

## Programación Avanzada

# Taller IV

## Desarrollo de Software Aplicando Lista con Nexo

### Pokedex Regional

#### Descripción

El profesor Oak está intentando desarrollar un sistema que sea capaz de buscar Pokémons en la región de Kanto, lugar donde se encuentra viviendo actualmente. El nombre pensado para el sistema es de 'Pokedex', y se le ha encargado a los estudiantes de la Universidad Católica del Norte desarrollar la primera versión de esta aplicación.

Para esto, el profesor brinda un archivo de texto llamado 'kanto.txt', el cual posee el siguiente formato:

*ID,Nombre,Etapa,Evolución Siguiente,Evolución Previa,Tipo 1,Tipo 2*

Estas cambiarán de acorde a la 'Etapa' en la que se encuentra el pokémon, y sus evoluciones. Por ejemplo, un pokémon 'Básico' no tendrá 'Evolución Previa'; mismo criterio se aplica a un pokémon que no tenga 'Primera Evolución' ni 'Segunda Evolución'.

1 <sup>er</sup> ejemplo:	115, Kangaskhan, Básico, Normal, Normal
2 <sup>do</sup> ejemplo:	93, Haunter, Primera Evolución, Gengar, Gastly, Fantasma, Veneno
3 <sup>ro</sup> ejemplo:	24, Arbok, Primera Evolución, Ekans, Veneno, Veneno
4 <sup>to</sup> ejemplo:	133, Eevee, Básico, Vaporeon, Jolteon, Flareon, Normal, Normal

- 1<sup>er</sup> ejemplo: Es un pokémon **básico** llamado *Kangaskhan*, donde su id es el número 115, no tiene evoluciones y solo es de **tipo** normal.
- 2<sup>do</sup> ejemplo: Es un pokémon de **primera evolución** llamado Haunter, donde su id es el número 93, su **siguiente evolución** es *Gengar*, su **evolución previa** es *Gastly* y sus **tipos** son fantasma y veneno.
- 3<sup>ro</sup> ejemplo: Es un pokémon de **primera evolución** llamado Arbok, donde su id es el número 24, su **evolución previa** es *Ekans*, no tiene **siguiente evolución** y solo es de **tipo** normal.

- 4<sup>to</sup> ejemplo: Es un pokémon **básico** llamado *Eevee*, donde su id es el número 133, posee 3 **siguientes evoluciones**, siendo estas Vaporeon, Jolteon y Flareon, y solo es de **tipo** normal.

## Sobre los pokémon

- En el mundo pokémon existen 4 etapas: bebé, básico, primera evolución y segunda evolución. Para efectos de este taller, **la etapa bebé no será considerada**.
- Cada pokémon tiene como mínimo 1 tipo y como máximo 2.
- Un pokémon **básico puede o no** tener una **primera evolución**, y en caso de tener, **podría o no tener** una **segunda evolución**. Sin embargo, no se puede tener **segunda evolución** sin tener una **primera evolución**.
- Hay casos en que un pokémon **básico** puede tener más de una **primera evolución** dadas ciertas condiciones, sin embargo, para efectos de este taller, solo existirá uno, y se encuentra en los ejemplos presentados previamente. Lo mismo se aplica para los casos en donde existen más de una **segunda evolución**.
- Los pokémon a considerar serán los primeros 151 de la primera generación, por lo que **todos aquellos por encima del 151 no serán considerados**. Para consultar la lista de pokémons de Kanto, pueden visitar el siguiente link <https://pokedex.net/pokedex/game/red-blue-yellow>.
- Los registros del archivo 'kanto.txt' vendrán desordenados y podrán tener espacios en blanco o saltos de línea entre uno y otro. Además, los campos estarán separados por ',' y podrán tener espacios entre el separador y el contenido como tal.

## Requerimientos

La aplicación debe estar desarrollada con interfaces, listas con nexos (se recomienda doble nexo) y, dentro del código, al menos utilizar una vez una instancia de 'ArrayList' y una instancia de 'LinkedList'. Se recomienda utilizar la clase 'Iterator' para este propósito.

Al iniciar el programa, el sistema debe cargar primeramente el archivo 'kanto.txt' y almacenar la información de todos los pokémons. Hecho esto, el sistema debe permitir al usuario:





- Desplegar los pokémons dado un **rango de números**, ordenados según su id en orden creciente.
- Desplegar todos los pokémons almacenados en el sistema, ordenados alfabéticamente.
- Desplegar los pokémons dado un **tipo** en particular, desplegando aquellos que coincidan con al menos uno de sus dos tipos.
- Desplegar todos los pokémon de **primera evolución**, ordenados según su id en orden decreciente.
- Búsqueda personalizada: el sistema le permitirá al usuario buscar a un pokémon por su nombre o su id asociada y desplegar su información (Id, nombre, etapa, primera evolución, segunda evolución, tipo 1, tipo 2). En caso de tener evoluciones (ya sea **primera evolución** o **segunda evolución**), el sistema debe permitir:
  - Elegir desplegar información sobre una de sus evoluciones (Esto debe ser un proceso repetitivo, es decir, navegar entre las evoluciones libremente hasta que se dicte lo contrario).
  - Salir de la búsqueda al menú principal.

## Entregables

La entrega del taller debe contener:

1. Código completo de la aplicación escrita en Java.
2. Diagrama de clases: en el archivo **clases.puml** ubicado en la raíz del proyecto.

## Condiciones de Entrega

- La fecha de entrega es a más tardar el día **18 de Junio**.
-  **No se aceptarán entregas fuera de plazo.**
- La resolución del taller **debe ser realizada en parejas**, no se aceptarán trabajos personales.
- El IDE utilizado debe ser IntelliJ (Community o Ultimate).
- La entrega debe estar documentada con JavaDOC.
- Se deben manejar excepciones o se realizarán los descuentos respectivos.
- Consultas sobre el taller serán respondida vía foro en Campus Virtual UCN.
- La entrega del taller **será por medio de GitHub** enviando la URL del repositorio privado al ayudante por medio de un correo electrónico:
  -  Prof. Álvaro Castillo: Ayudante Helmer Pizarro ([helmer.pizarro@alumnos.ucn.cl](mailto:helmer.pizarro@alumnos.ucn.cl)).
  -  Prof. Pablo Salas: Ayudante Marcelo Céspedes ([marcelo.cespedes@alumnos.ucn.cl](mailto:marcelo.cespedes@alumnos.ucn.cl)).
  -  Prof. Diego Urrutia: Ayudante Edgardo Ortiz ([edgardo.ortiz@alumnos.ucn.cl](mailto:edgardo.ortiz@alumnos.ucn.cl)).
- **La copia del taller será sancionada con nota 1,0 y los antecedentes del caso serán reportados a la jefatura de carrera y al registro curricular.**

La entrega del taller será evaluada por medio de la aplicación de los criterios de evaluación indicados en la tabla 1.

<b>Criterio</b>	<b>Descripción</b>	<b>Valor</b>
<b>JavaDOC</b>	Se utiliza documentación en estándar JavaDoc en cada clase y métodos del programa.	15%
<b>Comentarios</b>	Se utilizan comentarios de una o más líneas en el código explicando su funcionamiento.	10%
<b>POO</b>	El problema es resuelto por medio de la aplicación del paradigma de la orientación al objeto utilizando: clases, atributos, métodos y herencia.	30%
<b>Codificación</b>	El código de la solución se ajusta a los requisitos planteados en el enunciado y se ajusta completamente a los estándares de codificación de Java.	20%
<b>Ejecución</b>	El proyecto compila correctamente, se encuentra libre de errores de ejecución y responde a los requisitos del problema.	20%
<b>Plataforma</b>	Se utiliza una IDE moderna de desarrollo en la solución.	5%

Tabla 1: Criterios de Evaluación.