EXAMEN UNIDAD 5 y 6 – Proyecto Parte 2: Integración de Manejo de Errores, Archivos en Disco Local y Persistencia (BDD)

Profesor:	Emmanuel Gómez	Alumno:	
Materia:	Programación Orientada a Objetos	Fecha:	17/NOV/25
Grupo:	538	Periodo:	2025-2

INSTRUCCIONES GENERALES

Esta segunda parte del proyecto tiene como objetivo extender tu Pokédex desarrollada en la Parte 1. Incorporarás el manejo de errores y excepciones, además de integrar la funcionalidad de lectura y escritura de archivos (.txt) para registrar las batallas de combate, y persistencia de datos mediante SQLite3 para guardar tu partida y no se pierda nada.

Sigue cuidadosamente cada paso. Se evaluará no solo la correcta implementación técnica, sino también la creatividad y claridad en tu código. Asegúrate de probar cada nueva funcionalidad y las partes relevantes del programa.

- **RECORDATORIO: PRIMERO SE ESTARÁ REVISANDO LOS ERRORES Y OBSERVACIONES HECHAS EN PARTE 1.**

PROYECTO PARTE 2: EXTENDER LA POKÉDEX

➢ A) MANEJO DE ERRORES Y EXCEPCIONES (UNIDAD 5):

Asegúrate de que todo el código desarrollado en la Parte 1 incluye un manejo adecuado de errores y excepciones. Por ejemplo:

1. Validaciones de entrada

- Todo input() debe estar protegido con try / except y cualquier error para que el programa no se detenga. Deberán de ser muy cuidadosos y poner las excepciones necesarias para mantener controlado todo el programa.
- Si el usuario introduce una opción no válida, debe mostrarse un mensaje **sin detener el programa (o en caso de cualquier otro error mostrar un mensaje entendible para el usuario y que no se detenga el programa)**.

2. Bloques try–except

Incluye ejemplos de control de:

- ValueError (por datos numéricos inválidos).
- IndexError (al acceder a listas).
- ZeroDivisionError (puede usarse en una prueba).
- FileNotFoundError (al leer el archivo).
- IOError (errores de lectura/escritura).
- Etc.

3. Nueva opción en el menú principal

Agrega una opción extra en el menú principal llamada **"Pruebas de Manejo de Errores"**, donde el usuario podrá:

- Ejecutar una operación predefinida en la que se genere una excepción controlada.
- Mostrar un mensaje indicando qué tipo de error ocurrió y cómo se manejó.
- (Para la opción extra pasar a la última parte llamada: ESQUEMA DEL MENÚ PRINCIPAL).

➢ B) LECTURA Y ESCRITURA DE ARCHIVOS (UNIDAD 6):

Guardar los Combates:

- Extiende la funcionalidad de combate para registrar automáticamente los resultados de cada pelea en un archivo .txt en el cual deberás de agregarle la fecha y hora de esa batalla. Ejemplo: batalla_17-11-25_15-30.txt
- El archivo .txt de la batalla deberá de tener el siguiente formato:

=== COMBATE ===
Entrenador: (Nombre del entrenador)
Pokémon: Pikachu
Detalles: (solo atributos)
Enemigo: (Nombre del Pokemon enemigo)
Detalles Enemigo: (sus atributos)

== TURNO 1 ==
Pikachu usa ataque normal/especial
Daño de ataque: 70
Stats del Pokemon Enemigo (para saber cuánto le queda de Defensa y Vida)

Pokemon Enemigo usa ataque normal/especial
Pikachu recibe ataque de 50
Stats de mi Pokemon (para saber cuanto nos queda de Defensa y Vida)

== TURNO 2 ==
Etc..

== TURNO 3 ==
Etc..

Resultado: ¡Victoria!, Has derrotado al Pokémon enemigo y lo has atrapado!

Fecha y hora de la batalla: 27-10-2025 16:45

Este archivo deberá de guardarse como: batalla_27-10-25_16-45.txt

Lectura de Registros:

- Añade una opción en el menú principal para leer y mostrar en pantalla el contenido del archivo de esa batalla.txt.
- (Para ello pasar a la última parte llamada: ESQUEMA DEL MENÚ PRINCIPAL).
- Asegúrate de manejar errores como:
  - Archivo no encontrado (FileNotFoundError).
  - Problemas al leer el archivo (IOError).
- Realiza pruebas de cada funcionalidad para asegurarte de que el programa es robusto frente a errores.

➢ C) PERSISTENCIA MEDIANTE BASE DE DATOS (UNIDAD 6):

- Crear una base de datos llamada pokedex.db usando sqlite3.
- Guardar automáticamente los datos del jugador, su Pokémon actual y los Pokémon capturados en la base de datos.
- Permitir cargar una partida guardada al iniciar el programa o crear una nueva partida.

Requerimientos funcionales

1. INICIO DEL PROGRAMA

Al ejecutar el programa:

1. Conectar a la base de datos pokedex.db.
2. Verificar si existe alguna partida guardada (tabla partidas).

3. Si **no hay registros**, mostrar:

```
No hay partidas guardadas.
¿Deseas iniciar una nueva partida? (s/n):
```

Si el usuario elige “s”, solicitar:
   - Nombre del entrenador
   - Pokémon inicial (nombre y tipo)
   - Guardar la nueva partida.

4. Si **sí hay partidas guardadas**, mostrar una lista con el formato:

```
PARTIDAS GUARDADAS:
1. [2025-10-27 15:42] Emmanuel – Pikachu
2. [2025-10-27 15:45] Karen – Bulbasaur
Selecciona una partida para continuar:
```

Al seleccionar una, el programa carga los datos, muestra un mensaje:

```
¡Bienvenido de nuevo Emmanuel!
Tu Pokémon actual: Pikachu
(Mostrar aquí los detalles del Pokémon)
```

y continúa normalmente desde el menú principal.

2. GUARDADO DE LA PARTIDA

- Desde el **menú principal**, agregar una opción para guardar partida. (Para ello pasar a la última parte llamada: ESQUEMA DEL MENÚ PRINCIPAL).

Cuando el jugador elija **Guardar partida**, deberá de preguntar:

```
¿Deseas guardar el progreso actual? (s/n)
```

- Si elige “s”, guardar:
  - Entrenador (nombre)
  - Pokémon actual (nombre, tipo, nivel, vida, experiencia, etc.)
  - Lista de Pokémon capturados
  - Fecha y hora del guardado en la base de datos pokedex.db (Crea todas las tablas necesarias para que esto funcione, No olvides que estás trabajando con clases y objetos. Haz la investigación necesaria para que esto funcione).

Nota: Cada guardado sobrescribe la partida anterior del mismo jugador.

➢ ESQUEMA DEL MENÚ PRINCIPAL

El menú principal del programa debe incluir las siguientes opciones:

1. Mostrar detalles del Pokémon.
2. Hablar Pokémon.
3. Entrenar Pokémon (Aquí encontraremos submenú de los 4 tipos de entrenamiento según la Parte 1)
4. Ir a combate.
5. Crear un Pokémon enemigo.
6. Ver Pokémon atrapados.
7. **Pruebas de Manejo de Errores** (esta es la nueva opción a incluir).
8. **Registros de Batallas** (al seleccionar esta opción nos deberá de abrir el archivo batalla.txt que tengamos según nuestro usuario).
9. **Guardar Partida** (nueva opción para persistencia con SQLite3)
10. Salir.

RECORDATORIOS FINALES

- Este examen evalúa la capacidad de extender tu proyecto de forma funcional, integrando los temas de la Unidad 5 (Errores y Excepciones) y la Unidad 6 (Archivos y Persistencia).
- Entregables:
  - Código fuente actualizado.
  - Observaciones y errores de la Parte 1 ya completas.
  - Manejo de errores y excepciones en el programa (No debe de detenerse el programa en ningún momento)
  - Archivo batalla_formato_fecha_hora.txt generada de combates.
- La creatividad, claridad y manejo adecuado de conceptos serán factores clave para la evaluación aparte de las preguntas realizadas por el maestro.
- Hagan las investigaciones necesarias para terminar su Examen.