Deusto
Facultad de Ingeniería
Faculty of Engineering

Programación III
Curso 2025/2026

Nombre y Apellidos _____

7 de noviembre de 2025

Examen parcial - 10 puntos (Duración: 2,5h)

Antes de comenzar a resolver el examen debes seguir los pasos indicados en la hoja de instrucciones

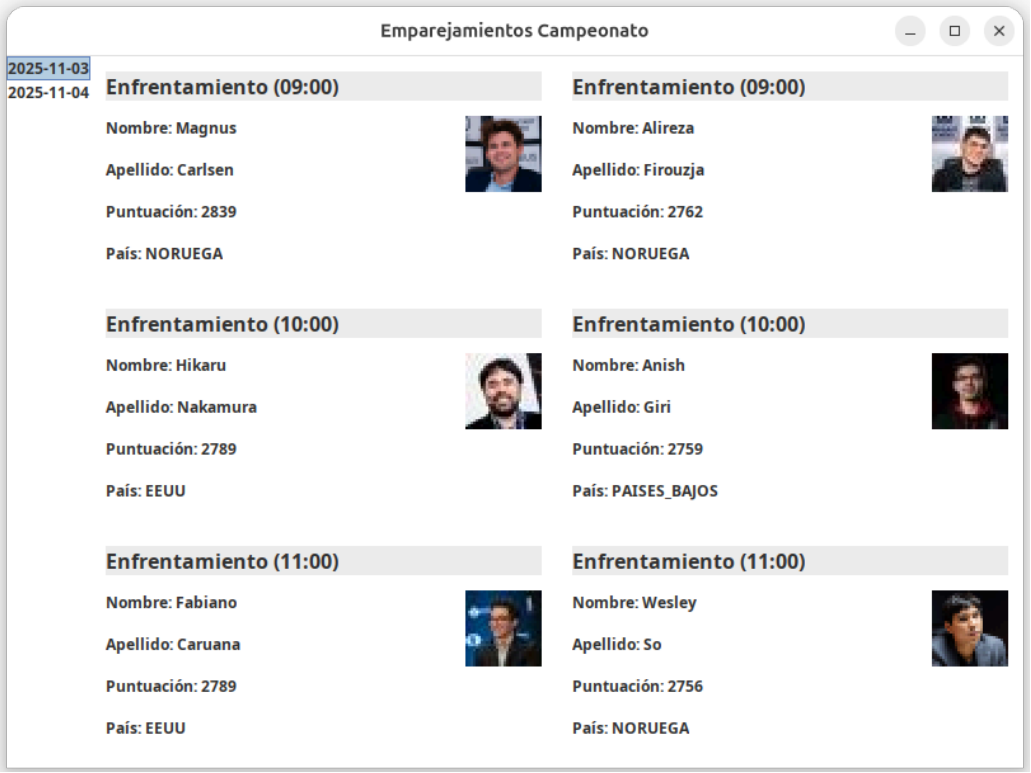
En este examen vamos a trabajar sobre una aplicación de gestión de un campeonato de ajedrez.

Para cada tarea, se indica su puntuación, la clase en la que debes añadir el código y una estimación del número de líneas de código necesarias para resolverlo. El código base proporcionado está organizado en varios paquetes, cada uno de ellos relacionado con una competencia y conteniendo una clase con un método `main()` para poder probar ese ejercicio de manera independiente al resto.

Competencia CE-1: Interfaces gráficas con Swing [6 puntos]

Tarea 1.1: Layouts [2 puntos] [VentanaEmparejamientos] [85 líneas]

El objetivo de esta tarea es desarrollar una ventana que permita visualizar los emparejamientos de ajedrez en el campeonato. En el código fuente proporcionado tienes un código inicial que muestra una ventana con un JList que permite seleccionar entre los días del campeonato (hay únicamente dos días según los datos de prueba). Cuando el usuario selecciona un día en la lista, la ventana debe mostrar los datos de los participantes que tienen enfrentamientos ese día, tal y como se muestra en la imágen.



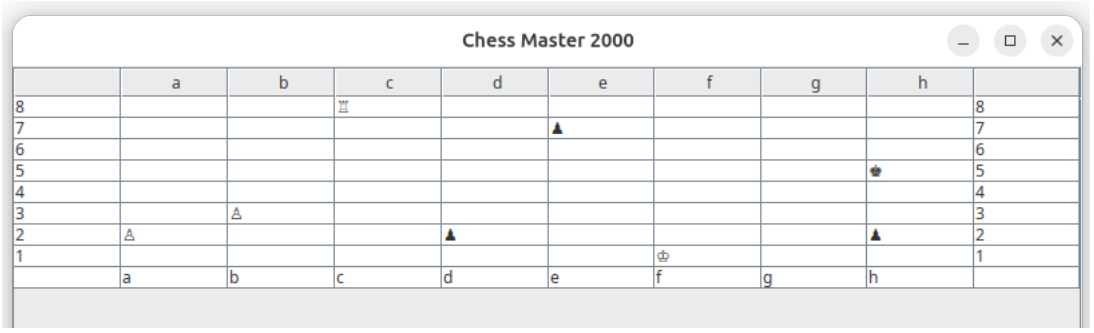
Ten en cuenta lo siguientes aspectos importantes:

- Para implementar este ejercicio debes usar componentes básicos y layouts (GridLayout, FlowLayout, BorderLayout).
- Los jugadores que se enfrentan en un día deben aparecer en la misma fila, indicando en el título la hora del enfrentamiento.
- Tienes disponible el método *cargarImagen* para cargar la imagen de perfil y escalarla a un tamaño. Las imágenes son de baja calidad, por lo que aparecen pixeladas.
- El número máximo de enfrentamientos en un mismo día es 3. Ten en cuenta que si un día tiene menos de 3 enfrentamientos, deberás rellenar los huecos disponibles con paneles vacíos.

Tienes un vídeo en el directorio ***resources/doc*** en el que se muestra el resultado final de la tarea.

Tarea 1.2: Modelo de datos [1,5 puntos] [VentanaTablero] [60 líneas]

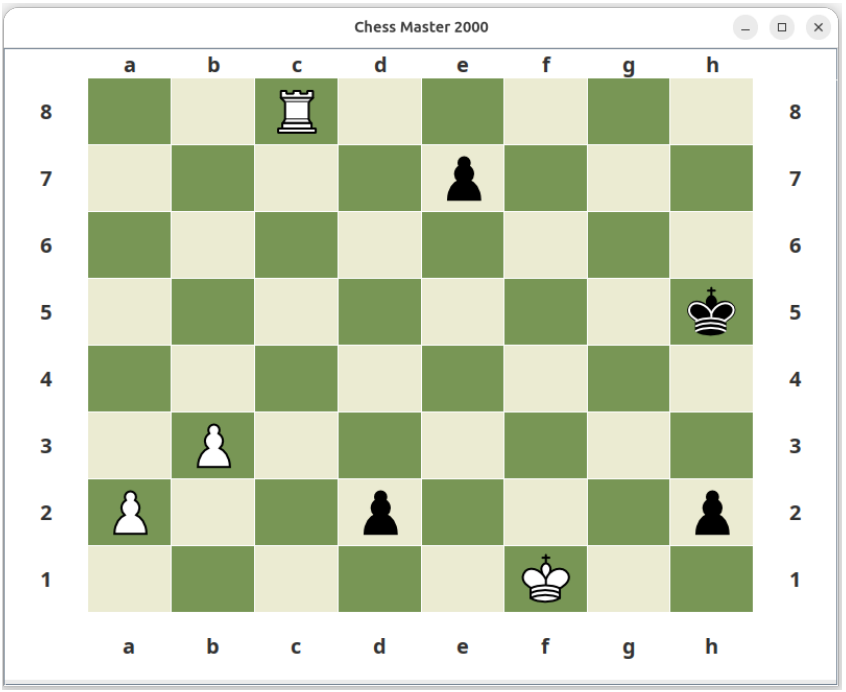
En esta tarea debes implementar un modelo de datos para que el JTable pueda utilizar los datos contenidos en la matriz *tablero (PiezasAjedrez[][] tablero)*, que representa el estado del tablero de ajedrez, y que ya está inicializada en la ventana con datos de prueba. La imagen siguiente muestra un resultado intermedio tras utilizar el nuevo modelo en la tabla, aunque puedes también utilizar la imagen final del tablero de ajedrez como referencia.



Ten en cuenta que las etiquetas para hacer referencia a una celda concreta del tablero deben aparecer tanto en la parte superior e inferior (letras), como en los dos lados del tablero (números). El símbolo de cada pieza de ajedrez está representado por un String con un símbolo Unicode, tal y como se muestra en la figura.

Tarea 1.3: Renderizado de la tabla [1,5 puntos] [VentanaTablero] [60 líneas]

En esta tarea debes conseguir que la tabla tenga, finalmente, el aspecto visual mostrado en la imagen. En el caso de que no hayas podido implementar el modelo en la tarea anterior, puedes realizar esta tarea usando directamente los datos de la matriz tablero. Ten en cuenta que cada PiezaAjedrez contiene la ruta a la imagen correspondiente. Los colores utilizados para los escaques son (120, 151, 88) para los oscuros y (239, 238, 211) para los claros.



Tarea 1.4: Eventos [1 puntos] [VentanaTablero] [10 líneas]

En esta tarea debes hacer que, cuando el usuario haga clic con el ratón, se añada un peón blanco en esa celda del tablero. Tienes un vídeo en el directorio ***resources/doc*** en el que se muestra el resultado final de la tarea.

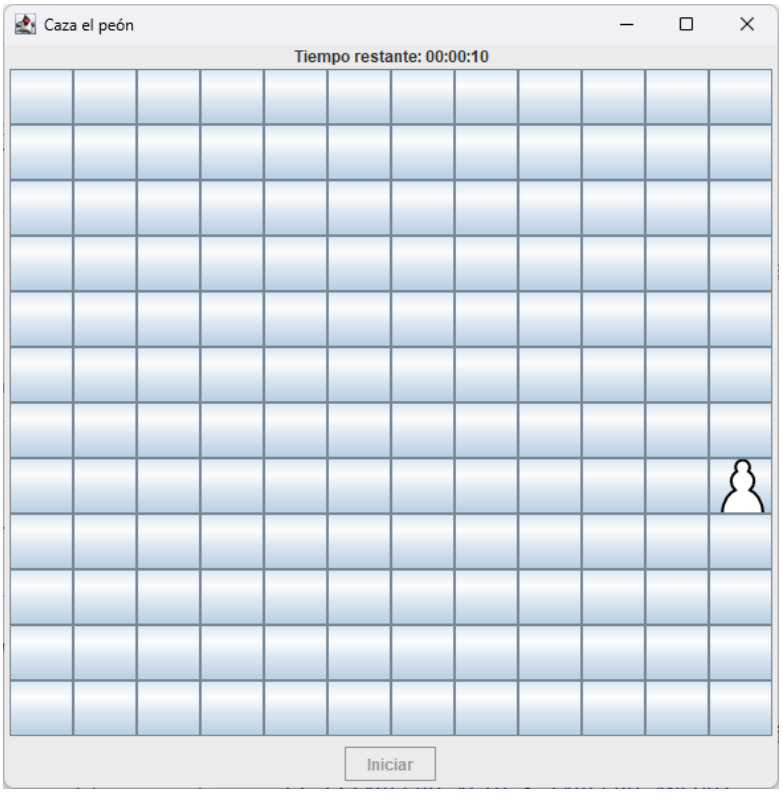
Competencia CE-2: Hilos

Tarea 1.5 Hilos [4 puntos] [VentanaJuego] [60 líneas]

En esta tarea debes completar un minijuego en el que el usuario debe cazar un peón que se mueve de manera aleatoria por un tablero. El usuario tiene un tiempo determinado para hacer click sobre la celda que contiene al peón, o perderá el juego. El peón se debe mover de manera aleatoria por el tablero permaneciendo en cada posición un tiempo de 500 ms. El juego no comienza hasta que el usuario pulsa sobre el botón "Iniciar" que se encuentra en la ventana.

Debes implementar tanto la funcionalidad del temporizador, que se muestra en la ventana con el tiempo restante, como el movimiento del peón por el tablero. El tablero es un array de botones (JButton[]) que se corresponde directamente con cada una de las casillas visuales del mismo. Si el tiempo termina sin haber cazado el peón se debe mostrar un mensaje de fin de juego, mientras que si el usuario consigue cazarlo se mostrará un mensaje de enhorabuena.

Para ayudarte en esta tarea, dispones ya de un código inicial que construye la interfaz visual del juego, eventos ya implementados y algunos métodos de utilidad.



Tienes un vídeo en el directorio ***resources/doc*** en el que se muestra el resultado final de la tarea.