Explicación de la Clase `GameController` en Profundidad

La clase `GameController` es el núcleo del control de la lógica del juego. Se encarga de coordinar la interacción entre el jugador, la inteligencia artificial (enemigos), el mapa, la interfaz gráfica y otros componentes esenciales

---

Declaración de la Clase

- **Interfaces Implementadas**:
  - `EmpezarTurnoEventListener`: Permite que la clase responda a eventos de inicio de turno.
  - `TileClickListener`: Permite que la clase responda a eventos cuando se hace clic en un tile del mapa

---

Importaciones

La clase importa varias otras clases y paquetes:

- **Clases del Proyecto**:

  `-com.utad.poo.practicaFinalPackage.inout.CreateLogs`: Gestiona los registros de eventos y logs del juego.

  `-com.utad.poo.practicaFinalPackage.inout.IniciarPartidaFichero`: Permite crear objetos PartidaFichero que inicializan la partida desde un archivo.

  `-com.utad.poo.practicaFinalPackage.interfazGrafica.GraphicWindowManager`: Maneja la interfaz gráfica del juego.

  `-com.utad.poo.practicaFinalPackage.interfazGrafica.MapGenerator`: Genera y maneja el mapa del juego.

  `-com.utad.poo.practicaFinalPackage.interfazGrafica.Tile`: Representa cada casilla o tile del mapa.

  `-com.utad.poo.practicaFinalPackage.interfazGrafica.Utility`: Proporciona utilidades para la interfaz gráfica.

  `-com.utad.poo.practicaFinalPackage.personajes.Personaje`: Clase base para los personajes del juego.

  `-com.utad.poo.practicaFinalPackage.partida.Turno`: Gestiona los turnos dentro del juego.

  `-com.utad.poo.practicaFinalPackage.partida.GameArranger`: Configura y organiza los elementos del juego al inicio.

-`com.utad.poo.practicaFinalPackage.partida.AI`: Implementa la inteligencia artificial de los enemigos

Atributos de la Clase

1. **Turno turno**: Gestiona las fases y acciones que ocurren en cada turno del juego. Coordina las acciones de los personajes y actualiza el estado del juego al finalizar el turno.
2. **Personaje jugador**: Representa al personaje controlado por el usuario. Contiene información sobre su estado, posición, atributos y habilidades.
3. **GameArranger gameArranger**: Se encarga de organizar y configurar los elementos iniciales del juego. Crea instancias de personajes, enemigos y establece sus posiciones iniciales.
4. **GraphicWindowManager graphicWindowManager**: Maneja la interfaz gráfica de usuario (GUI). Controla ventanas, paneles, botones y otras componentes visuales. Interactúa con el usuario y refleja el estado actual del juego.
5. **MapGenerator mapa**: Genera y administra el mapa donde se desarrolla el juego. Crea los tiles y coloca los elementos (personajes, enemigos, items) en el mapa.

Constructor

- **Inicialización**: Crea una instancia de `GraphicWindowManager` utilizando el mapa proporcionado. Inicializa `GameArranger` para configurar los elementos del juego. Asigna el mapa al atributo `mapa`. Inicializa `jugador` y `turno` como `null`, para ser configurados posteriormente

---

Métodos Principales

1. startGame()

- **Funcionalidad**: Inicia el juego configurando todos los elementos necesarios. Integra los componentes del juego con la interfaz gráfica. Establece los listeners para manejar eventos de usuario e interacción con el mapa.
- **Relaciones**:
  - `GameArranger`: Se utiliza para crear los personajes y enemigos.
  - `GraphicWindowManager`: Configura la interfaz gráfica con el jugador.
  - `MapGenerator`: Genera el mapa y posiciona a los personajes.
  - Listeners: `GameController` se registra como listener para eventos de turno y clics en el mapa

---

2. ejecutarTurno()

- **Funcionalidad**:
  - Gestiona las acciones que ocurren durante un turno.
  - Coordina las acciones de los enemigos a través de la IA.
  - Actualiza el estado del juego y de la interfaz gráfica al finalizar el turno.
- **Relaciones**:
  - `AI`: Controla las decisiones y acciones de los enemigos.
  - `Turno`: Ejecuta las acciones de todos los personajes durante el turno.
  - `GraphicWindowManager`: Refleja los cambios en la interfaz gráfica tras el turno

---

3. onExecuteTurn()

- **Funcionalidad**:
  - Se ejecuta cuando se inicia un nuevo turno.
  - Llama a `ejecutarTurno()` para procesar el turno.
  - Verifica las condiciones de victoria o derrota.
  - Utiliza `JOptionPane` para mostrar mensajes al usuario.
  - Registra eventos importantes utilizando `CreateLogs`.
- **Relaciones**:
  - `CreateLogs`: Registra eventos como la victoria o derrota del jugador.
  - `JOptionPane`: Interactúa con el usuario mostrando diálogos informativos.
  - `GameArranger`: Verifica si todos los enemigos han sido derrotados

---

4. onTileClicked(Tile tile)

- **Funcionalidad**: Maneja el evento cuando el usuario hace clic en un tile del mapa. Actualiza el tile objetivo del jugador. Marca visualmente el tile seleccionado. **Relaciones**: `Tile`: Interactúa con los tiles para establecer y marcar el objetivo. `Personaje`: Actualiza el estado del jugador con el nuevo objetivo

---

Relaciones con Otras Clases

1. GraphicWindowManager **Función**: Gestiona la interfaz gráfica del juego. **Interacción**: `GameController` utiliza `graphicWindowManager` para configurar y actualizar la interfaz. Métodos como `updateInventoryPanel`, `updateStatsPanel`, `updateMapPanel` reflejan cambios en la GUI.

2. MapGenerator Función: Genera y administra el mapa del juego. **Interacción**: `GameController` utiliza `mapa` para generar el mapa y posicionar a los personajes. Accede a métodos como `generateMap` y `setPlayers`.

3. GameArranger Función: Configura los elementos del juego al inicio. **Interacción**: `GameController` llama a `gameArranger.startGame()` para inicializar los personajes y enemigos. Utiliza `gameArranger` para obtener el jugador y la lista de enemigos.

4. AI Función: Controla las decisiones y acciones de los enemigos mediante inteligencia artificial. **Interacción**: `GameController` crea una instancia de `AI` y llama a `decideActions()` para que los enemigos actúen.

5. Turno Función: Gestiona las acciones durante cada turno del juego. **Interacción**: `GameController` crea una instancia de `Turno` con todos los personajes y ejecuta `iniciarTurno()`.

6. Personaje Función: Clase base para el jugador y enemigos. **Interacción**: `GameController` maneja al `jugador`, actualizando su estado y respondiendo a sus acciones. Actualiza componentes de la interfaz relacionados con el `jugador`.

7. Interfaces de Eventos `EmpezarTurnoEventListener`: Permite a `GameController` responder al evento de inicio de turno desde la interfaz. `TileClickListener`: Permite a `GameController` manejar el evento de selección de tiles en el mapa.

8. CreateLogs

- **Función**: Gestiona el registro de eventos y logs del juego. **Interacción**:
- `GameController` utiliza `CreateLogs` para registrar eventos importantes como victorias o derrotas

---

Resumen de Flujo

1. **Inicio del Juego**: Se crea una instancia de `GameController`, inicializando los componentes esenciales. Se llama a `startGame()`, configurando el mapa, personajes y la interfaz gráfica. Los listeners se establecen para manejar eventos de usuario.
2. **Ejecución de Turnos**: Al iniciarse un turno (`onExecuteTurn()`), se llama a `ejecutarTurno()`. La IA decide las acciones de los enemigos. Se ejecutan las acciones de todos los personajes a través de `Turno`. Se actualiza la interfaz gráfica para reflejar el nuevo estado del juego.

3. **Interacción del Usuario**: El usuario interactúa con el mapa haciendo clic en tiles. `onTileClicked(Tile tile)` maneja estos eventos, actualizando el objetivo del jugador.

4. **Finalización del Juego**: Se verifica si el jugador ha ganado o perdido. Se muestran mensajes al usuario y se registran los eventos. El juego se cierra si ha terminado.

La clase `GameController` es esencial para el funcionamiento del juego, actuando como puente entre la lógica del juego, la inteligencia artificial, y la interfaz gráfica. Coordina la interacción entre múltiples componentes, asegurando que el juego se desarrolle de manera fluida y que las acciones del usuario y de la IA se gestionen adecuadamente.

Al analizar los objetos que contiene y las importaciones, se puede ver que `GameController`: **Administra el flujo del juego**: Inicio, ejecución de turnos y finalización. **Interactúa con la interfaz gráfica**: A través de `GraphicWindowManager`, reflejando los cambios y recibiendo input del usuario. **Coordina la IA y las acciones de los personajes**: Utilizando `AI` y `Turno`. **Gestiona eventos y logs**: A través de `CreateLogs` y mostrando información al usuario mediante `JOptionPane`.

Su relación con otras clases es simbiótica, ya que depende de ellas para funcionar y, a su vez, proporciona dirección y control al flujo del juego.