# Sistema de Gestión de Personajes

# Práctica 1


**PONDERACIÓN: 5Pts**

**Horas Aproximadas: 36**

**Universidad San Carlos de Guatemala**

**Facultad de ingeniería**

**Ingeniería en Ciencias y Sistemas**

# Índice

**Índice** ..... **1**
**Competencias** ..... **2**
**Objetivos** ..... **2**
General ..... 2
Específicos ..... 2
**Descripción / Enunciado** ..... **3**
Menú de Gestión de Personajes ..... 3
Agregar Personaje ..... 4
Modificar Personaje ..... 4
Eliminar Personaje ..... 5
Ver Datos de un Personaje ..... 5
Ver Listado de Personajes ..... 5
Realizar Pelea entre Personajes ..... 6
Ver Historial de Peleas ..... 6
Ver Datos de Estudiante ..... 7
**Entregables** ..... **7**
**Consideraciones** ..... **8**
**Cronograma** ..... **9**
**Rúbrica de Calificación** ..... **10**
**Valores** ..... **13**
**Referencias** ..... **13**

# Competencias

- Diseña algoritmos aplicando diagramas de flujo y pseudocódigo para resolver problemas básicos de lógica computacional.
- Interpreta código fuente apoyándose en diagramas de flujo y pseudocódigo para detectar errores lógicos y de sintaxis.
- Documenta programas desarrollados usando convenciones de codificación y comentarios estructurados para facilitar su comprensión y mantenimiento.
- Implementa programas simples utilizando lenguaje de programación estructurado (JAVA) conforme a especificaciones dadas.

# Objetivos

## General

Desarrollar un sistema en Java para gestionar personajes y registrar sus peleas, utilizando conceptos fundamentales de programación como vectores, ciclos, sentencias de control y manejo de errores, con el fin de fortalecer las habilidades en programación orientada a la lógica, estructuras de datos y control de flujo.

## Específicos

- Implementar operaciones de gestión de personajes, como agregar, modificar, eliminar y visualizar personajes mediante el uso de vectores y ciclos.
- Registrar y visualizar peleas entre personajes, almacenando la fecha y hora de cada enfrentamiento, aplicando sentencias de control para el manejo de errores y validación de entradas.
- Desarrollar un menú interactivo que permita al usuario interactuar con el sistema de manera eficiente, manejando las operaciones de forma clara y coherente con el uso de estructuras de control y manejo de excepciones.

# Descripción

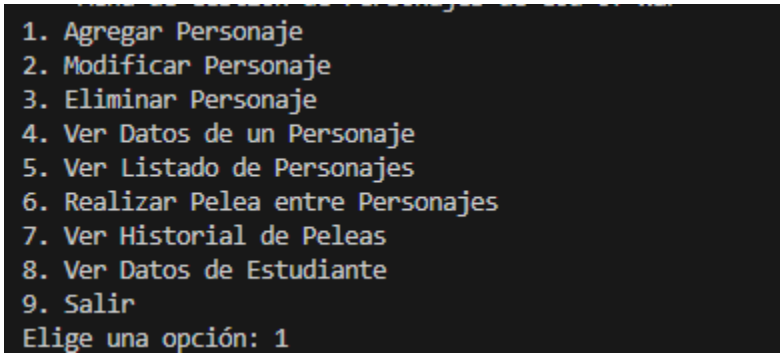
En esta práctica, se desarrollará un sistema de gestión de personajes y registro de peleas utilizando Java. El sistema permitirá gestionar personajes, almacenar datos como nombre, arma, habilidades y nivel de poder, y registrar peleas entre los personajes con la fecha y hora correspondiente. A través de este ejercicio, se pretende que los estudiantes aprendan a implementar operaciones básicas de programación como el uso de vectores, ciclos, sentencias de control y manejo de errores.

El sistema debe incluir un menú interactivo que permita al usuario realizar las siguientes acciones:

- Agregar personajes.
- Modificar los datos de los personajes.
- Eliminar personajes.
- Visualizar los personajes registrados, junto con su ID, nombre y nivel de poder.
- Registrar peleas entre personajes, guardando la fecha y hora de cada enfrentamiento.
- Consultar el historial de peleas.

## Menú de Gestión de Personajes

El menú deberá tener las siguientes opciones para que el usuario pueda interactuar de manera en que al terminar cada acción se regrese al mismo menú.



### Agregar Personaje

**Descripción**: Esta opción permite al usuario agregar un nuevo personaje al sistema. Al seleccionar esta opción, el sistema solicitará la siguiente información:

- Nombre del personaje (debe ser único).
- Arma del personaje (por ejemplo, Leviatán de Kratos).
- Habilidades (hasta 5 habilidades) que el personaje puede tener.
- Nivel de poder (un número entre 1 y 100, que indica la fuerza o capacidad del personaje).

**Proceso**:

- Se verificará si el nombre ya existe en el sistema (para evitar duplicados).
- Se validarán los datos ingresados, especialmente el nivel de poder.
- El personaje se añadirá a la lista de personajes si todos los datos son válidos.
- El ID del personaje debe ser correlativo y generado por el sistema.

### Modificar Personaje

**Descripción**: Esta opción permite modificar los atributos de un personaje que ya existe en el sistema. El usuario debe seleccionar el personaje a modificar por ID (o nombre, si es único).

- **Modificación**: El usuario podrá cambiar el arma, las habilidades y el nivel de poder del personaje seleccionado.

**Proceso**:

- Se solicitará el ID o nombre del personaje a modificar.
- Se mostrarán los datos actuales del personaje para permitir su edición.
- El sistema validará las nuevas entradas antes de realizar la modificación.

### Eliminar Personaje

**Descripción**: Esta opción permite eliminar un personaje del sistema, de manera que ya no estará disponible en la lista ni podrá ser utilizado en las peleas.

**Proceso**:

- Se solicita el ID del personaje a eliminar.
- El sistema confirmará la eliminación del personaje.
- Si el personaje existe, será eliminado de la lista de personajes.

### Ver Datos de un Personaje

**Descripción**: Esta opción permite al usuario ver la información completa de un personaje registrado en el sistema.

- El usuario debe ingresar el ID del personaje que desea consultar.

**Proceso**:

- El sistema mostrará los datos del personaje: ID, nombre, arma, habilidades y nivel de poder.
- Se mostrará una lista de las habilidades del personaje y su nivel de poder actual.

### Ver Listado de Personajes

**Descripción**: Esta opción permite visualizar todos los personajes registrados en el sistema.

- El sistema mostrará un listado con los ID, nombre y nivel de poder de todos los personajes almacenados.

**Proceso**:

- Si hay personajes registrados, el sistema mostrará todos los datos relevantes.
- Si no hay personajes en el sistema, se informará al usuario que la lista está vacía.

### Realizar Pelea entre Personajes

**Descripción**: Esta opción permite registrar una pelea entre dos personajes seleccionados.

**Proceso**:

- Se solicita el ID de ambos personajes.
- El sistema valida que ambos personajes existan en el sistema.
- Se registra la pelea, almacenando la fecha y hora en que se lleva a cabo el enfrentamiento.

### Ver Historial de Peleas

**Descripción**: Esta opción permite al usuario ver el historial de peleas registradas entre personajes.

- El sistema mostrará la fecha y hora de todas las peleas realizadas.

**Proceso**:

- El sistema listará todas las peleas registradas, indicando los personajes involucrados y la fecha y hora de cada pelea.
- Si no hay peleas registradas, se mostrará un mensaje indicando que no hay peleas en el historial.

### Ver Datos del Estudiante

**Descripción**: Esta opción permite ver la información del estudiante que desarrolló el sistema.

**Proceso**:

- El sistema mostrará los datos del estudiante: nombre, carnet, curso, sección.

## Entregables

Los entregables de la práctica serán los siguientes:

- **Repositorio en GitHub:** Se debe entregar un repositorio en GitHub con el nombre IPC1<SECCIÓN>_2S2025_#Carnet. En este repositorio, se debe incluir el código fuente completo de la práctica, debidamente organizado en una carpeta correspondiente al proyecto. Estructura del repositorio:
    1. Carpeta Práctica1 con el código Java correspondiente.
    2. Código fuente.
    3. Manual Técnico (descripción de los métodos creados, estructura de la aplicación y lógica general) en PDF.
    4. Manual de Usuario (funcionamiento de la aplicación, capturas de pantalla indicando el uso del programa y descripción de cada opción, además de posibles errores y sus respectivas soluciones) en PDF.
    5. Diagrama de flujo

## Consideraciones

- La aplicación debe ser desarrollada en el lenguaje de programación JAVA.
- No se permite utilizar código copiado o bajado de internet.
- El repositorio debe estar bien organizado y contener todos los archivos necesarios (código, manuales, etc).
- La identación y formato del código deben seguir las buenas prácticas de JAVA.
- El código debe estar bien estructurado y limpio, implementando funciones (en caso de ser necesario) reutilizables para cada operación.
- Cualquier código realizado por inteligencia artificial se anulará y se va a reportar a la escuela.
- La práctica debe ser ejecutada en consola (sin interfaz gráfica).
- El IDE por utilizar queda a discreción del estudiante (se recomienda el uso de NetBeans).
- Las copias obtendrán nota de 0 y reporte a la Escuela de Ciencias y Sistemas.
- Durante la calificación se le solicitará al estudiante modificar el código del proyecto con el objetivo de validar la creación de este.
- El estudiante no tendrá derecho a calificación si no muestra sus dos hojas de calificación impresas en el día de la práctica.

- Tiene que tener agregado al auxiliar como colaborador del repositorio para monitorear los avances realizados en el proyecto, de lo contrario obtendrá una penalización.

## Cronograma

| Tarea | Fecha |
|---|---|
| Asignación de la práctica / Entrega del enunciado | 02/Agosto/2025 |
| Fecha de entrega | 15/Agosto/2025 |
| Fecha de calificación | 16/Agosto/2025 |

# Rúbrica de Calificación

| Criterio | Descripción | Funcion | Subfunción | punteo | Total |
|---|---|---|---|---|---|
| **Parte Funcional** | | | | | **40** |
| **Gestión de Personajes** | El sistema permite agregar, modificar, eliminar y visualizar los personajes correctamente | Agregar personaje | Validación de duplicados en nombre | 6 | 12 |
| | | | Entrada Correcta de Datos | 6 | |
| | | Modificar personaje | Modificación de arma y habilidades | 2.5 | 5 |
| | | | Modificación del nivel de poder | 2.5 | |
| | | Eliminar personaje | Eliminación de personaje correctamente | 2.5 | 5 |
| | | | Reajuste de la lista de personajes | 2.5 | |
| | | Ver datos personaje | Visualización del personaje | 4 | 8 |
| | | | Información clara y completa | 4 | |
| **Registro de peleas** | El sistema permite registrar peleas entre personajes con la hora y fecha | Registro de peleas | Registro correcto con fecha y hora | 2.5 | 3.5 |
| | | | Visualización en el historial de peleas | 1 | |
| **Interacción y navegación** | El menú debe ser claro y permitir al usuario seleccionar las opciones adecuadas para cada operación | Menú de opciones | Interactividad y opciones correctas | 3 | 3 |

| | | | | | |
|---|---|---|---|---|---|
| **Control de Errores** | El sistema debe manejar entradas incorrectas y errores de forma adecuada (duplicados, valores fuera de rango) | Manejo de errores | Validación de nombre | 1 | 3.5 |
| | | | Validación de nivel de poder | 1 | |
| | | | Validación de tipo de dato | 0.75 | |
| | | | Validación de opciones | 0.75 | |
| **Parte No Funcional** | | | | | **60** |
| **Uso de ciclos y estructuras de datos (Vectores)** | El sistema debe utilizar correctamente vectores para el almacenamiento de los personajes y ciclos para el recorrido de los mismos. | Uso correcto de ciclos | Recorrido de personajes en arreglo | 2 | 8 |
| | | | Recorrido para habilidades y batallas | 2 | |
| | | Uso correcto de vectores | Almacenamiento de personajes | 2 | |
| | | | Modificación y eliminación | 2 | |
| **Calidad de código** | El código debe estar bien estructurado, organizado y documentado. | Código organizado | Estructura clara y modular | 2 | 8 |
| | | | Código limpio y sin redundancia | 2 | |
| | | Comentarios y buenas prácticas | Comentarios explicativos en funciones clave | 2 | |
| | | | Uso adecuado de nombres descriptivos | 2 | |
| **Entrega & Documentación** | El código debe estar correctamente entregado en GitHub, junto con su documentación especificada. | Repositorio & Documentación | Repositorio bien estructurado | 2 | 12 |
| | | | Manual Técnico | 2 | |
| | | | Manual de Usuario | 2 | |
| | | | Diagrama de flujo | 6 | |

| | | | | | |
|---|---|---|---|---|---|
| **Interactividad del menú** | El menú debe ser interactivo y mostrar todas las opciones, mostrando de manera correcta las selecciones del usuario. | Respuesta correcta del menú | Funcionalidad de cada opción del menú | 1 | 1 |
| **Preguntas** | El estudiante deberá responder preguntas de forma teórica y cambio de código. | Preguntas teóricas | Pregunta 1 | 1 | 20 |
| | | | Pregunta 2 | 1 | |
| | | | Pregunta 3 | 1 | |
| | | | Pregunta 4 | 1 | |
| | | | Pregunta 5 | 1 | |
| | | Cambio de código | Cambio de funciones de código según indique el auxiliar | 15 | |

**Penalizaciones**

| Criterio | Descripción | Porcentaje |
|---|---|---|
| **Incompletitud de entregables** | Deducción de puntos por falta de documentación adecuada o falta de informes. | -30% |
| **Código mal organizado** | Deducción de puntos por mal uso de estructura de código, comentarios o identación (facilidad de entender el código). | -20% |
| **Entrega fuera de tiempo** | El estudiante realizó commits fuera de la fecha de entrega. | -100% |
| **Aplicación desarrollada en otro lenguaje** | La aplicación no está desarrollada en Java. | -100% |
| **Librerías** | Uso de librerías no permitidas. | -75% |
| **Auxiliar de colaborador** | No se agregó al auxiliar como colaborador del repositorio. | -100% |
| **Estructuras no válidas** | Uso de ArrayList, LinkedList, List, Hash Map u otra librería mediante la cual se obtengan métodos de ordenamiento, que no sean vectores. | -75% |
| **Nombre de repositorio** | El repositorio no cuenta con el formato de nombre indicado en el enunciado. | -25% |

| | | |
|---|---|---|
| **Commits insuficientes** | El estudiante no realizó 2 commits semanales | -20% |
| **Copia** | El estudiante realizó copia de código. | -100% |
| **Uso de IA** | El estudiante utiliza inteligencia artificial en la mayoría del proyecto (Deepseek, Chat GPT, Cloude, Copilot, etc) | -100% |

## Valores

En el desarrollo de la práctica, se espera que cada estudiante demuestre honestidad académica y profesionalismo. Por lo tanto, se establecen los siguientes principios:

1. **Originalidad del Trabajo**
    - Cada estudiante o equipo debe desarrollar su propio código y/o documentación, aplicando los conocimientos adquiridos en el curso.
2. **Prohibición de Copias y Plagio**
    - Si se detecta la copia total o parcial del código, documentación o cualquier otro entregable, la calificación será de **0 puntos**.
    - Esto incluye la reproducción de código entre compañeros, la reutilización de proyectos de semestres anteriores o el uso de código externo sin la debida referencia.
3. **Revisión y Detección de Plagio**
    - Se podrán utilizar herramientas automatizadas y revisiones manuales para identificar similitudes en los proyectos.

Todas las copias o plagios obtendrán una nota de 0 y se reportarán al catedrático y a la Escuela de Ingeniería en Ciencias y Sistemas.

## Referencias


- Deitel, P. Deitel, H. (2016). Cómo programar JAVA 10a: Edicion. Pearson Education
- Joyanes, L. (s.f). JAVA 2 Manual de Programación McGrawHill