LEA ESTE DOCUMENTO DETENIDAMENTE

Este es un proyecto **grupal**. Por tanto, todos los miembros del grupo deben involucrarse en el diseño e implementación de la solución que se solicita. La entrega del proyecto incluirá un reporte de auto y coevaluación, donde cada miembro del grupo calificará sus contribuciones al proyecto y las de sus compañeros.

Es su responsabilidad agruparse a tiempo para la entrega del proyecto. Usted debe indicar su grupo en el Aula Virtual hasta el **martes 19 de mayo**. Los estudiantes que no se hayan unido a ningún grupo para entonces, serán asignados aleatoriamente por el profesor. A partir de entonces, no se aceptarán cambios en la conformación de los grupos. **EN NINGUNA CIRCUNSTANCIA SE ACEPTARÁN ENTREGAS INDIVIDUALES.**

Si algún grupo experimenta el incumplimiento sistemático de alguno de sus miembros, esta situación debe ser reportada al profesor **TAN PRONTO COMO SE PRESENTE**. Cualquier estudiante cuyo incumplimiento sea comprobado, será retirado del grupo y **DEBERÁ IMPLEMENTAR EL PROYECTO DE MANERA INDIVIDUAL SOBRE EL 50% DEL PUNTAJE POSIBLE.**

Si algún estudiante siente que otro (u otros) toma(n) el control del grupo y no permite(n) a los demás miembros hacer aportes o colaborar en el proyecto, debe reportar esta situación al profesor **TAN PRONTO COMO SE PRESENTE**. Cualquier estudiante que pretenda monopolizar el trabajo del proyecto o impida a los demás hacer aportes será separado del grupo y **DEBERÁ IMPLEMENTAR EL PROYECTO DE MANERA INDIVIDUAL Y SOBRE EL 50% DEL PUNTAJE POSIBLE.**

Si algún estudiante reporta de manera extemporánea cualquiera de las dos situaciones detalladas en los dos párrafos precedentes, será penalizado con su 25% de la nota total obtenida en el proyecto. Reporte novedades al profesor **TAN PRONTO COMO ÉSTAS SE PRESENTEN**.

Considere las políticas de buena conducta académica que se explicaron en la primera clase del curso respecto al plagio y demás violaciones del código de Ética de la ESPOL. Los autores del proyecto deben ser usted y sus compañeros de grupo (no otras personas, **ni motores de Inteligencia Artificial**). Cualquier indicio de lo contrario, será reportado a las unidades correspondientes para el tratamiento pertinente.

Finalmente, **siga las instrucciones de este documento** para evitar inconvenientes. Si tiene dudas, consulte al profesor en lugar de asumir cosas que pueden ser incorrectas.

El incumplimiento de instrucciones explícitas indicadas en este documento o indicadas en nuestras sesiones teóricas derivará, de manera inapelable, en la penalización y rebaja de puntos sobre su entrega.

Proyecto – PAO 1 2025

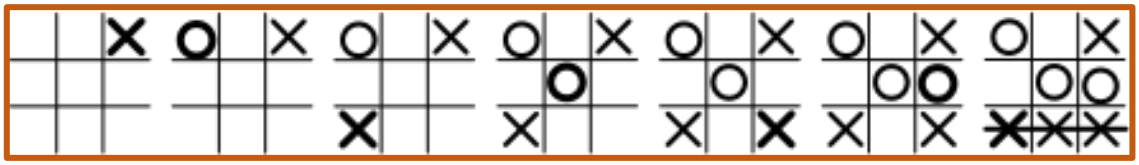
"Tres en raya" contra el computador

Introducción

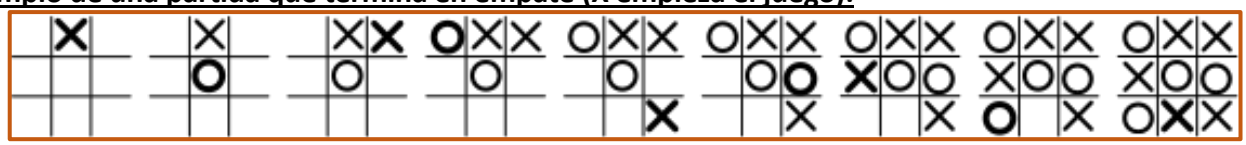
"Tres en Raya" es un juego en el que dos jugadores (representados por los símbolos **O** y **X**), marcan los espacios de un **tablero de 3×3** de manera alternada. En cada turno, cada jugador debe colocar su símbolo una vez sobre una casilla libre. El primer jugador que logra llenar una fila, columna, o diagonal con tres de sus símbolos, gana el juego.

La siguiente secuencia de una partida de tres en raya es iniciada por el jugador **X**. Cada tablero mostrado corresponde a un turno distinto.

En este ejemplo, X gana:



Ejemplo de una partida que termina en empate (X empieza el juego):



En este proyecto, usted implementará una aplicación gráfica de JavaFX que permitirá a una persona jugar el "Tres en Raya" contra la computadora. A lo largo de la implementación de su solución, usted y sus compañeros de grupo deberán decidir cuáles de las estructuras de datos revisadas en el curso son las más apropiadas para implementar distintas partes del juego. En combinación con otros elementos, algunas de estas estructuras se utilizarán para dotar a la computadora de "inteligencia" para que pueda jugar el "Tres en Raya" en contra de un jugador humano.

Utilidad de un Tablero

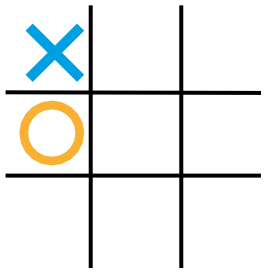
Al igual que los jugadores humanos, en un juego, la computadora debe "decidir" qué movimiento realizar cada vez que sea su turno. Como no es factible calcular de antemano todos los posibles movimientos que podrían derivar en una jugada ganadora, la computadora debe tomar una decisión óptima **en cada turno**. Dicha decisión debe considerar el estado actual del tablero y una función de utilidad **u** que indica lo cerca que se está de una jugada ganadora (o perdedora). Al considerar los valores retornados por esta función de utilidad, la computadora puede decidir qué movimiento es más conveniente dado un estado específico del tablero.

Para el "Tres en Raya", dato un tablero **t**, la función de utilidad se define como:

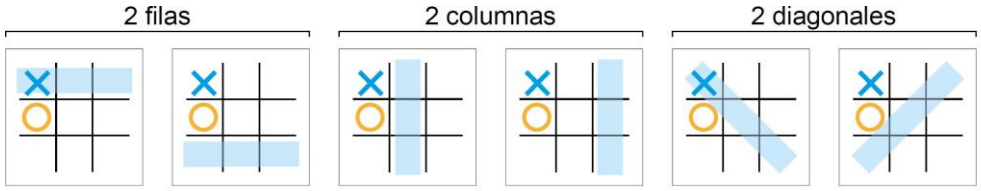
$$u_{jugador}(t) = P_{jugador} - P_{oponente}$$

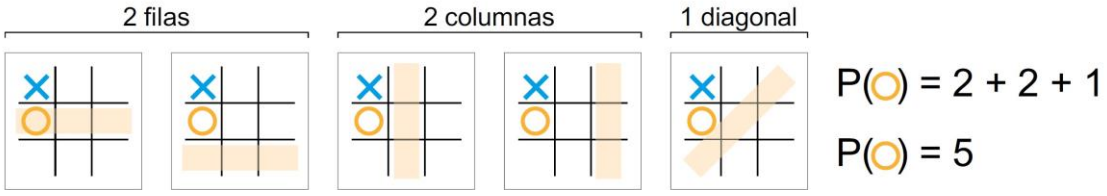
donde P_j es el número total de filas, columnas y diagonales disponibles en el tablero **t** para el jugador **j**.

Considere el siguiente tablero:



El cálculo de los valores de **P** para cada jugador se ilustra en las figuras mostradas a continuación:





Por tanto, la utilidad del tablero mostrado para el jugador que juega con el símbolo **X** sería:

- $u_X = P_X - P_O$
- $u_X = 6 - 5$
- $u_X = 1$

Decidiendo Jugadas

Utilizando la función de utilidad descrita en la sección anterior, su proyecto dotará a la computadora de una estrategia de decisión para jugar el "Tres en Raya". Dicha estrategia puede resumirse en la siguiente frase:

> "*Computadora, elije el mejor movimiento para ti misma, asumiendo que el humano escogerá el peor para ti*".

Esta filosofía, aplicada ampliamente en el mundo de la inteligencia artificial y los juegos, se conoce con el nombre de ***minimax***. En términos matemáticos, ***minimax*** consiste en **mini**mizar la pérdida **máx**ima esperada.

En el proyecto que usted implementará, cada vez que le toque jugar, la computadora aplicará la estrategia **minimax** para decidir qué movimiento realizar.

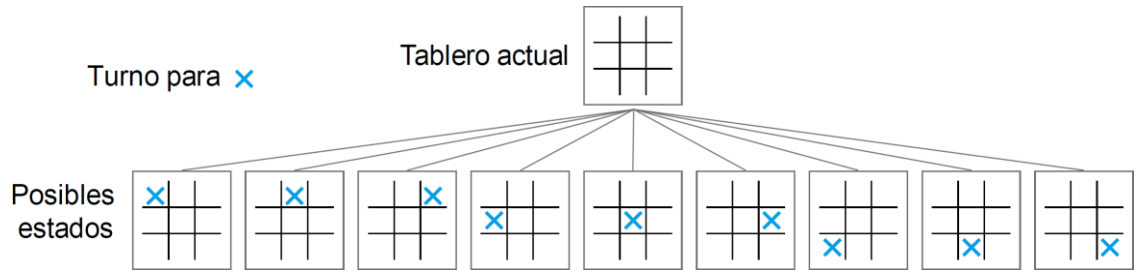
Para el **"Tres en Raya",** el *algoritmo* de **minimax** consiste en:

1. Generar los posibles estados que un tablero podría tomar después de dos turnos (el propio y el del oponente).
2. Calcular la utilidad de cada uno de los tableros que podrían ser generados por el oponente.
3. Encontrar la utilidad mínima de cada familia de tableros que podría generar el oponente y asociar esa utilidad mínima al padre respectivo.
4. Elegir el tablero propio (es decir, el tablero del jugador que tiene el turno) con la máxima de todas las utilidades mínimas y efectuar el movimiento que lleve a este tablero.

Para el mejor desarrollo, se describe con más detalle cada uno de estos pasos.

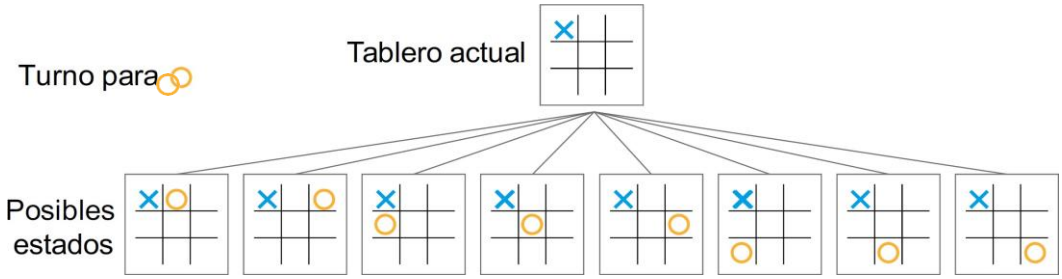
Paso 1: Generar posibles estados

Generar los posibles estados de un tablero significa producir todos los tableros que podrían resultar a partir del tablero actual, dado un turno específico. **Por ejemplo**, si el tablero actual está vacío y le toca el turno al jugador con el símbolo **X**, hay nueve posibles tableros (es decir, estados) que podrían generarse. Estos se muestran en la siguiente figura:



Lo de arriba ocurre, porque en un tablero vacío, hay nueve posibles lugares donde el jugador con el símbolo X podría jugar y, cada uno de estos lugares, derivaría en un tablero distinto.

Para el tablero mostrado a continuación, si el turno es para el jugador con el símbolo **O**, habría únicamente ocho posibles estados:

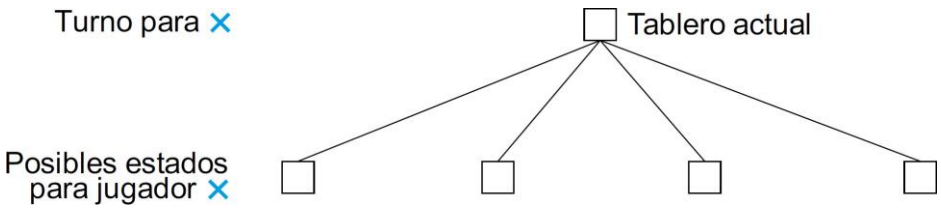


El algoritmo de **minimax** indica que se deben generar los estados del tablero considerando **dos turnos**.

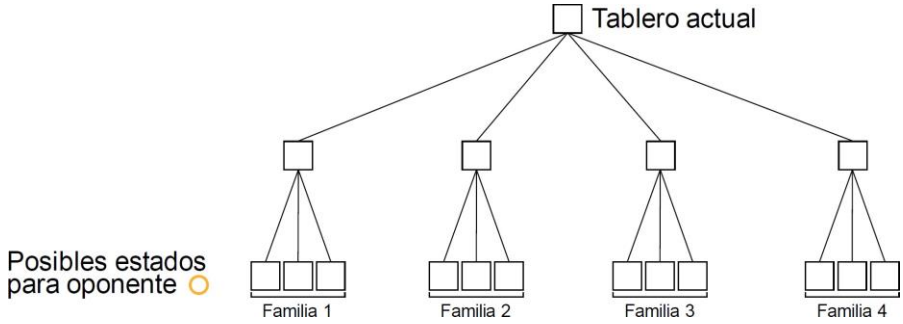
Esto significa que se deben generar los posibles estados que resulten del turno actual y, por cada uno de estos, los tableros que podrían resultar por el turno del oponente.

Las siguientes figuras ilustran esto:

Primeros estados generados:



Segunda generación de estados:



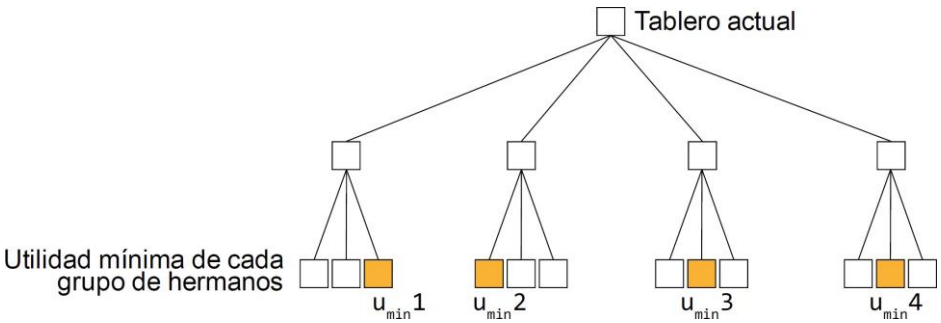
Como se puede apreciar, el Paso 1 genera un árbol n-ario, es decir, un árbol en el que cada uno de sus nodos puede tener cero o más hijos.

Pasos 2 y 3: Cálculo de utilidades mínimas en tableros posibles del oponente

En este paso, se calcula la **utilidad de cada uno de los tableros** que podrían surgir tras una jugada del oponente. Estos tableros corresponden a las **hojas del árbol de decisión** mostrado en el paso anterior.

Una vez asignadas las utilidades a cada hoja, se procede a **evaluar cada conjunto de tableros hermanos** (es decir, todos los que pertenecen a una misma rama o "familia") para identificar cuál representa la **peor opción posible** desde el punto de vista del jugador actual, asumiendo que el oponente jugará de forma óptima para minimizar su utilidad.

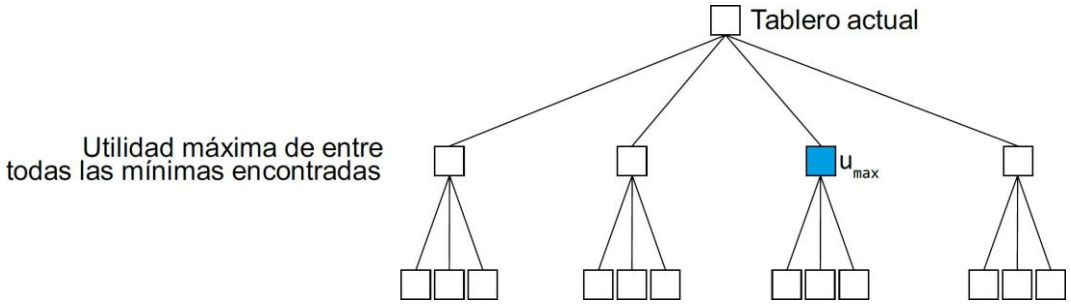
A este valor se le llama **utilidad mínima (u_{min})** de la familia, y se selecciona el menor entre todos los tableros del grupo. En la figura que acompaña esta explicación, los tableros marcados en **color naranja** representan estos valores mínimos de utilidad para cada conjunto de hojas.



Paso 4: Selección del tablero con utilidad máxima para el jugador actual

Una vez calculadas las utilidades mínimas (u_{min}) de cada grupo de tableros posibles del oponente (es decir, las "familias" de jugadas posteriores), estas utilidades se asignan a los nodos padre correspondientes, que representan las decisiones disponibles para el jugador actual.

En este punto, el jugador debe decidir qué movimiento realizar. Para ello, se evalúan todas las jugadas disponibles (los nodos padre) y se selecciona aquella que tenga la mayor utilidad, bajo el supuesto de que el oponente responderá de forma óptima (es decir, minimizando la utilidad del jugador).

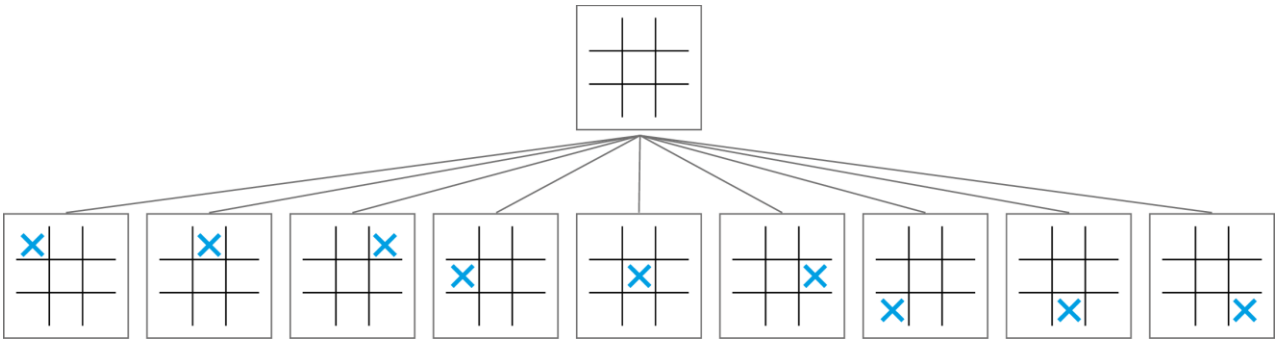


El tablero con la utilidad máxima entre todas las opciones posibles se considera la mejor jugada posible para el turno actual. En la figura asociada, dicho tablero está resaltado en color celeste, indicando claramente cuál debería ser el movimiento elegido por el jugador.

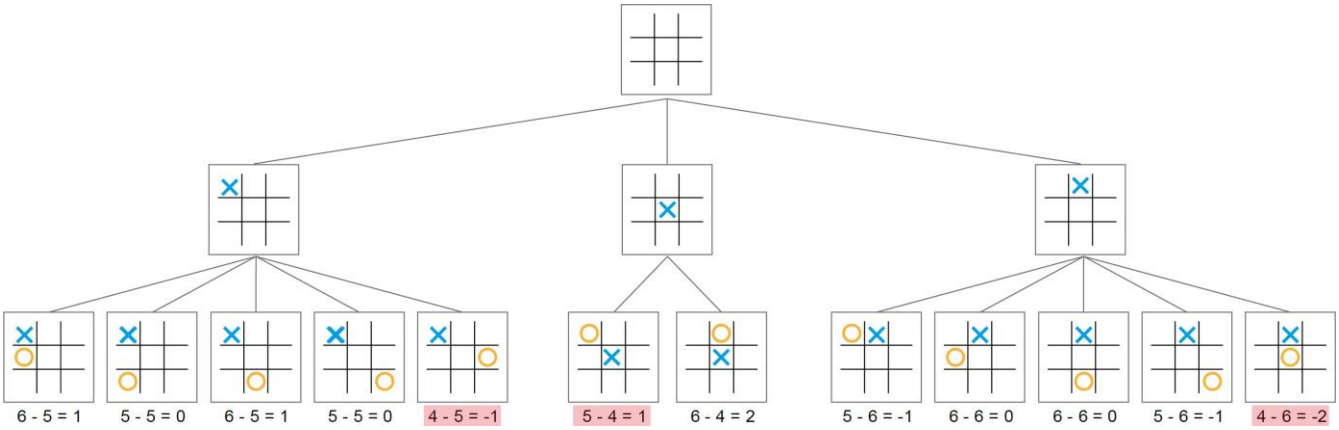
Ejemplo Ilustrativo

El algoritmo descrito en la sección anterior se ilustra aquí con un ejemplo concreto, que inicia con un tablero vacío. El turno inicial corresponde a la computadora, quien juega con el símbolo **X**. Para decidir en qué posición le conviene marcar su símbolo, la computadora aplicará el algoritmo minimax, de acuerdo con lo descrito anteriormente.

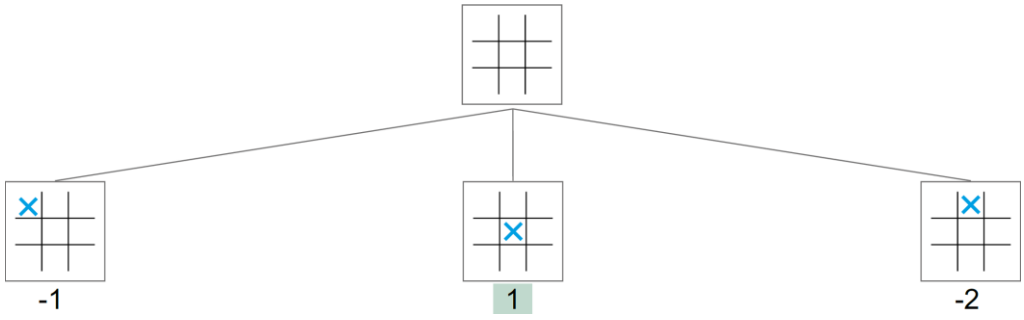
En primer lugar, se generan los posibles tableros que podría producir la jugada de la computadora:



Por **cada uno** de estos tableros, se generan aquellos que podrían resultar del movimiento hecho por el oponente (es decir, por el jugador humano).
Una muestra de estos tableros se muestra en la siguiente figura. Por restricciones de espacio, no se muestran todos los posibles 72 tableros del segundo nivel ni los 9 del primero (que sí se muestran en la figura anterior).



La imagen anterior incluye también los valores de la función de utilidad de cada uno de los tableros del segundo nivel del árbol.
Los valores resaltados con rojo indican el mínimo dentro de cada familia. Este valor mínimo es luego asociado a cada uno de los padres, como se muestra en la siguiente figura:



En este punto, la computadora debe escoger al tablero que garantiza la mayor utilidad. En el ejemplo dado, es el tablero con utilidad 1, que se encuentra en el centro de la figura anterior y cuya utilidad aparece resaltada con color verde. Esto significa que, dado un tablero vacío, a la computadora le conviene colocar su símbolo en la celda central del tablero, ya que este movimiento maximiza su ganancia potencial y, al mismo tiempo, minimiza las posibles de perder.

Modelado del problema y requerimientos mínimos

El escenario descrito previamente puede representarse eficazmente utilizando **árboles n-arios** combinados con otras **estructuras de datos abstractas (TDAs) lineales**. En el contexto de este curso, se espera que el grupo diseñe e implemente aquellos TDAs que resulten más adecuados para resolver el problema propuesto.

1. **Restricción importante:** Aunque pueden apoyarse en las colecciones lineales provistas por el **Java Collection Framework** (como listas, pilas, colas, conjuntos y mapas), **toda estructura de datos no lineal** que se utilice —especialmente los árboles— **debe ser implementada por el equipo de desarrollo**.
2. **Uso obligatorio de árboles:** El uso de estructuras arbóreas no es opcional.
   Cualquier solución que **no incluya árboles** como parte de su lógica principal será **automáticamente invalidada**, y se asignará una calificación de **cero (0)** a todos los integrantes del grupo.
   Recuerde: este curso está orientado al estudio y aplicación de estructuras de datos. Por tanto, la implementación de estructuras adecuadas es un componente esencial del proyecto.

Requisitos funcionales mínimos

La aplicación debe consistir en una **interfaz gráfica desarrollada con JavaFX**, en la cual el usuario pueda jugar una partida de *tres en raya* contra la computadora.
La interfaz debe incluir controles que permitan:

1. **Seleccionar el símbolo** para cada jugador (X u O).
2. **Indicar quién inicia la partida** (humano o computadora).

Estos parámetros deben ser seleccionables por el usuario mediante **controles gráficos intuitivos** (por ejemplo, botones, menús desplegables o cuadros de selección). Asimismo, el jugador humano deberá poder indicar sus movimientos de forma directa e interactiva (por ejemplo, haciendo clic sobre la celda del tablero donde desea colocar su símbolo).

Comportamiento esperado

- La computadora debe tomar sus decisiones utilizando un algoritmo basado en estructuras de datos (por ejemplo, **Minimax** con poda alfa-beta).
- Al finalizar la partida, el programa debe mostrar un mensaje claro indicando el resultado: victoria del jugador humano, victoria de la computadora o empate.
- El juego debe impedir movimientos inválidos y mantener siempre la **consistencia del estado del tablero**.

Funcionalidad opcional

- Las funcionalidades descritas anteriormente constituyen los **REQUISITOS MÍNIMOS OBLIGATORIOS** para que su proyecto sea **aceptado y calificado**.
- **El incumplimiento de cualquiera de estos requisitos implicará una calificación automática de cero (0) para todo el grupo**, sin importar si se han desarrollado funcionalidades adicionales o si la interfaz es atractiva.
- Cumplir con los requisitos mínimos le permitirá aspirar a una nota base de **hasta el 80%**.

¿Cómo alcanzar una mejor calificación?

Este proyecto no solo evalúa conocimientos técnicos, sino también su capacidad para **proponer soluciones innovadoras y aplicar estructuras de datos de forma creativa**.
Por ello, el grupo tiene la libertad de incorporar **funcionalidades adicionales** que enriquezcan la experiencia del juego y demuestren un mayor dominio del contenido del curso.

A continuación, se presentan algunas ideas de **funcionalidades opcionales** que pueden implementar para optar por puntos adicionales:

- **Asistente de jugadas**: sugerencias al usuario humano sobre el movimiento más conveniente, según la evaluación del tablero.
- **Visualización del razonamiento de la IA**: mostrar los tableros intermedios generados por la computadora al aplicar Minimax, junto con los valores de utilidad asociados.
- **Visualización del árbol de decisiones**: representar gráficamente el árbol completo del juego desde el estado inicial hasta su conclusión.
- **Modo de IA vs. IA**: permitir que la computadora juegue contra sí misma.
- **Modo humano vs. humano**: permitir que dos jugadores humanos jueguen uno contra otro desde la misma interfaz.
- **Guardar y reanudar partidas**: permitir almacenar una partida en curso y continuarla posteriormente.
- **Estadísticas de juego**: registrar y mostrar el número de victorias, empates o jugadas realizadas por tipo de jugador.

El grupo **puede proponer cualquier otra funcionalidad adicional** que consideren relevante para los objetivos del curso y que aporte valor a la estructura del juego o a su análisis computacional.

Importante: sonidos no otorgan puntos extra

La inclusión de efectos sonoros durante el juego **no será considerada una funcionalidad válida para puntos adicionales**. Aunque su implementación es bienvenida para enriquecer la interfaz, la reproducción de sonidos en Java es trivial y **no representa un reto significativo** desde la perspectiva de Estructuras de Datos.

Entregables

Entrega parcial (40%): **domingo 03 de agosto 8:00pm**

Su proyecto de NetBeans (enviado como un archivo .zip) debe incluir, por lo menos:

Área	Elementos esperados al 40%	¿Debe funcionar?
Diseño POO	Clases Tablero, Jugador, Juego diseñadas y con métodos implementados.	Sí
Lógica de juego	Funciona correctamente: turnos, colocar ficha, verificar ganador, empate.	Sí
Pruebas básicas	Partida entre dos jugadores humanos por consola, sin errores.	Opcional, pero recomendado
Estructura del proyecto	Archivos organizados con paquetes (modelo, main, etc.).	Sí
Árbol de decisión	Clase NodoArbol diseñada, con atributos y métodos para expandir hijos.	Aún no es necesario que funcione
Minimax (IA)	Solo planeado (comentarios o esqueleto de código).	No
JavaFX (UI)	No es necesario todavía.	No

Entrega final (60%): **lunes 18 de agosto 8:00pm**

- Esta entrega corresponde al proyecto de NetBeans que implementa correctamente la interfaz gráfica final de su proyecto, con –al menos– las funcionalidades mínimas detalladas anteriormente.
- Su entrega deberá incluir además un archivo **.docx**, que es la plantilla que se adjunta a este documento.
- Su proyecto de Netbeans y el archivo .docx deben ser entregados a través del Aula Virtual en un único archivo comprimido .zip.

No se aceptarán:

- **Entregas atrasadas.**
- **Entregas sin reporte.**
- **Entregas que no implementen los requerimientos mínimos.**

Recuerde que este **es un proyecto grupal**, esto significa que usted debe trabajar **con** sus compañeros de grupo para sacar el proyecto adelante. Su grupo deberá reunirse periódicamente para coordinar y discutir acciones relacionadas al proyecto. Dividir funcionalidad y repartir responsabilidades podría no ser la mejor estrategia, sobre todo en las fases tempranas del proyecto.

Presentaciones

El proyecto ya tiene que haber sido entregado en el Aula Virtual según las indicaciones dadas. El **martes 19 de agosto de 13:00 a 15:00** en el laboratorio donde se desarrollas las clases el grupo tendrá **15 MINUTOS** para efectuar la respectiva presentación, las observaciones sobre la entrega serán dadas de manera verbal.

Solo los grupos que hayan entregado un proyecto funcional podrán presentarse a esta reunión.

La reunión iniciará con uno de los miembros del grupo mostrando su proyecto en funcionamiento. Esta demostración debe evidenciar que el grupo ha cumplido con los requerimientos mínimos del proyecto y debe mencionar explícitamente toda funcionalidad extra que hayan implementado. **Su grupo debe definir con anticipación qué miembro del grupo estará a cargo de esta presentación inicial.**

Recomendaciones y condiciones para la presentación del proyecto

- La presentación deberá centrarse exclusivamente en **lo que su grupo ha desarrollado**. No repitan ni describan los requerimientos del proyecto, ya que estos son conocidos por el docente.

- Se espera una **demostración funcional del sistema** en ejecución. Muestren con claridad qué partes han implementado, qué problemas enfrentaron y cómo los resolvieron. El foco debe estar en **lo logrado, no en lo planeado**.

- Cada integrante del grupo **debe responder al menos una pregunta técnica** durante la exposición. Estas preguntas evaluarán su comprensión del código, su participación en el desarrollo y su dominio del proyecto. Las respuestas serán calificadas individualmente y representarán uno de los dos **factores multiplicativos** que compondrán la nota final del proyecto.

- Cada estudiante deberá tener abierto el código fuente **en su estación de trabajo**, en la computadora del laboratorio donde se realice la presentación. No se aceptará que un solo miembro manipule el proyecto durante toda la demostración.

- Está **terminantemente prohibido** presentar código que no haya sido desarrollado por el equipo. Se realizarán preguntas orientadas a identificar posibles casos de copia o uso indebido de material externo. Las inconsistencias en las respuestas técnicas o el desconocimiento del funcionamiento del sistema serán tomadas como **evidencia de no autoría** y afectarán gravemente la calificación.

Restricciones adicionales y criterios de evaluación rigurosos

- Los estudiantes deben demostrar **conocimiento profundo** del código que presentan. Se penalizará el desconocimiento de funciones implementadas, estructuras utilizadas o decisiones de diseño tomadas durante el desarrollo.

- El proyecto debe reflejar **criterios mínimos de diseño**, como separación de responsabilidades, claridad en la lógica de control y uso adecuado de estructuras de datos.

- Se evaluará negativamente el uso excesivo o injustificado de código genérico de internet (plantillas, fragmentos no comprendidos, soluciones “copiadas y pegadas” sin adaptación ni entendimiento).

- Se valorará la **capacidad de defender decisiones técnicas** (por qué eligieron determinada estructura de datos, cómo gestionaron la persistencia, cómo validaron sus pruebas, etc.).

Instrucciones Finales

- Recuerde que **esto es un proyecto grupal.**
- **Siga las instrucciones de este documento** para evitar inconvenientes, en caso de tener dudas, consulte al profesor (en lugar de asumir cosas que pueden ser incorrectas). Los ayudantes de la materia podrían no ser la fuente de información más confiable cuando se trata de responder preguntas sobre el proyecto.
- Sea oportuno(a) en cuanto a sus preguntas: **No se atenderán consultas sobre el proyecto después del 04 de agosto del 2025**. Tampoco se responderán preguntas que consulten sobre instrucciones indicadas explícitamente en este documento. Antes de hacer preguntas, **LEA ESTE DOCUMENTO DETENIDAMENTE**.

¡Muchos éxitos a todos!