Enunciado: Sistema de Gestión Pokémon (Java)

El objetivo de la práctica es desarrollar una aplicación de consola en Java que permita gestionar un registro de personajes (Entrenadores y Líderes) y sus Pokémon, utilizando **Programación Orientada a Objetos (POO)** y **persistencia en ficheros.**

1. Estructura de Datos (Clases)

Deberás implementar el siguiente modelo de clases:

- **Clase Pokémon**: Representa a las criaturas.
  - Atributos: `nombre`, `tipo` (uso de **Enum**) y `pokedex` (identificador vinculado a un fichero de datos).
- **Clase Persona (Abstracta)**: Base para los personajes del mundo.
  - Atributos comunes: `nombre`, `fecha_nac`, `id, Visualizar(), Sacaredad(), agregarPokemon()`.
- **Clase Entrenador (Hereda de Persona)**:
  - Atributos específicos: `equipo` (Lista de Pokémon) y `procedencia, maximo 6 exception`.
- **Clase Líder de Gimnasio (Hereda de Persona)**:
  - Atributos específicos: `tipo` (Enum), `medalla` (nombre de la medalla que entrega) y `equipoLider` (Lista de Pokémon) solo pueden tener pokemon del mismo tipo, maximo 6 exception.

2. Persistencia

El sistema debe ser capaz de volcar y leer la información de objetos en ficheros (recomendado `Personas.txt` y `Pokemon.txt`) para que los datos no se pierdan al cerrar el programa.

3. Menú de Usuario

El programa presentará un menú interactivo con las siguientes funcionalidades:

Gestión de Personas:

1. **Añadir Pokemon Solo fichero**

   Tendrá que pedirle al usuario, el nombre del pokemon y el tipo del pokemon

2. **Visualizar Pokemon solo con fichero**

3. **Añadir Persona Estructura de datos:** Registro de nuevos Entrenadores o Líderes y enlazar con los pokemon.

   Al usuario le pedirás que quiere añadir si un entrenador o un líder de gimnasio

- El entrenador le pedimos que el id sea autoincrementable, el nombre, la fecha nac, Procedencia y el equipo que tiene máximo de 6 pokemon pero puede dejar huecos libres
- Al líder le pedimos que añada el id, el nombre, la fecha nac , tipo , medalla que da y quipo que tiene máximo de 6 pokémon pero puede dejar huecos libres y solo puede ser del tipo del lider.
- no se pueden repetir los pokemons en los equipos

4. **Visualizar Personas Estructura de datos:** Listado completo de todos los registros .
5. **Modificar Personas SUBMENUS:** Editar datos de un registro existente mediante su ID, y visualización esto implica poder eliminar los pokemon del equipo o añadir alguno más (si tiene 6 no puede añadir más) repetir hasta darle a la opción de salir . También se podrá modificar el nombre y la procedencia.
6. **Ordenar Personas:** Submenú de ordenación:
   - 4.1 Por Rol (Entrenador y Líder).
   - 4.2 Por Nombre (Alfabético).
   - 4.3 Por Código de Entrenador.
7. **Eliminar Persona por id.**
8. **Eliminar mas de una Personas por id**
9. **Eliminar Pokemon por código de pokedex**

Assignment: Pokémon Management System (Java)

Objective: Develop a Java console application to manage a registry of characters (**Trainers** and **Gym Leaders**) and their **Pokémon**. The project must utilize Object-Oriented Programming (OOP) principles and file persistence.

1. Data Structure (Classes)

You must implement the following class model:

- **Class `Pokémon`**: Represents the creatures.
  - **Attributes:** `name`, `type` (using an **Enum**), and `pokedex` (an identifier linked to a data file).
- **Class `Persona` (Abstract)**: The base class for characters.

    - **Common Attributes/Methods:** `name`, `birth_date`, `id`, `Visualize()`, `CalculateAge()`, `addPokemon()`.
- **Class `Entrenador` (Inherits from `Persona`)**:
    - **Specific Attributes:** `team` (List of Pokémon) and `origin`.
    - **Constraint:** Maximum of 6 Pokémon; must throw an **exception** if exceeded.
- **Class `Líder de Gimnasio` (Inherits from `Persona`)**:
    - **Specific Attributes:** `type` (Enum), `badge` (name of the badge they award), and `leaderTeam` (List of Pokémon).
    - **Constraints:** Can only have Pokémon of the same type as the Leader; maximum of 6 Pokémon (**exception** handling required).

2. Persistence

The system must be able to save and read object information using files (recommended: `Personas.txt` and `Pokemon.txt`) to ensure data is not lost when the program closes.

3. User Menu

The program will feature an interactive menu with the following functionalities:

Pokémon Management (File-only)

- **Add Pokémon:** Prompt the user for the Pokémon's name and type. Save directly to the file.
- **View Pokémon:** List all Pokémon stored in the file.

Person Management (Data Structure)

- **Add Person:** Register new Trainers or Leaders and link them to Pokémon.
    - **Trainer:** Ask for ID, Name, Birth Date, Origin, and their team (max 6 Pokémon, allows empty slots).
    - **Gym Leader:** Ask for ID, Name, Birth Date, Type, Badge name, and their team (max 6 Pokémon, allows empty slots, **must match the Leader's type**).
    - *Constraint:* Pokémon cannot be duplicated within teams.
- **View People:** Display a complete list of all registered Trainers and Leaders.
- **Modify Person (Submenu):** Edit an existing record by ID.
    - Includes adding or removing Pokémon from their team (respecting the 6-Pokémon limit).
    - Option to modify Name and Origin.
    - Repeat until the user chooses to exit the submenu.
- **Sort People (Submenu):**
    - **4.1** By Role (Trainer vs. Leader).
    - **4.2** By Name (Alphabetical).
    - **4.3** By Trainer ID.
- **Delete Person:** Remove a specific record by ID.

- **Bulk Delete:** Remove multiple people by their IDs.
- **Delete Pokémon:** Remove a Pokémon from the registry using its Pokedex ID.