“Sistema de Gestión Atletas Judo (JUK)”.

Ingeniería Informática

Departamento de Ciencias de la Computación e Informática

Facultad de Ingeniería, Ciencias y Administración

Prof: Oscar Aguayo

Universidad de la Frontera

Integrantes del grupo:

- Integrante 1: Ignacio Essus
- Integrante 2: Benjamin Beroiza
- Integrante 3: Alonso Romero

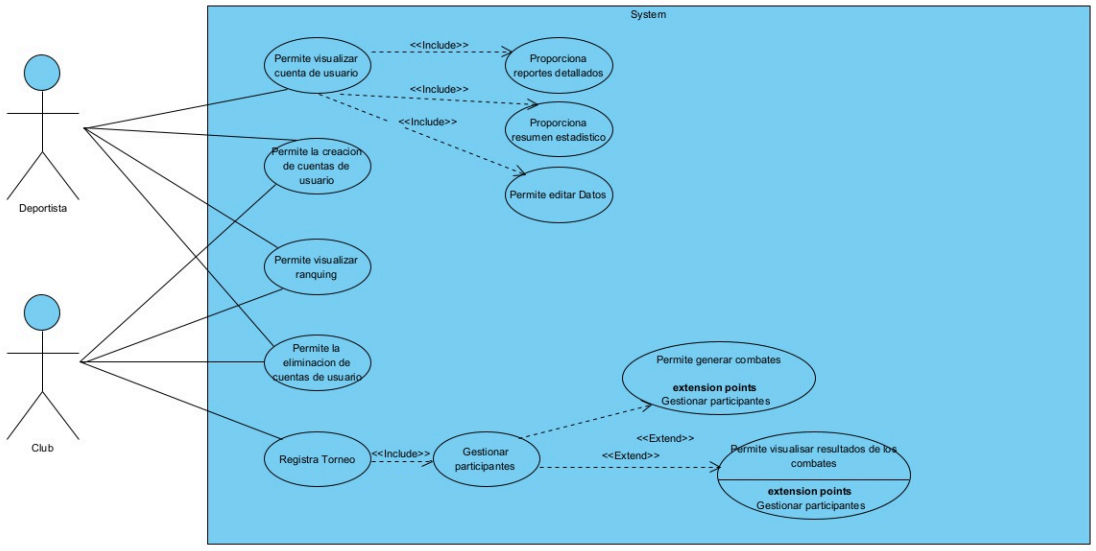
Introducción

Este proyecto tiene como objetivo principal desarrollar un sistema para gestionar la información de competencias de judo, atletas y sus estadísticas de rendimiento (victorias, derrotas, empates). El sistema permite manipular los datos mediante operaciones básicas (CRUD) y generar automáticamente rankings y estadísticas en base a los datos de desempeño de los atletas.

Se diseñó este sistema para facilitar la administración en clubes deportivos, permitiendo almacenar y procesar estadísticas a través de una interfaz de consola simple y clara, orientada a satisfacer las necesidades de organizadores, atletas y entrenadores.

Diseño del Sistema:

Diagrama de casos de uso



Especificaciones de casos de uso

1. **Visualizar cuenta de usuario**
   Consulta los datos personales del usuario.

2. **Crear cuenta de usuario**
   Registro de nuevos usuarios mediante un formulario validado.

3. **Visualizar ranking**
   Muestra la clasificación de los deportistas en el sistema.

4. **Eliminar cuenta de usuario**
   Permite borrar cuentas existentes con confirmación.

5. **Registrar torneo**
   El Club puede registrar nuevos torneos con sus datos básicos.

6. **Gestionar participantes**
   Agrega, edita o elimina participantes de un torneo.

7. **Generar combates**
   Crea emparejamientos entre participantes de un torneo.

8. **Visualizar resultados de combates**
   Presenta los resultados obtenidos en los enfrentamientos.

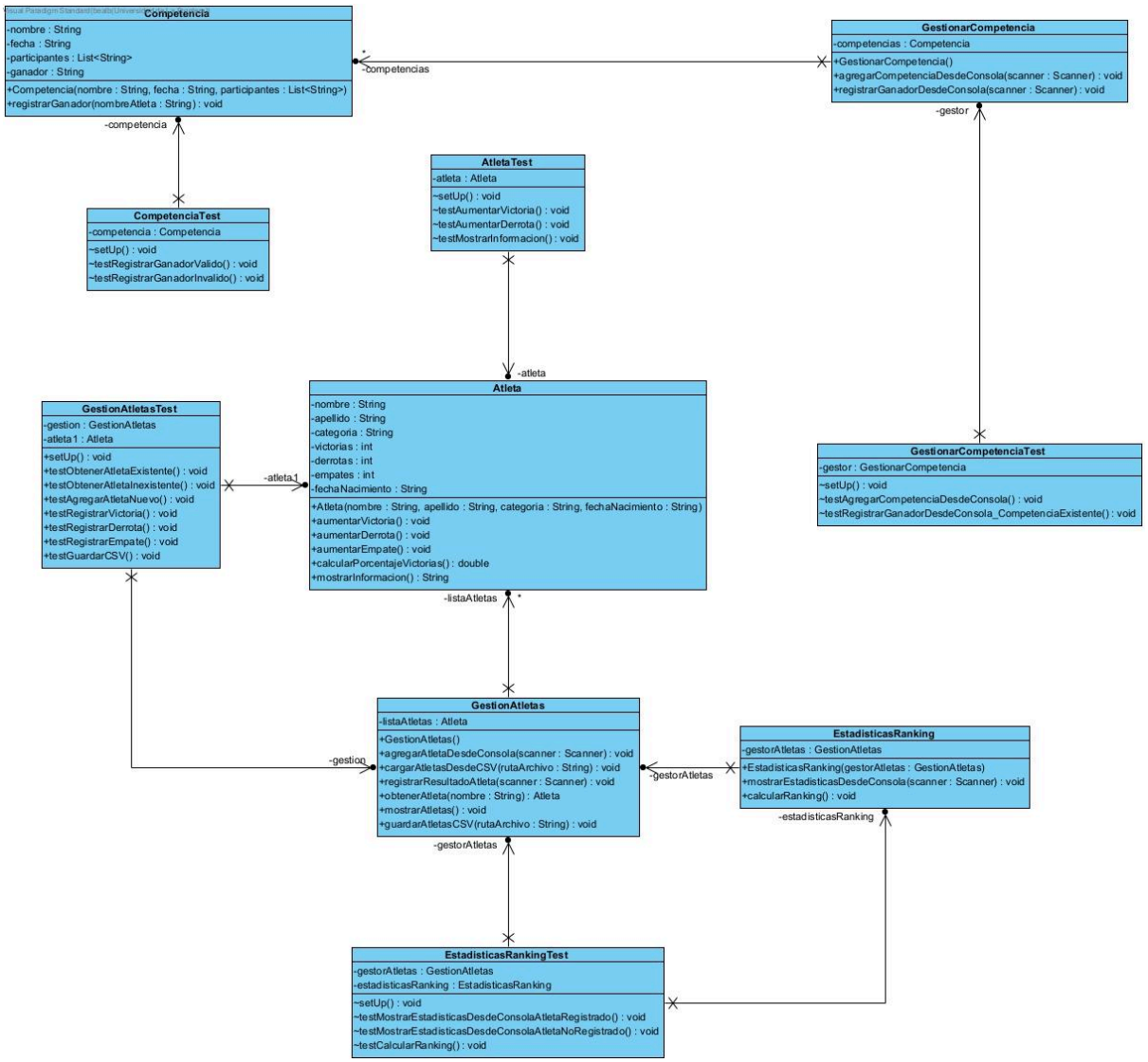
9. **Reportes detallados**
   Genera informes completos del rendimiento de usuarios.

10. **Resumen estadístico**
    Muestra estadísticas generales en forma de resumen.

11. **Editar datos**
    Modifica la información de usuarios o torneos registrados.

Obs: Se consideran todos estos casos de uso para el final del proyecto.

Diagrama de clases



Implementación

1. Descripción de la Solución Desarrollada

El sistema está basado en los principios de la **Programación Orientada a Objetos (POO)**, donde se definieron las siguientes clases clave:

Clases principales:

1. **Atleta**:
   Representa a los atletas con atributos como nombre, categoría, estadísticas (victorias, derrotas y empates), y la fecha de nacimiento. Además, incluye métodos para calcular el porcentaje de victorias y mostrar estadísticas personales.

2. **Competencia**:
   Administra la información de cada competencia: nombre, fecha, lista de participantes y el ganador. Esta clase incluye funcionalidades para registrar a los ganadores, validando que se encuentren inscritos.

3. **GestionAtletas**:
   Encargada de manejar una lista de atletas. Incluye métodos para agregar nuevos atletas, registrar resultados (victorias, derrotas o empates) y guardar/cargar los datos en un archivo CSV.

4. **GestionarCompetencia**:
   Permite agregar y gestionar competencias. Además, posibilita registrar a los ganadores directamente desde la consola.

5. **EstadisticasRanking**:
   Genera y muestra la clasificación de atletas en función de sus estadísticas de victorias.

Interfaz principal:

La clase **Main** implementa un menú interactivo desde la consola, ofreciendo varias funcionalidades al usuario como agregar atletas, registrar competencias, consultar rankings y estadísticas.

---

2. Código Fuente

El archivo principal cuenta con el siguiente esquema modular:

- **Atleta.java:** Clase que implementa los atributos y la lógica para gestionar la información de un atleta. :
- **Competencia.java:** Clase para gestionar competencias, incluyendo participantes y ganador.
- **GestionAtletas.java :** Implementa las operaciones CRUD para la gestión de atletas y trabaja directamente con archivos CSV.
- **GestionarCompetencia.java:** Permite agregar competencias y registrar ganadores.
- **EstadisticasRanking.java:** Calcula el ranking de atletas basado en sus estadísticas.
- **Main.java:** Configura la interfaz de interacción por consola para los usuarios.

Un fragmento representativo de la clase **Atleta** sería:

```
public double calcularPorcentajeVictorias() {
   int total = victorias + derrotas + empates;
   return total == 0 ? 0 : (victorias * 100.0) / total;
}

public String mostrarInformacion() {
   return "Nombre: " + nombre + " " + apellido + ", Categoria: " +
categoria +
           ", Victorias: " + victorias + ", Derrotas: " + derrotas
+ ", Empates: " + empates +
           ", % Victorias: " + String.format("%.2f",
calcularPorcentajeVictorias());
}
```

Aplicación de POO:

1. **Encapsulación**: El acceso a los atributos de las clases está protegido mediante setters y getters.
2. **Modularidad**: Cada funcionalidad se asigna a una clase específica e independiente.

Pruebas Unitarias

1. Estrategia de Pruebas

Para garantizar la calidad y robustez del sistema, se implementaron casos de prueba utilizando el framework **JUnit 5**. Se dividieron en dos categorías principales:

1. **Pruebas de integración**: Comprobación general del sistema operando múltiples módulos.
2. **Pruebas unitarias**: Verificación de métodos individuales en cada clase.

Las herramientas utilizadas incluyeron:

- **JUnit 5**: Para la ejecución y validación de casos de prueba.
- **Java IDE**: IntelliJ para ejecutar y depurar el código.

2. Casos de Prueba y Resultados

Ejemplo 1: Prueba del cálculo correcto de victorias en AtletaTest

```
@Test
void testAumentarVictoria() {
   atleta.aumentarVictoria();
   assertEquals(1, atleta.getVictorias());}
```

Resultado: La prueba asegura que al incrementar una victoria, el conteo total se actualice correctamente.

Ejemplo 2: Registro de un ganador válido en CompetenciaTest

```
@Test
void testRegistrarGanadorValido() {
   competencia.registrarGanador("Ignacio Essus");
   assertEquals("Ignacio Essus", competencia.getGanador());
}
```

Resultado: Valida que los ganadores solo puedan ser registrados si están previamente inscritos.

Ejemplo 3: Ranking en EstadisticasRankingTest

```
@Test
void testCalcularRanking() {
   gestorAtletas.getListaAtletas().sort((a, b) ->
Integer.compare(b.getVictorias(), a.getVictorias()));
   assertEquals("Ignacio",
gestorAtletas.getListaAtletas().get(0).getNombre());
}
```

Resultado: Se verifica que el algoritmo de clasificación ordene correctamente según las victorias

---

Conclusiones y Análisis

El proyecto Sistema de Gestión Atletas Judo (JUK) representa una solución eficaz para la administración de competencias, atletas y estadísticas en el ámbito del judo. Basado en principios de Programación Orientada a Objetos (POO), el sistema destaca por su modularidad, encapsulación y una interfaz de consola intuitiva que facilita operaciones CRUD, generación de rankings y análisis de rendimiento.
Entre sus fortalezas se encuentran:

- Una estructura bien organizada en clases como Atleta, Competencia y GestionAtletas, que promueve la reutilización y mantenibilidad del código.
- Pruebas unitarias robustas con JUnit 5, asegurando la precisión de funcionalidades clave como el cálculo de victorias y el registro de ganadores.
- Documentación clara de casos de uso, desde la gestión de torneos hasta la visualización de estadísticas.

Como áreas de mejora, se podría ver:

- Implementar una interfaz gráfica para mejorar la experiencia de usuario.
- Ampliar las funcionalidades, como comparativas entre atletas o integración con bases de datos externas.

En conclusión, JUK cumple su objetivo de optimizar la gestión deportiva, sentando las bases para futuras iteraciones que podrían expandir su alcance y usabilidad.

LINK REPOSITORIO

https://github.com/Nakotex7906/Judo.git