Evaluación Continua 3 - Herencia y Polimorfismo

Con la salida de los *remakes* de la cuarta generación de Pokémon[1], los niveles de nostalgia del profesorado de la asignatura están fuera de control. Para mitigarlo, se os pide implementar un simulador de capturas con un pequeño componente gráfico.

Datos

De todos los Pokémon tendremos la siguiente información[2]:

- **id:** Su identificador numérico (entero). Es único y aparece ordenado, pero puede haber discontinuidades.
- **name:** Cadena de caracteres representando su nombre en inglés
- **description:** Cadena de caracteres representando su descripción en inglés
- **height:** Entero correspondiente a su altura en decímetros (10dm = 1m)
- **weight:** Entero correspondiente a su peso en hectogramos (10hg = 1kg)
- **captureRate:** Entero usado en el cálculo de la probabilidad de captura (más fácil cuanto mayor el valor)
- **sprite:** Cadena de caracteres representando un enlace a su imagen
- **types:** Array de una o dos cadenas de caracteres, cada una representando el nombre de uno de sus tipos

Además, sabemos que algunos son más poderosos que el resto. Los llamaremos *legendarios* y también guardaremos su **power**, un entero representando su fuerza.

Finalmente, también sabemos que algunos son extremadamente poco comunes. Los llamaremos *míticos* y nos guardaremos su **rarity**, un entero representando su rareza.

Funcionalidades

Como es habitual, tendremos que ofrecer un pequeño menú en que el usuario escoja la opción a ejecutar y pueda salir del programa.

En primer lugar, hay que poder visualizar gráficamente los Pokémon a partir de su identificador. Tendremos dos opciones para ello, que se basarán en tecnologías distintas: HTML y Swing. Los resultados se pueden ver en la Figura 1 y Figura 2, respectivamente.

[1] Pokémon y los distintos nombres asociados son marcas registradas de Nintendo

[2] Datos obtenidos de la página web PokéAPI

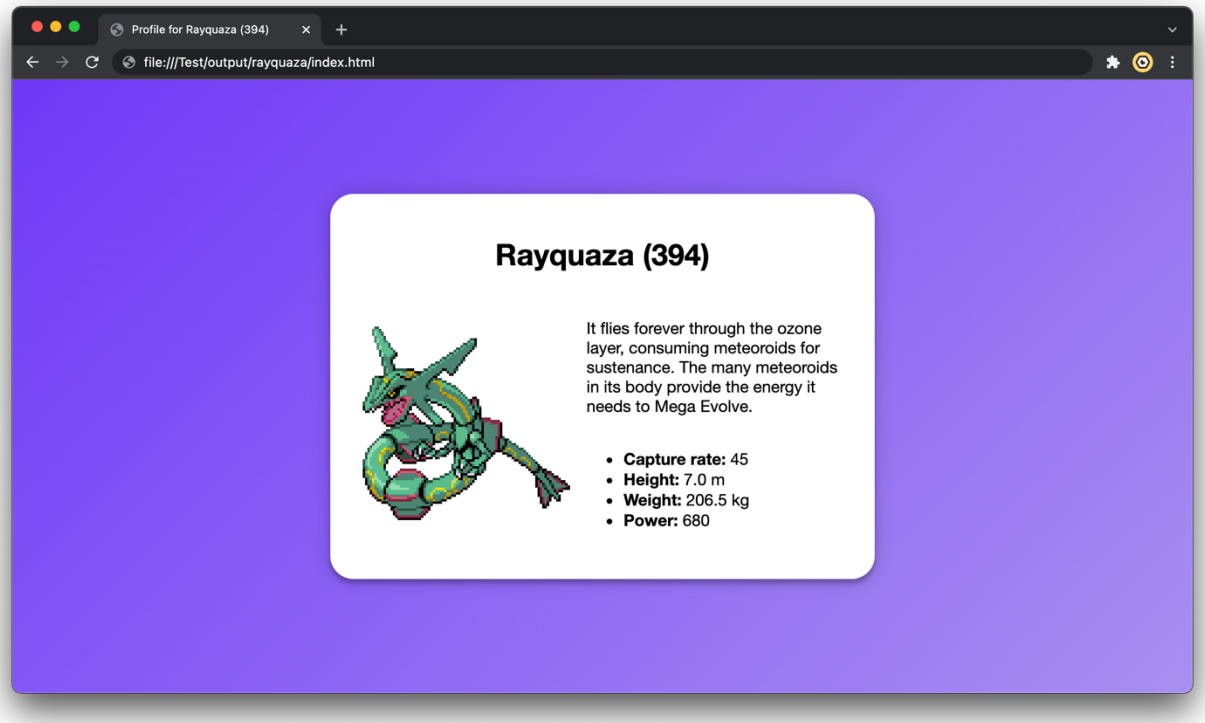

Figura 1. Resultado de visualizar el Pokémon con id 394 en HTML, abierto en el navegador Google Chrome

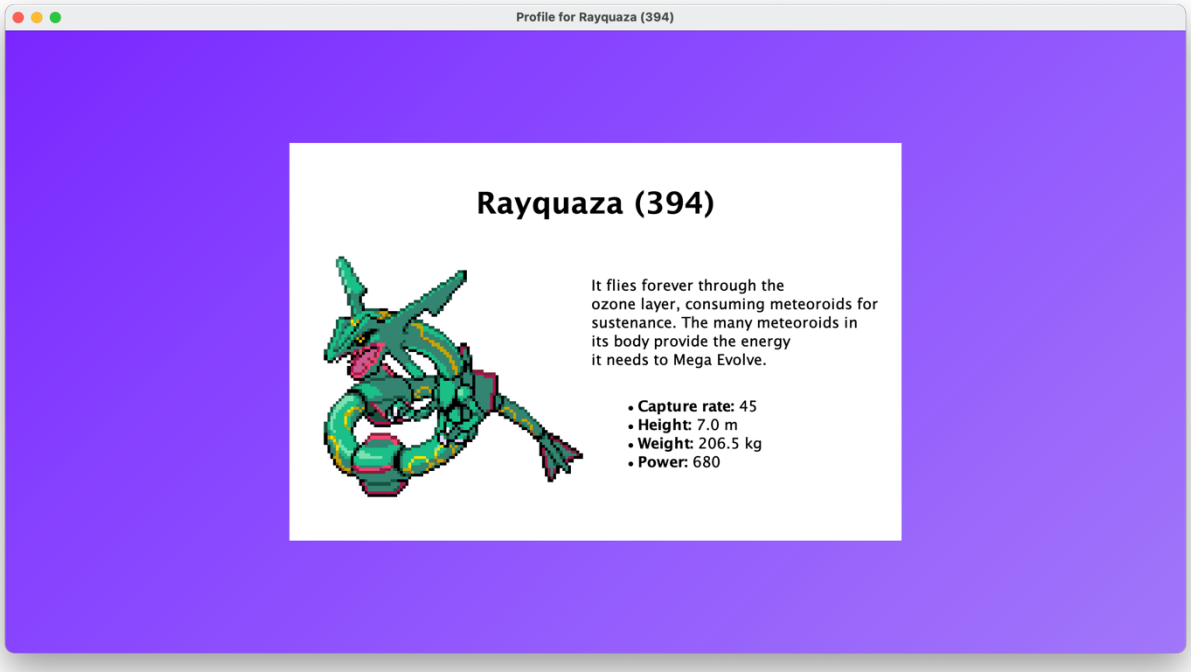

Figura 2. Resultado de visualizar el Pokémon con id 394 en Swing, abierto en una ventana

Nota: Hay que mostrar toda la información disponible (recordad que algunos Pokémon tienen más atributos que otros). Los tipos se representan en forma de degradado en el fondo (será un color sólido si solo hay uno). Tenéis la tabla de colores al final del enunciado.

A continuación, habrá que poder simular un proceso de captura de un Pokémon a partir de su identificador. Al final de cada intento se dará la opción de volverlo a intentar o salir.

Para saber si un Pokémon cualquiera se captura, habrá que generar un entero aleatorio entre 1 y 100, al que llamaremos **p**. La captura será satisfactoria si se cumple que:

$$p \leq \frac{captureRate}{1.5}$$

Si el Pokémon es legendario, la captura será más difícil, y habrá que comprobar que:

$$p \leq \frac{captureRate}{1.5} * \left(1 - \frac{power}{1440}\right)$$

Finalmente, los Pokémon míticos seguirán un proceso de captura diferente, reflejado en la fórmula:

$$p \geq \left(\frac{captureRate}{rarity} \cdot \frac{rarity}{captureRate}\right)^2$$

Recursos

Los datos del programa se proporcionan en datasets en formato JSON, que se pueden leer al iniciar la ejecución o cada vez que sea necesario:

- **mythical.json:** Contiene la información de los Pokémon míticos
- **legendary.json:** Contiene la información de los Pokémon legendarios
- **common.json:** Contiene la información de los demás Pokémon

Para la representación gráfica de los Pokémon **hay que usar** la librería genérica Profile.jar, desarrollada por el profesorado. Disponéis de la documentación relevante en formato Javadoc y diagrama de clases UML.

Requerimientos

La fecha de entrega será el día 8/12/2021 a las 23:55. Se puede entregar hasta 5 días tarde, con una penalización de un punto por día. Hay que entregar un archivo comprimido con:

- Proyecto IntelliJ implementando las funcionalidades pedidas, incluyendo los archivos y librerías que hagan falta, ya configurados (con *paths* relativos siempre que haga falta). Hay que seguir los **patrones GRASP** vistos en clase. El código debe estar comentado en **Javadoc**.
- Archivo StarUML con el diagrama de clases representando el código entregado. Deben aparecer las clases e interfaces de la librería que se hayan usado directamente. Es altamente recomendable diseñar antes de programar.

Disponéis de un ejemplo de ejecución en las páginas siguientes, incluyendo puntos en que se pide entrada de datos al usuario, que **siempre hay que comprobar que sea correcta**, ya sea respecto a los tipos de datos (por ejemplo, que no se introduzca texto si se pide un entero) como en el contexto del ejercicio (por ejemplo, que si se pide un identificador este exista).

A forma de anexo, disponéis de la tabla de colores para tipos al final del documento.

```
--- PokéNostalgia ---

1. Show (Java)
2. Show (HTML)
3. Catch
4. Exit

Enter an option: 1

Which Pokémon? 394

Opening the window showing the Profile for Rayquaza (394)...


--- PokéNostalgia ---

1. Show (Java)
2. Show (HTML)
3. Catch
4. Exit

Enter an option: 2

Which Pokémon? 394

Opening the HTML file showing the Profile for Rayquaza (394)...


--- PokéNostalgia ---

1. Show (Java)
2. Show (HTML)
3. Catch
4. Exit

Enter an option: 3

Which Pokémon? 394

Attempting to catch Rayquaza (394)...

Gah! It was so close, too! Want to try again? [y/n] Y
Aargh! Almost had it! Want to try again? [y/n] y
Aww! It appeared to be caught! Want to try again? [y/n] n

Couldn’t catch Rayquaza (394)...
```

```
--- PokéNostalgia ---

1. Show (Java)
2. Show (HTML)
3. Catch
4. Exit

Enter an option: 3

Which Pokémon? 394

Attempting to catch Rayquaza (394)...

Aww! It appeared to be caught! Want to try again? [y/n] y
Aargh! Almost had it! Want to try again? [y/n] Y

Gotcha! Rayquaza (394) was caught!


--- PokéNostalgia ---

1. Show (Java)
2. Show (HTML)
3. Catch
4. Exit

Enter an option: 4

See you later, Feraligatr!
```

Nota: El ejemplo usa mensajes aleatorios (de un conjunto predefinido) cuando el Pokémon no se captura satisfactoriamente, pero también se aceptarán entregas donde se repita siempre lo mismo.

Anexo: Colores

La siguiente tabla indica los colores a usar per representar cada tipo.

Topo	Color
normal	#A8A878
fighting	#C03028
flying	#A890F0
poison	#A040A0
ground	#E0C068
rock	#B8A038
bug	#A8B820
ghost	#705898
steel	#B8B8D0
fire	#F08030
water	#6890F0
plant	#78C850
electric	#F8D030
psychic	#F85888
ice	#98D8D8
dragon	#7038F8
dark	#705848
fairy	#EE99AC
???	#68A090