**📘 MANUAL TÉCNICO**

**Sistema de Gestión de Personajes y Registro de Peleas**  
**Lenguaje:** Java (sin POO, basado en arrays y consola)  
**Desarrollado por:** Jimmy Brian Hurtarte López (un poco de ayuda de un experto, un Monster, un café negro, explicaciones de código de IA)  
**Carnet:** 202303768

**📌 1. ESTRUCTURA DEL PROYECTO**

El sistema está compuesto por **tres archivos .java** principales, todos ubicados en la misma carpeta:

/SistemaPersonajes/

├── Main.java → Contiene el menú principal y flujo del programa.

├── Personaje.java → Manejo de personajes (agregar, modificar, eliminar, listar).

├── Peleas.java → Registro y consulta de peleas.

**🧱 2. DESCRIPCIÓN DE CADA ARCHIVO**

**✅ Main.java**

Contiene:

* Menú interactivo en consola.
* Control de flujo mediante switch-case.
* Llamadas a los métodos de las otras clases.
* Visualización de datos del estudiante.

📌 Entrada del usuario validada para evitar errores.

**✅ Personaje.java**

Maneja los siguientes datos usando **arrays estáticos**:

| **Campo** | **Tipo** | **Detalle** |
| --- | --- | --- |
| ids | int[] | ID único correlativo para cada personaje. |
| nombres | String[] | Nombre del personaje (único). |
| armas | String[] | Arma asignada. |
| habilidades | String[][] | Hasta 5 habilidades por personaje. |
| nivelesPoder | int[] | Nivel de poder (1 a 100). |
| contadorPersonajes | int | Lleva el conteo actual de personajes. |

**Métodos incluidos:**

* agregarPersonaje(Scanner)
* modificarPersonaje(Scanner)
* eliminarPersonaje(Scanner)
* mostrarDatosPersonaje(Scanner)
* listarPersonajes()

**✅ Peleas.java**

Maneja el historial de peleas entre personajes.

| **Campo** | **Tipo** | **Detalle** |
| --- | --- | --- |
| idPersonaje1 | int[] | ID del primer personaje. |
| idPersonaje2 | int[] | ID del segundo personaje. |
| fechaHora | String[] | Fecha y hora de la pelea. |
| contadorPeleas | int | Número total de peleas registradas. |

**Métodos incluidos:**

* realizarPelea(Scanner) → Registra pelea entre dos personajes.
* verHistorial() → Muestra todas las peleas realizadas.

**🛠️ 3. INSTRUCCIONES DE COMPILACIÓN Y EJECUCIÓN**

**⚙️ Requisitos:**

* Java Development Kit (JDK) instalado.
* IDE como **NetBeans**, **Eclipse**, o uso de terminal.

**🔧 Compilación en consola:**

javac Main.java Personaje.java Peleas.java

**▶️ Ejecución:**

java Main

**🧪 4. FUNCIONALIDADES DEL SISTEMA**

| **Opción** | **Funcionalidad** | **Descripción breve** |
| --- | --- | --- |
| 1 | Agregar personaje | Crea un personaje nuevo con validaciones. |
| 2 | Modificar personaje | Permite cambiar arma, habilidades o nivel. |
| 3 | Eliminar personaje | Borra un personaje por ID. |
| 4 | Ver datos de un personaje | Muestra todos los detalles por ID. |
| 5 | Ver listado de personajes | Lista todos los personajes registrados. |
| 6 | Realizar pelea | Registra una pelea entre dos personajes. |
| 7 | Ver historial de peleas | Muestra historial con fecha y hora. |
| 8 | Ver datos del estudiante | Muestra nombre, carnet, curso y sección. |
| 9 | Salir | Termina el programa. |

**📐 5. CONSIDERACIONES TÉCNICAS**

* No se usa **POO (Programación Orientada a Objetos)** como lo exige la práctica.
* Todos los datos se guardan en **arrays estáticos** (vectores).
* Los IDs se asignan **de forma automática y correlativa**.
* La entrada del usuario está protegida contra errores comunes de entrada.
* Se utilizan **validaciones** de nombre único y rango de nivel de poder (1–100).
* El sistema no guarda datos al cerrar (solo se mantiene en ejecución).

**🧑‍💻 7. INFORMACIÓN DEL ESTUDIANTE**

| **Campo** | **Valor** |
| --- | --- |
| Nombre | Jimmy Brian Hurtarte López |
| Carnet | 202303768 |
| Curso | Laboratorio de Introducción a la Programación y Computación 1 |
| Sección | A |
| **📘 MANUAL TÉCNICO DETALLADO**  **Sistema de Gestión de Personajes y Registro de Peleas** **Desarrollado por:** Jimmy Brian Hurtarte López **Carnet:** 202303768  **⚙️ DESCRIPCIÓN GENERAL**  Este sistema, desarrollado en Java sin POO (Programación Orientada a Objetos), utiliza **arrays estáticos, ciclos y sentencias de control** para permitir:   * Gestión de personajes (crear, editar, eliminar, ver). * Registro de peleas entre personajes. * Visualización del historial de peleas. * Interacción mediante un menú en consola.   **🔍 EXPLICACIÓN DE CADA ARCHIVO Y FRAGMENTO**  **1. Main.java – Flujo Principal**  **▶️ Importación de utilidades**  import java.util.Scanner;  Se importa la clase Scanner para permitir la entrada de datos por consola.  **▶️ Método main() - Punto de entrada**  public static void main(String[] args) {  Este método es el punto de inicio del programa. Controla el flujo del menú.  **▶️ Bucle del menú interactivo**  do {  mostrarMenu();  ...  } while (opcion != 9);  Usa un bucle do-while para mostrar el menú continuamente hasta que el usuario elija salir (opción 9).  **▶️ Validación de entrada**  while (!scanner.hasNextInt()) {  ...  }  Evita errores si el usuario ingresa texto en lugar de números para las opciones del menú.  **▶️ Switch para cada opción del menú**  switch (opcion) {  case 1: Personaje.agregarPersonaje(scanner); break;  ...  }  Cada caso llama un método de la clase correspondiente (Personaje o Peleas).  **▶️ Menú visual**  private static void mostrarMenu() {  System.out.println("======= MENÚ PRINCIPAL =======");  ...  }  Imprime las opciones disponibles para el usuario.  **▶️ Datos del estudiante**  private static void mostrarDatosEstudiante() {  ...  }  Muestra información del estudiante desarrollador.  **2. Personaje.java – Gestión de Personajes**  **▶️ Declaración de arrays estáticos**  static final int MAX\_PERSONAJES = 100;  static final int MAX\_HABILIDADES = 5;  static int[] ids = new int[MAX\_PERSONAJES];  static String[] nombres = new String[MAX\_PERSONAJES];  ...  Aquí se definen los vectores para guardar información de hasta 100 personajes.  **▶️ agregarPersonaje(Scanner scanner)**  Este método permite agregar un personaje nuevo:   * Verifica que el nombre sea único. * Valida que el nivel de poder esté entre 1 y 100. * Solicita hasta 5 habilidades.   Fragmento clave:  for (int i = 0; i < contadorPersonajes; i++) {  if (nombres[i].equalsIgnoreCase(nombre)) {  ...  }  }  → Verifica si el nombre ya existe.  **▶️ modificarPersonaje(Scanner scanner)**  Permite al usuario cambiar el **arma, habilidades y nivel de poder** de un personaje existente:  int id = scanner.nextInt();  int index = buscarPersonajePorId(id);  → Busca el personaje por su ID.  **▶️ eliminarPersonaje(Scanner scanner)**  Permite eliminar un personaje:  for (int i = index; i < contadorPersonajes - 1; i++) {  ids[i] = ids[i + 1];  ...  }  → Recorre los arrays para "borrar" al personaje y mantener la continuidad.  **▶️ mostrarDatosPersonaje(Scanner scanner)**  Muestra toda la información (ID, nombre, arma, habilidades, nivel) de un personaje.  **▶️ listarPersonajes()**  Lista todos los personajes registrados, mostrando solo ID, nombre y nivel de poder.  **▶️ buscarPersonajePorId(int id)**  Método interno para buscar un personaje por ID. Devuelve el índice o -1 si no existe.  **3. Peleas.java – Registro de Peleas**  **▶️ Arrays para almacenar peleas**  static int[] idPersonaje1 = new int[100];  static int[] idPersonaje2 = new int[100];  static String[] fechaHora = new String[100];  → Almacenan los IDs de los personajes involucrados y la fecha/hora del enfrentamiento.  **▶️ realizarPelea(Scanner scanner)**  Registra una pelea:   * Pide el ID de dos personajes. * Verifica que ambos existan. * Guarda la pelea con fecha y hora actual.   LocalDateTime ahora = LocalDateTime.now();  → Obtiene fecha y hora exacta de la pelea.  **▶️ verHistorial()**  Muestra el historial completo de peleas registradas:  System.out.println(idPersonaje1[i] + " vs " + idPersonaje2[i] + " - " + fechaHora[i]);  → Lista todas las peleas con los IDs de los participantes y la fecha.  **🧪 FLUJO GENERAL**   1. El usuario abre el sistema (Main) 2. Desde el menú puede realizar todas las acciones. 3. Personaje.java controla los datos de personajes. 4. Peleas.java registra y muestra peleas. 5. Toda la interacción es por consola. 6. No se usa POO, ni archivos externos (todo en memoria con arrays). |  |