Introducción a Ciencias de la Computación 2021-1
Práctica 7: Herencia.

Pedro Ulises Cervantes González
confundeme@ciencias.unam.mx

Emmanuel Cruz Hernández
emmanuel_cruzh@ciencias.unam.mx

Yessica Janeth Pablo Martínez
yessica_j_pablo@ciencias.unam.mx

América Montserrat García Coronado
ame_coronado@ciencias.unam.mx

Fecha límite de entrega: 14 de enero de 2021.
Hora límite de entrega: 23:59.

1. Objetivo

Explotar las aplicaciones que tiene la herencia en código, tal como reutilizar código de manera inteligente a partir de la extensión de clases.

Conocer el comportamiento que toman los atributos y métodos de una clase padre sobre una clase hija. Además de explotar las cualidades del polimorfismo.

2. Actividad

Considera las clases *Pokemon* y *Entrenador*.

2.1. Actividad 1 *(1 punto)*

Crea las clases *PokemonFuego*, *PokemonAgua*, *PokemonTierra* y *PokemonAire*. Por cada clase creada, añade 2 atributos que describan específicamente a las clases hijas definidas anteriormente, es decir, dos atributos con los que cuente un pokemon de fuego, dos atributos con lo que cuente un pokemon de agua, etc.

2.2. Actividad 2

Implementa los métodos abstractos `ataca(Pokemon)`, `defiende()`, `descansa()` y `evoluciona()` en las clases creadas en la actividad 1. El comportamiento de cada uno de estos métodos varia dependiendo de cada clase.

Método Ataca *(2.5 puntos)*

```java
/**
 * Ataca a un pokemon, restando a la salud la intensidad
 * del ataque. Cada pokemon ataca dependiendo de su tipo.
 * @param atacado el pokemon a atacar.
 * @return true si el pokemon pudo ser atacado, false en otro caso.
 * Un pokemon no puede ser atacado cuando su salud es negativa o es 0.
 */
public abstract boolean ataca(Pokemon atacado);
```

1. El pokemon *atacado* puede o no defenderse del ataque. En caso de que se defienda, este no disminuye su salud.
2. Si el pokemon atacado tiene salud negativa o salud igual a 0, este no puede ser atacado.
3. Imprimir la forma del ataque.
   - Un pokemon de fuego ataca lanzando bolas de fuego.
   - Un pokemon de agua ataca lanzando chorros de agua.
   - Un pokemon de tierra ataca usando la naturaleza.
   - Un pokemon de aire ataca con la expansión de ondas sonoras.
4. La salud que el pokemon disminuirá al pokemon atacado es el resultado del método `ayudaAPokemon()` multiplicado por la intensidad del ataque.

Método Defiende *(1.5 puntos)*

```java
/**
* Permite al pokemon defenderse.
* @return true si el pokemon se defiende con exito, false en otro caso.
*/
public abstract boolean defiende();
```

1. Puedes apoyarte del método `generaAleatorio()`. Si este regresa true el pokemon se defiende. En caso contrario no se defiende.
2. En caso de defenderse, se mostrará un mensaje con la forma de defensa:
   - Un pokemon de fuego se defiende con una barrera de fuego.
   - Un pokemon de agua se defiende con una ola del mar.
   - Un pokemon de tierra se defiende con una barrera de rocas.
   - Un pokemon de aire se defiende con un fuerte viento.

Método Descansa *(1.5 puntos)*

```java
/**
* Permite descansar un pokemon. Esto es recuperar salud.
* NOTA: La recuperación depende del tipo de pokemon.
* @return true si el pokemon puede descansar, false en otro caso.
* Un pokemon no puede aumentar salud si la salud sobrepasa de 100.
*/
public abstract boolean descansa();
```

1. Para determinar la salud que un pokemon aumentará al descansar se considera lo siguiente:
   - Un pokemon de fuego aumenta 10 de salud al descansar.
   - Un pokemon de agua aumenta 8 de salud al descansar.
   - Un pokemon de tierra aumenta 12 de salud al descansar.
   - Un pokemon de aire aumenta 9 de salud al descansar.
2. Si el descanso del pokemon sobrepasa de 100 de salud, entonces este no puede descansar.
3. En caso contrario, el pokemon aumenta de salud dependiendo de su tipo.

Método Evoluciona *(1.5 puntos)*

```java
/**
* Permite evolucionar un pokemon.
* Un pokemon evoluciona dependiendo de su tipo. Cuando un pokemon
* evoluciona aumenta la intensidad de su ataque.
*/
public abstract void evoluciona();
```

1. Cuando un pokemon evoluciona, aumenta la intensidad de su ataque.
    - Un pokemon de fuego aumenta la intensidad de su ataque en un 15 %.
    - Un pokemon de agua aumenta la intensidad de su ataque en un 17 %.
    - Un pokemon de tierra aumenta la intensidad de su ataque en un 19 %.
    - Un pokemon de aire aumenta la intensidad de su ataque en un 14 %.

3. Actividad 3 *(1 punto)*

Crea las clases *Novato*, *Intermedio* y *Experto*. Implementa el método abstracto llamado `ayudaPokemon()` en cada una de las clases creadas.

4. Actividad 4 *(1 punto)*

Crea una clase *Batalla* que tenga dos atributos de tipo *Entrenador*. Crea los métodos que consideres necesarios para simular una batalla pokemon entre los pokemones del primer entrenador contra los pokemones del segundo entrenador.

Crea un método main en la clase *Batalla* que permita al usuario decir el tipo de ambos entrenadores, así como la cantidad de pokemones máximos que puede tener cada uno.

Finalmente, implementa un menú con las siguientes funcionalidades:

- Hacer que un entrenador capture un pokemon. Esto es, crear un nuevo objeto de tipo Pokemon y agregarlo al entrenador con el método `capturaPokemon(Pokemon)`.
- Atacar un pokemon del entrenador rival con un pokemon del otro entrenador.
- Poner a descansar un pokemon de alguno de los dos entrenadores.
- Evolucionar un pokemon de alguno de los dos entrenadores.
- Mostrar los pokemones que tiene un entrenador. Puedes apoyarte del método `muestraPokemones()`.
- Salir del programa.

5. Nota importante

Esta práctica se puede entregar en parejas. El formato de entrega es el siguiente:

- Apellido1Nombre1Apellido2Nombre2
    - src
        - Batalla.java
        - Entrenador.java
        - Experto.java
        - Intermedio.java

    - Pokemon.java
    - PokemonAgua.java
    - PokemonAire.java
    - PokemonFuego.java
    - PokemonTierra.java
    - Novato.java
- Readme.txt

En caso de realizar la práctica en parejas, uno de los integrantes debe enviar un correo a `emmanuel_cruzh@ciencias.unam.mx` a más tardar un día antes de la fecha de entrega de la práctica, mencionando el nombre de ambos integrantes.

El archivo *Readme.txt* debe contener una breve descripción de qué hizo cada integrante y cual fue la forma en que se organizaron para realizar la práctica. **Ambos integrantes deben contribuir en la implementación de la práctica**.

Sólo una persona debe subir y entregar la práctica en su versión final a Classroom, el otro integrante sólo debe marcar la práctica como entregada.

6. Materiales para consultar

1. Ejemplo de herencia: `https://youtu.be/wqoyQ3BxK4A`
2. Ejemplo de herencia II: `https://youtu.be/rEOFpdI3HY0`
3. Polimorfismo: `https://youtu.be/BSw1MLEc4PQ`
4. Herencia con Minecraft: `https://youtu.be/yh8bTKqCOtU`
5. Polimorfismo con Minecraft: `https://youtu.be/bblFTvuk4pY`

7. Reglas Importantes

- **No se recibirán prácticas en las que estén involucrados más de dos integrantes**.
- Cumple con los lineamientos de entrega.
- Todos los archivos deberán contener nombre y número de cuenta.
- Tu código debe estar comentado. Esto abarca clases, atributos, métodos y comentarios extra.
- Para cada clase solicitada, crea un nuevo archivo.
- Utiliza correctamente las convenciones para nombrar variables, constantes, clases y métodos.
- Sólo se permite el uso de las bibliotecas Scanner y Random.
- En caso de no cumplirse alguna de las reglas especificadas, se restará 0.5 puntos en tu calificación obtenida.