informarle que no tiene Pokémon aún.
    - Darle la opción de elegir un Pokémon entre las subclases que creaste (Agua, Fuego, Eléctrico, Hierba).
- Luego de elegir el Pokémon, se deben mostrar todos sus detalles.
- Después permitir al usuario poder elegir alguna opción del menú principal, como entrenarlo o realizar acciones adicionales.
- Todo lo referente a **entrenamiento** debe de estar en una opción llamada **Entrenar Pokémon.**

Ejemplo: Si el usuario elige la opción **Entrenar Pokémon, deben de estar las siguientes 4 opciones dentro de él:**

1. Entrenamiento Normal: actualiza solo ataque, defensa y nivel (al mismo tiempo).
2. Entrenamiento Individual: actualizar de manera individual ataque, defensa o vida.
3. Entrenamiento Intensivo: actualiza ataque, defensa y vida (al mismo tiempo utilizando boost).
4. Entrenamiento Personalizado: Nos debe de permitir actualizar manualmente todos los valores del Pokémon.

- Al elegir cualquiera de las 4 opciones anteriores nos debe de mostrar sus atributos rápidamente para saber si aumentó.
- Permite en una opción crear un Pokémon enemigo para poder ingresar sus valores y que esté ingresado en la lista para poder combatir contra él (si es elegido de manera aleatoria).

Nota: Todas las interacciones deben realizarse mediante input() y permitir al usuario elegir qué hacer con el Pokémon en cada momento, no se te olvide siempre usar adecuadamente las opciones de cancelar y/o salir.

---

6. EVALUACIÓN Y CRITERIOS

El examen se evaluará en base a los siguientes criterios:

- ¿Cumple todos los requerimientos técnicos del exámen?.
- ¿Todo el programa se ejecuta correctamente en consola y hace lo que se supone que debe de hacer?.
- ¿Se observa herencia, polimorfismo, métodos, sobreescritura, etc?
- Participación equitativa entre los integrantes del equipo (programadores).
- Cada alumno responderá preguntas individuales al final (Preguntas hechas por el maestro).

Recordatorios Finales

- Asegúrate de que el programa es interactivo y permite múltiples interacciones hasta que el usuario quiera terminar/cerrar el programa.
- Prueba cada método y cada interacción para verificar que todo funcione correctamente.
- No es necesario que les guste Pokémon, solo investiguen los nombres, evoluciones, cuantas de ellas tiene, el sonido que emite, un ataque normal y un ataque especial.
- NO COMENTARIOS EN CODIGO (esto es para poder realizar las preguntas en el código el día del exámen).
- Está permitido personalizar de manera visual o gráfica el programa, como poner barras de ataque, iconos, figura del Pokémon hecha con caracteres, poner color, hacer mejoras en la visualización de los detalles del Pokémon, etc.
- Usa estructuras condicionales, ciclos para controlar las acciones o fragmentos del programa.

Visualización del menú principal en la terminal:

1. Detalles de mi Pokémon
2. Hablar Pokémon
3. Entrenamiento (aquí irán los 4 entrenamientos)
4. Combatir
5. Ver Pokémon Atrapados
6. Crear Pokémon Enemigo
7. Salir