UNIVERSIDAD SAN CARLOS DE GUATEMALA
CENTRO UNIVERSITARIO DE OCCIDENTE

INTRODUCCIÓN A LA PROGRAMACIÓN Y LA COMPUTACIÓN
SECCIÓN 1 A

CATEDRÁTICO:
ING. JOSE MOISES GRANADOS GUEVARA

ESTUDIANTE:
JORGE ANIBAL BRAVO RODRÍGUEZ
CARNÉ: 202131782

PRÁCTICA 2
“POKEGOTCHI”
MANUAL TÉCNICO

1. INFORMACIÓN GENERAL DEL DESARROLLO

ENTORNO DE DESARROLLO

Kubuntu: Es una distribución del sistema operativo GNU/Linux derivado del desarrollo de Ubuntu y mantenido por Canonical, la empresa patrocinadora del proyecto. Kubuntu es un sistema operativo libre de código abierto que, a diferencia de su distribución madre Ubuntu (que a la vez deriva de Debian) usa el entorno de escritorio Plasma y su conjunto de aplicaciones de sistema desarrollado por la comunidad KDE, caracterizado por hacer uso de las librerías Qt.

Para este desarrollo se utilizó Kubuntu en su versión 21.10 siendo la última versión estable disponible hasta el momento de la realización de la práctica. A diferencia de mis desarrollos anteriores, este fue realizado en su totalidad desde un sistema Linux, utilizando Windows únicamente para unas cuantas pruebas de compatibilidad.

Windows: Este es un sistema operativo desarrollado por Microsoft, siendo de código cerrado y distribuido bajo una licencia de pago. La versión en la que se hicieron unas cuantas pruebas del programa fue Windows 11.

IntelliJ IDEA: IntelliJ IDEA es un entorno de desarrollo integrado IDE desarrollado por la empresa checa JetBrains. La versión utilizada para el desarrollo fue la 2021.3. La elección de este IDE para el desarrollo se basa en el hecho de que al inicio del curso mi pc no era capaz de ejecutar NetBeans debido a problemas con Java. Este problema fue solucionado pero decidí seguir usando IntelliJ por comodidad.

Java: Java es una plataforma y lenguaje de programación cuyo desarrollo es promocionado por Oracle, aunque es distribuido en paquetes de código abierto denominados “open jdk” (jdk hace referencia a Java development Kit, o Kit de desarrollo Java) bajo la licencia GPL de GNU, lo que impide cerrar el código fuente a quienes realicen cambios en este. Para este desarrollo se utilizó:

- Oracle JDK 15: Debido a que las versiones open-jdk por alguna razón presentan incompatibilidades con mi sistema operativo. Con Oracle JDK en cambio no se presentan problemas de ningún tipo.
- Amazon Coretto 15: Es una distribución de openJDK mantenida por Amazon. Utilicé esta versión ya que IntelliJ permite instalarla con un solo click para usarla con este entorno de desarrollo.
-

Para las interfaces de usuario se usó la biblioteca gráfica para Java denominada Swing.

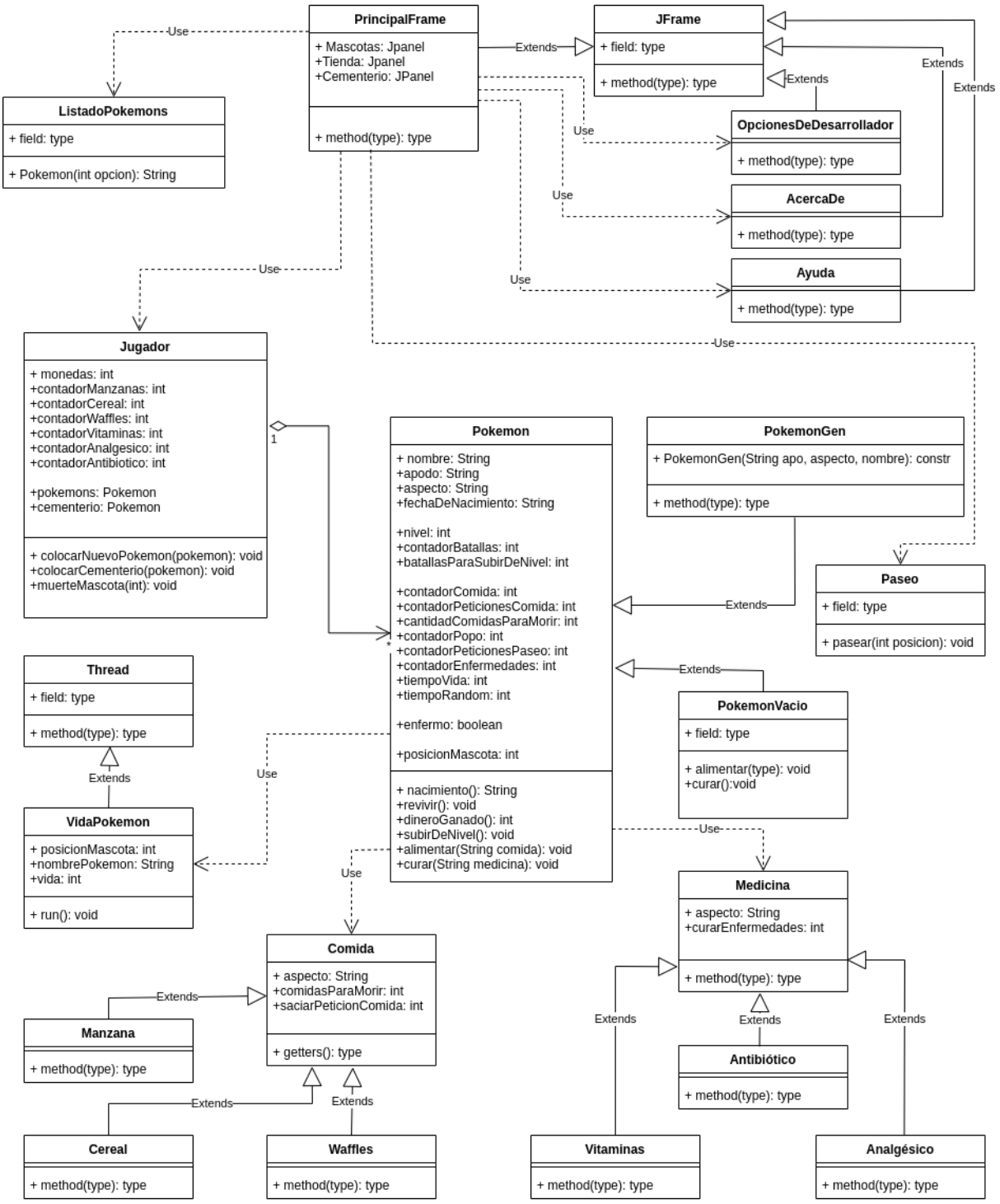
REQUISITOS

Debido a que Java es multiplataforma, este programa extiende su soporte a las plataformas que puedan ejecutar Java 15 o superior, aunque solo se realizaron pruebas en Ubuntu Linux 21.04 y Windows 11.

2. FUNCIONAMIENTO DEL JUEGO

El juego se desarrolló en base a los conceptos adquiridos en clase sobre Programación Orientada a Objetos. El código consta de diferentes clases que interactúan de maneras diferentes, haciendo uso de otros objetos o enredándose. Esto se detalla de mejor manera en el siguiente diagrama de clases: (se adjunta un link al diagrama alojado en drive en caso de que la legibilidad se vea afectada por la compresión del documento).

https://drive.google.com/file/d/1lZ-r_nNRhoxAL-imhv40QUxnmTo8ZmDi/view?usp=sharing



El juego gira en torno al jugador y a los pokemones. Fue planteado de la siguiente manera:
Existe un jugador que posee dos colecciones de mascotas (en dos arrays diferentes), una donde tiene a las mascotas que se encuentran con vida y la otra para las mascotas que fallecieron, cumpliendo un papel de cementerio. La mayoría de acciones que se realizan durante la partida se encuentran repartidas en las propias mascotas, como curarla o alimentarla.

Vida:
Para el ciclo de vida se usan hilos que hacen uso de variables dentro de las mascotas para calcular el tiempo de espera de cada acción (se toma en cuenta una variable para generar tiempos aleatorios y el propio nivel de esta), como pedir comida, pedir que la saquen a pasear y limpiar. Cada vez que se realiza un cambio por el propio ciclo de vida se notifica al usuario por medio de un label del frame principal, al cual simplemente se le coloca el texto asociado a lo que sucede con ella. Los hilos finalizan cuando la mascota muere, aunque se inician de nuevo si esta revive.

Tienda:
La tienda hace uso de una clase externa la cual pide un entero (desde 1 hasta 151 el cual toma de un spinner) y devuelve una cadena de texto con el nombre de la mascota y el path de su aspecto. Al adoptar una nueva mascota se crea una nueva instancia de PokemonGen con las propiedades descritas anteriormente, además de que se calcula su fecha de nacimiento con una función propia de cada mascota. Si una mascota revive calcula de nuevo su fecha de nacimiento.

Alimentación:
Para la aplicación de efectos de las comidas simplemente se reemplaza una variable en la propia mascota que recibe la comida. Esta variable es un tope de peticiones de comida a la cual puede llegar la mascota. Una vez alcanzado este límite entonces muere. Por defecto el límite es de 5. La manzana cambia este límite a 5, el cereal a 7 y los waffles a 10.
Cada alimento tiene un String con el path de su aspecto.
El método de alimentación se encuentra dentro de la propia mascota, sobreescrito en la mascota vacía, advirtiendo que no se puede alimentar un espacio vacío.

Medicina:
La medicina contiene un valor entero que es el que le resta al contador de enfermedades de la mascota. El método curar se encuentra dentro de la propia mascota, sobreescrito en la mascota vacía, advirtiendo que no se puede curar un espacio vacío. El aspecto de cada alimento se encuentra dentro de un String con el path del aspecto de la medicina.

Interfaz de usuario:
Como se describió al principio, para las interfaces gráficas se utilizó la biblioteca swing. La mayoría de los eventos realizados por elementos de la interfaz llaman a métodos dentro de los objetos que forman parte del backend del juego, sirviendo como intermediarios en la comunicación de estos.

Para dibujar los aspectos de medicina, comida y mascotas se toman directamente desde las variables de cada objeto con el path donde se encuentra su aspecto.