Reto 5

Técnicas de Programación y Laboratorio

Enunciado

Pokemon es una saga de juegos originaria de la consola portátil GameBoy. Con decenas de juegos, se desenvuelve en el mundo Pokemon, lleno de criaturas del mismo nombre. El presente trabajo consiste en modelar una miuy pequeña parte de ese mundo Pokemon. El contexto del problema y sobre todo en qué consiste la evolución de los mismos se puede encontrar en un sinnúmero de lugares en internet.

Implementar:

1. Una clase abstracta `Pokemon` que cuente con:
    a. Un método abstracto `evolucionar`, que devuelva un objeto de tipo `Pokemon`.
    b. Un método abstracto `gritar`, que devuelva una cadena tipo `String`.
2. Una excepción personalizada llamada `NoEvolucionaException`, y cuyo mensaje diga *"Este pokemon no puede evolucionar más!"*.
3. Una clase `Charmander`, que herede de la clase `Pokemon`, y que cuente con:
    a. Un nombre (`String`).
    b. Un nivel (`byte`).
    c. Un puntaje de salud (`int`)
    d. El método `evolucionar`, que implementará el método abstracto mencionado arriba, y que devolverá un objeto de la clase `Charmeleon`, con los mismos atributos del objeto actual.
    e. El método `gritar`, que implementará el método abstracto mencionado arriba, y que devolverá la cadena de caracteres "*Charmander!*".
4. Una clase `Charmeleon`, que herede de la clase `Pokemon`, y que cuente con:
    a. Un nombre (`String`).
    b. Un nivel (`byte`).
    c. Un puntaje de salud (`int`)
    d. El método `evolucionar`, que implementará el método abstracto mencionado arriba, y que devolverá un objeto de la clase `Charizard`, con los mismos atributos del objeto actual.
    e. El método `gritar`, que implementará el método abstracto mencionado arriba, y que devolverá la cadena de caracteres "*Charmeleon!*".
5. Una clase `Charizard`, que herede de la clase `Pokemon`, y que cuente con:
    a. Un nombre (`String`).
    b. Un nivel (`byte`).
    c. Un puntaje de salud (`int`)
    d. El método `evolucionar`, que implementará el método abstracto mencionado arriba, y que al ser llamado lanzará una excepción `NoEvolucionaException`.
    e. El método `gritar`, que implementará el método abstracto mencionado arriba, y que devolverá la cadena de caracteres "*Charizard!*".

6. Las tres clases anteriores, corresponden a tres tipos de pokemones que evolucionan uno en el otro, en ese orden. Bajo esa lógica, cree también las clases correspondientes a `Pikachu` y aquel en el cual evoluciona, `Raichu` (que no evoluciona más). Ambas heredan de `Pokemon`.
7. Siguiendo la misma lógica de *Charmander*, *Charmeleon* y *Charizard*, cree las clases correspondientes a `Squirtle`, que evoluciona en `Wartortle`, el que a su vez evoluciona en `Blastoise` (que no evoluciona más). Esto, nos da lugar a tres clases que heredan también de `Pokemon`.
8. Una clase `Pokebola`, que deberá tener:
    a. Un atributo que indique el tamaño de la pokebola ("`grande`", "`mediana`" o "`pequeña`").
    b. Un atributo de tipo `Pokemon`. Esto indica que ahí se podrá guardar un objeto de la clase `Pokemon` (y por extensión de cualquiera de sus subclases). Y será el pokemon atrapado dentro de la misma.
9. Un método `void main`, donde se prueben el funcionamiento de las clases anteriores. Especialmente:
    - Creación de objetos (pokemones y pokebolas).
    - Mostrar en pantalla los gritos de los diferentes pokemones.
    - Evolucionarlos.
    - Crear pokebolas y guardar los pokemones en ellas.

Entregable

Entregar:

- Diagrama de clases (usando draw.io).
- Carpeta del proyecto, primordialmente los archivos fuente (.java).
- El código fuente debe incluir:
    - Un método main que permita verificar el funcionamiento correcto de la aplicación.
    - Tanto las clases e interfaces que soporten la funcionalidad deseada como las clases de las diferentes excepciones.