PRACTICA 1 - MANUAL TECNICO

202111134 Mario René Mérida Taracena

Descripción del Proyecto

La práctica 1 es una aplicación en Java que simula un torneo de batallas entre personajes. El sistema utiliza programación orientada a objetos (POO) y estructuras de datos dinámicas para gestionar los personajes y simular los combates. Además, genera reportes en formato HTML.

Bases Lógicas de Programación

1. **Programación Orientada a Objetos (POO):**
   - El sistema está diseñado utilizando los principios de POO, con clases como Character, BattleSimulator, CharacterFileReader, ReportGenerator.
2. **Estructuras de Datos:**
   - Se utiliza ArrayList para gestionar la lista de personajes y los sobrevivientes de cada ronda.
3. **Manejo de Archivos:**
   - El sistema lee un archivo de texto (.lfp) utilizando BufferedReader y FileReader.
4. **Generación de Reportes:**
   - Los reportes se generan en formato HTML utilizando FileWriter.

Descripción de los Archivos

Character.java:

- *Ubicación:* src/models/Character.java
- *Descripción:* Representa a un personaje con atributos como nombre, salud, ataque y defensa. Incluye métodos para recibir daño y verificar si está vivo.

CharacterFileReader.java:

- *Ubicación:* src/services/CharacterFileReader.java
- *Descripción:* Lee un archivo .lfp y carga los personajes en una lista de objetos Character.

BattleSimulator.java:

- *Ubicación:* src/services/BattleSimulator.java

- *Descripción:* Simula las batallas entre dos personajes. Calcula el daño y determina el ganador de cada combate.

ReportGenerator.java:

- *Ubicación:* src/services/ReportGenerator.java
- *Descripción:* Genera reportes en formato HTML con los 5 personajes que tienen el mayor ataque y defensa.

Logger.java:

- *Ubicación:* src/utils/Logger.java
- *Descripción:* Proporciona un método estático para mostrar mensajes en la consola.

Main.java:

- *Ubicación:* src/Main.java
- *Descripción:* Es la clase principal del programa. Contiene el menú y la lógica para cargar archivos, simular el torneo y generar reportes.

characters.lfp:

- *Ubicación:* data/characters.lfp
- *Descripción:* Archivo de texto que contiene la información de los personajes.

Reportes HTML:

- *Ubicación:* reports/top_attack.html y reports/top_defense.html
- *Descripción:* Archivos HTML generados por el sistema con los rankings de ataque y defensa.

Estructura del Proyecto

Figura 1. Estructura del proyecto
Fuente: Elaboración Propia (2025)

Librerías Empleadas

1. java.util

La librería java.util es una de las más utilizadas en Java, ya que proporciona una amplia gama de utilidades para trabajar con colecciones, fechas, generación de números aleatorios y más. En el proyecto LFP Battle, esta librería ha sido esencial para el manejo de estructuras de datos dinámicas.

- *ArrayList:* Se utilizó para almacenar y gestionar la lista de personajes que participan en el torneo. ArrayList es una implementación de la interfaz List que permite almacenar elementos de manera dinámica y acceder a ellos de forma eficiente.
- *List:* Esta interfaz fue utilizada para definir la estructura de datos que contiene a los personajes. List permite operaciones como agregar, eliminar y recorrer elementos de manera sencilla.
- *Scanner:* Se empleó para la interacción con el usuario a través de la consola. Scanner permite leer entradas del usuario, como la selección de opciones del menú y la ruta del archivo .lfp.

Relevancia: Sin java.util, el manejo de datos dinámicos y la interacción con el usuario hubieran sido mucho más complicados. Esta librería simplifica el trabajo con colecciones y entradas de usuario.

2. java.io

La librería java.io es fundamental para el manejo de operaciones de entrada y salida (I/O) en Java. En el proyecto LFP Battle, esta librería se utilizó para leer archivos de texto y generar reportes en formato HTML.

- *BufferedReader:* Se empleó para leer el archivo .lfp línea por línea. BufferedReader es eficiente para la lectura de archivos de texto, ya que reduce el número de operaciones de I/O al leer bloques de datos.
- *FileReader:* Esta clase se utilizó en conjunto con BufferedReader para abrir y leer el archivo .lfp. FileReader permite leer caracteres de un archivo de texto.
- *FileWriter:* Se utilizó para escribir los reportes en formato HTML. FileWriter permite escribir caracteres en un archivo, lo que fue esencial para generar los archivos top_attack.html y top_defense.html.

Relevancia: La librería java.io fue crucial para la carga de datos desde el archivo .lfp y la generación de reportes. Sin ella, no habría sido posible interactuar con archivos externos.

3. java.lang

La librería java.lang es una de las más básicas e importantes de Java, ya que proporciona clases fundamentales para el desarrollo de cualquier aplicación. En el proyecto LFP Battle, esta librería se utilizó de manera implícita en casi todas las operaciones.

- *String:* Se utilizó para manejar cadenas de texto, como los nombres de los personajes y los mensajes mostrados en la consola.
- *Integer:* Se empleó para convertir cadenas de texto a valores numéricos al leer el archivo .lfp.
- *System:* Esta clase se utilizó para mostrar mensajes en la consola a través de System.out.println.

Relevancia: Aunque java.lang es una librería básica, su importancia es incuestionable. Proporciona las clases fundamentales que permiten el funcionamiento de cualquier programa en Java.

4. Clases Propias del Proyecto

Además de las librerías estándar de Java, el proyecto LFP Battle utiliza varias clases propias que fueron desarrolladas específicamente para este sistema. Estas clases encapsulan la lógica del negocio y permiten una estructura modular y organizada.

1. *Character:* Representa a un personaje con atributos como nombre, salud, ataque y defensa. Esta clase es el núcleo del sistema, ya que todos los personajes son instancias de Character.
2. *CharacterFileReader:* Se encarga de leer el archivo .lfp y convertir los datos en objetos Character.
3. *BattleSimulator:* Contiene la lógica para simular las batallas entre personajes. Calcula el daño y determina el ganador de cada combate.
4. *ReportGenerator:* Genera los reportes en formato HTML con los rankings de ataque y defensa.
5. *Logger:* Proporciona un método estático para mostrar mensajes en la consola, lo que facilita la depuración y el seguimiento del programa.

Relevancia: Estas clases propias son el corazón del proyecto. Encapsulan la lógica del sistema y permiten una estructura modular y fácil de mantener.

Cómo se Generaron los Reportes

La generación de reportes es una parte fundamental del proyecto LFP Battle. Los reportes se generan en formato HTML y contienen los rankings de los 5 personajes con mayor ataque y defensa. A continuación, se describe el proceso de generación de reportes.

1. Ordenamiento de los Personajes

Para generar los reportes, primero se ordenan los personajes según su ataque y defensa. Esto se hace utilizando el método sort de la clase Collections de Java, junto con una expresión lambda para definir el criterio de ordenamiento.

2. Creación del Contenido HTML

Una vez ordenados los personajes, se crea el contenido HTML utilizando la clase StringBuilder. Este contenido incluye una tabla con los rankings de ataque y defensa.

3. Escritura del Archivo HTML

Finalmente, se escribe el contenido HTML en un archivo utilizando la clase FileWriter.

Compilación y Ejecución

1. Compilación:

Ejecuta el siguiente comando en la terminal:

Figura 2. Compilación de la Practica
Fuente: Elaboración Propia (2025)

2. Ejecución:

Ejecuta el programa con el siguiente comando:

Figura 3. Ejecución de la Practica
Fuente: Elaboración Propia (2025)

Consideraciones Técnicas

1. Manejo de Excepciones:

El sistema maneja excepciones al leer archivos y escribir reportes.

2. Optimización:

El código está optimizado para evitar bucles infinitos en los combates mediante el uso de un daño mínimo de 1.

3. Extensibilidad:

El sistema es fácil de extender. Por ejemplo, se pueden agregar nuevos atributos a los personajes o nuevas funcionalidades al menú.