Dokumentáció

Fodor Dávid – D02DBR

Tic-Tac-Toe game

A feladat:

Tic-Tac-Toe játék, magyar nyelvén Amőba. A játék lényege, hogy két játékos játszik egyszerre az egyik a „O” karakterrel, másik az „X” karakterrel játszik. A játék célja, hogy valamilyen formában, legyen az vertikálisan, horizontálisan vagy diagonálisan el kell helyezni három szimbólumot a sajátodból egymás után, úgy, hogy a másik ne tudja megakadályozni. A játékot az a játékos nyeri, aki ezt a feltételt teljesíteni tudja, tehát elegendő karaktere van egy vonalban elhelyezve megszakítás nélkül. Ha már elfogynak a rendelkezésre álló rubrikák és egyik játékos sem nyerte meg a játékot, a meccs döntetlennel zárul, ebben az esetben a játék újrakezdése javasolt. Természetesen minél kisebb táblán játszott a játék annál könnyebb a döntetlen elérése két racionális játékossal. A játéktábla alapesetben egy 3x3-as négyzetrács, de különböző játékmódjai is ismertek, például az 5x5-ös rács. Ebben a módban a szabályok ugyan úgy érvényesek azzal a különbséggel, hogy itt 4 szimbólum kell egymás után kerüljön a játék megnyeréséhez. Fontos megemlíteni, hogy szabály szerint csak az üres rubrikákra lehet tenni, nem lehet a már lent lévő szimbólumokra újat helyezni.

Felhasználói kézikönyv:

Amikor elindítja a programot, egy menüablak jelenik meg, ahol kiválaszthatja a játék módját és a játéktábla méretét. A „Player vs Player” mód lehetővé teszi két játékos számára, hogy egymás ellen játszanak, míg a „Player vs AI” módban a játékos a számítógépes ellenféllel, vagyis az AI-val versenyezhet. A táblaméret kiválasztásakor két lehetőség közül választhat: a klasszikus 3x3-as táblát, ahol három azonos jel kell a győzelemhez, illetve az 5x5-ös táblát, ahol négy egyforma jel szükséges az egymás melletti mezőkön a nyeréshez. Miután beállította a kívánt módot és táblaméretet, a „Start Game” gombbal elkezdheti a játékot, vagy a „Load Saved Game” gombbal egy korábban mentett játékot tölthet vissza.

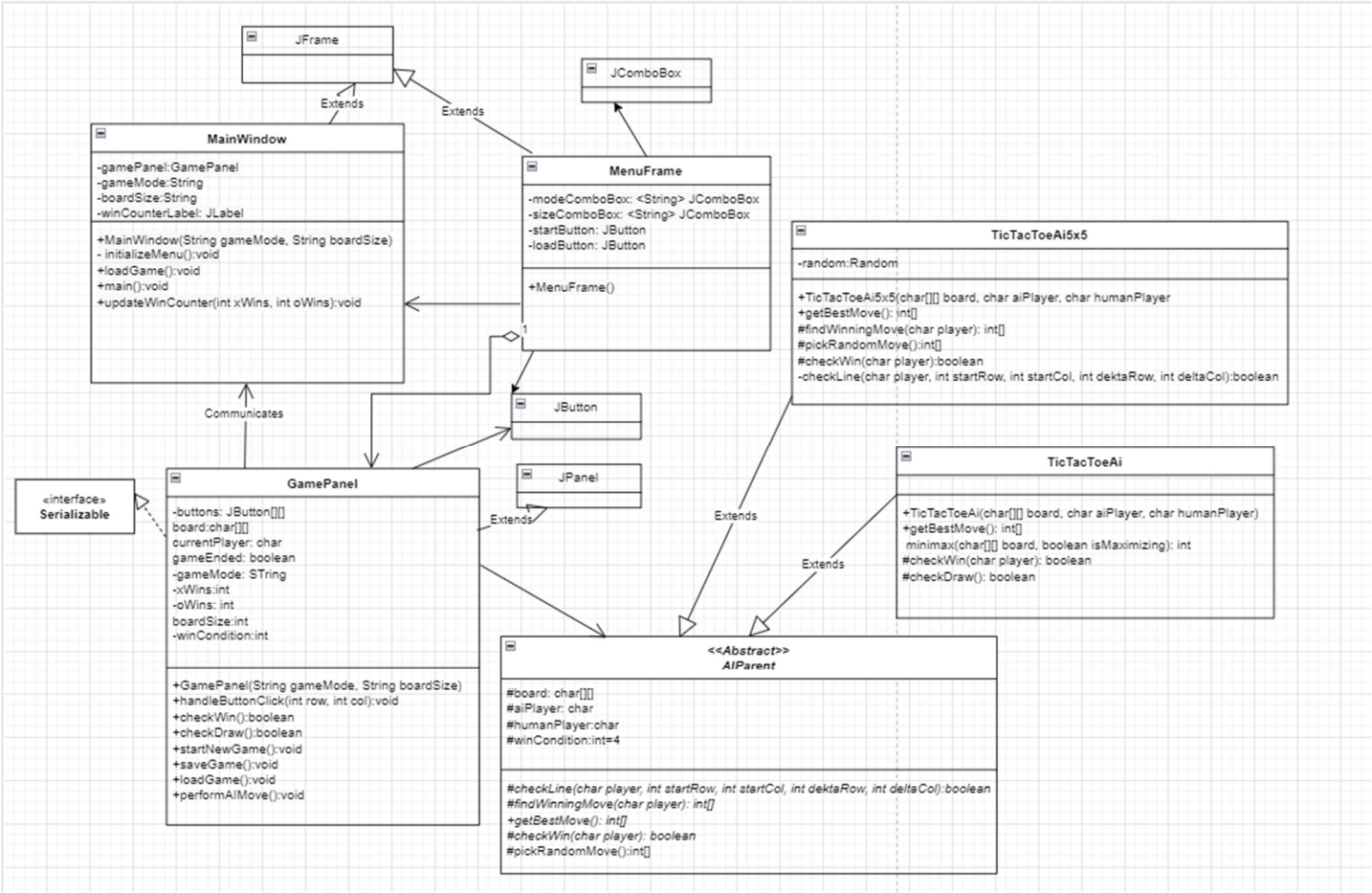
A fő játékablakban látható maga a játéktábla, valamint jobb oldalon néhány vezérlőgomb. Az „End Game” gombra kattintva bezárhatja a játékot, míg a „Restart Game” új játékot indít az aktuális beállításokkal. Az „X Wins” és „O Wins” felirat alatt a program nyilvántartja, hogy hányszor nyertek a játékosok az aktuális játékmenetben.

A játék során a táblán található üres mezőkre kattintva helyezheti el saját jelét (X vagy O) a kiválasztott játékmódnak megfelelően. A „Player vs AI” módban az AI automatikusan végrehajtja a lépését, amikor a játékos befejezte saját körét. Az AI különböző stratégiákat használ, hogy megpróbálja megnyerni a játékot vagy megakadályozni a játékost abban, hogy nyerjen, így változó szintű kihívást jelent a játék során.

A játék mentéséhez bármikor használhatja a „Save Game” gombot. Ez a funkció elmenti a játék aktuális állapotát, amely később a főmenüből újra betölthető a „Load Saved Game” opcióval, így folytathatja ott, ahol abbahagyta. Ez különösen hasznos, ha hosszabb játékot játszik az 5x5-ös táblán, vagy ha a „Player vs AI” módot választotta, és szeretné megfigyelni az AI lépéseit egy hosszabb játék során.

A program célja, hogy könnyen kezelhető és szórakoztató játékélményt biztosítson. A grafikus felület egyszerű és átlátható, így a játékosok gyorsan eligazodhatnak benne. Az üres mezőkre kattintva könnyedén bevihetik saját lépéseiket, az AI pedig automatikusan válaszol, ha a gépi ellenfél opciót választották.

Osztálydiagramm:



Osztályok és metódusaik:

MenuFrame osztály:

- Leírás:
  - A MenuFrame egy induló képernyőt biztosít a játék számára, ahol a felhasználó kiválaszthatja a játékmódot (játékos a játékos ellen vagy játékos a gép ellen) és a tábla méretét (3x3 vagy 5x5). A kiválasztás után a játék elindul a megadott beállításokkal, vagy betölthet egy korábban elmentett játékot. Egyszerű grafikus elemeket tartalmaz, mint a mód és méret kiválasztó menü, valamint indítás és betöltés gombok.
- Attribútumok:
  - **JComboBox<String> modeComboBox**: Legördülő menü a játékmód kiválasztásához, amely „Player vs Player” vagy „Player vs AI” opciókat kínál.
  - **JComboBox<String> sizeComboBox**: Legördülő menü a tábla méretének kiválasztásához (3x3 vagy 5x5).
  - **JButton startButton**: „Start Game” gomb, amely a játék indításáért felelős az aktuális beállításokkal.
  - **JButton loadButton**: „Load Saved Game” gomb, amely betölti egy korábban mentett játékállást.

- Metódusok:
    - **MenuFrame()**: A menü ablak konstruktora, amely beállítja az alapvető ablak tulajdonságokat és hozzáadja a felhasználói felület elemeit a játékmód és a tábla méretének kiválasztásához.

MainWindow osztály:

- Leírás:
    - Ez az osztály a főablakot (játékablak) valósítja meg a játék számára. Egy JFrame típusú ablak, amely a játékpanelt (GamePanel) jeleníti meg a kiválasztott játékmóddal és táblamérettel. Különböző felhasználói felület elemeket tartalmaz, mint a mentés, újraindítás és kilépés gombok, valamint egy nyereményszámláló kijelzőt. Tartalmaz egy menüt is, amely lehetőséget ad új játék indítására, játék mentésére és betöltésére, valamint a program bezárására.
- Attribútumok:
    - **GamePanel gamePanel**: A játéktábla, amely a játék állapotát és az egyes mezők gombjait tartalmazza.
    - **String gameMode**: Az aktuálisan kiválasztott játékmód, amely lehet „Player vs Player” vagy „Player vs AI”.
    - **String boardSize**: Az aktuálisan kiválasztott tábla méret, amely lehet „3x3” vagy „5x5”.
    - **JLabel winCounterLabel**: Felirat, amely a győzelmi számlálót mutatja az X és O játékos számára.
- Metódusok:
    - **MainWindow(String gameMode, String boardSize)**: A fő ablak konstruktora, amely beállítja a játékmódot és a táblaméretet, valamint hozzáadja a vezérlőgombokat és az eredmény számlálót.
    - **initializeMenu()**: A menüsor létrehozása, amely lehetőségeket tartalmaz az új játékra, mentésre, betöltésre és kilépésre.
    - **updateWinCounter(int xWins, int oWins)**: Frissíti az eredményszámláló feliratát az X és O játékos győzelmeinek számával.
    - **loadGame()**: Betölti a mentett játékállapotot a gamePanel segítségével.
    - **main(String[] args)**: A program belépési pontja, amely inicializálja és megjeleníti a MenuFrame menü ablakot.

GamePanel osztály

- Leírás:
    - Ez az osztály felelős a játéktábla kezeléséért és a játék logikájáért. JPanel típusú, és egy kétdimenziós gombmátrixot (JButton[][]) tartalmaz, amelyek a játékmezőket képviselik.A táblaméret és a nyerési feltétel dinamikusan állítható (3x3, ahol 3 egy sorban a győzelem, vagy 5x5, ahol 4 egy sorban). Tartalmazza a játékmenet vezérlését, mint a győzelem és döntetlen ellenőrzése, játékállás mentése és betöltése, valamint az AI mozgásának kezelése.
- Attribútumok:
    - **JButton[][] buttons**: Kétdimenziós tömb, amely a játékmezők gombjait tárolja.
    - **char[][] board**: Kétdimenziós karaktertömb, amely a játéktáblát és az egyes mezők aktuális állapotát tárolja (X, O vagy üres).
    - **char currentPlayer**: Az aktuális játékos karaktere (X vagy O), amely jelzi, hogy melyik játékos következik.
    - **boolean gameEnded**: Logikai érték, amely jelzi, hogy a játék véget ért-e.
    - **String gameMode**: A kiválasztott játékmód (Player vs Player vagy Player vs AI).
    - **int xWins**: Az X játékos győzelmeinek száma.

    - **int oWins**: Az O játékos győzelmeinek száma.
    - **int boardSize**: A tábla mérete, amely lehet 3 vagy 5.
    - **int winCondition**: A nyerési feltétel, amely megadja, hány karakter szükséges egy sorban, oszlopban vagy átlóban a győzelemhez.
- Metódusok:
    - **GamePanel(String gameMode, String boardSize)**: A konstruktor inicializálja a játékpanelt a kiválasztott játékmóddal és tábla mérettel, létrehozva a gombokat és beállítva a nyerési feltételeket.
    - **handleButtonClick(int row, int col)**: Kezeli a mezőkre történő kattintást, frissíti a játéktáblát, ellenőrzi a nyerési és döntetlen feltételeket, majd vált a játékosok között.
    - **checkWin()**: Ellenőrzi, hogy teljesült-e a nyerési feltétel bármely sorban, oszlopban vagy átlóban. Ha igen, akkor a metódus true-t ad vissza.
    - **checkDraw()**: Ellenőrzi, hogy van-e még üres mező a táblán. Ha nincs, akkor a metódus döntetlent jelez (true).
    - **startNewGame()**: Új játék indítása, amely törli a tábla állapotát és alaphelyzetbe állítja a játékot.
    - **saveGame()**: Mentést készít a játék jelenlegi állapotáról egy fájlba. Szerializálással dolgozik
    - **loadGame()**: Betölt egy mentett játékállást egy fájlból, majd frissíti a felhasználói felületet a mentett állapot szerint.
    - **performAIMove()**: Ha a játékmód „Player vs AI”, akkor a számítógépes játékos lépését hajtja végre az AI segítségével.

AIParent osztály (absztrakt)

- Leírás:
    - Ez az osztály definiálja a AI alapvető funkcióit, amelyeket a játék különböző méretű verzióiban lehet alkalmazni. Tartalmazza a ellenfél, illetve a győzelmi feltétel ellenőrzésének alapvető metódusait. Többek között biztosítja a győzelem kereséséhez és véletlenszerű mozdulatok kiválasztásához szükséges alapokat, amelyeket a AI osztályok implementálnak.
- Attribútumok:
    - **char[][] board**: A játéktábla, amely tartalmazza a mezők állapotát.
    - **char aiPlayer**: A karakter, amely az AI játékost jelzi (általában 'O').
    - **char humanPlayer**: A karakter, amely a humán játékost jelzi (általában 'X').
    - **int winCondition**: A nyerési feltétel, amely meghatározza, hány azonos karakter szükséges egy sorban a győzelemhez.
- Metódusok:
    - **AIParent(char[][] board, char aiPlayer, char humanPlayer, int winCondition)**: Konstruktor, amely inicializálja az AI jellemzőit, például a tábla állapotát, az AI és humán játékos karaktereit, valamint a nyerési feltételt.
    - **abstract int[] getBestMove()**: Absztrakt metódus, amely az AI legjobb lépését határozza meg, és visszaadja a lépés sor- és oszlopindexeit. A konkrét AI osztályoknak kell implementálniuk.
    - **protected abstract int[] findWinningMove(char player)**: Absztrakt metódus, amely megkeresi a játékos nyerő lépését, ha van ilyen.
    - **protected abstract boolean checkWin(char player)**: Absztrakt metódus, amely ellenőrzi, hogy egy adott játékos győzött-e.
    - **protected abstract int[] pickRandomMove()**: Absztrakt metódus, amely véletlenszerű lépést választ az AI számára.

- **protected boolean checkLine(char player, int startRow, int startCol, int deltaRow, int deltaCol)**: Egy adott játékosra ellenőrzi, hogy egy adott sorban, oszlopban vagy átlóban teljesül-e a nyerési feltétel.

TicTacToeAI5x5 osztály

- Leírás:
  - A TicTacToeAI5x5 osztály az AIParent osztályból származik, de a 5x5-ös változatban működik, ahol a győzelemhez négy egymás melletti jel szükséges.
- Attribútumok: (AIParent osztályét örökli)
  - **char[][] board**: A játéktábla kétdimenziós tömbje, amely a mezők állapotát tárolja.
  - **char aiPlayer**: Az AI játékos karaktere, amely általában 'O'.
  - **char humanPlayer**: A humán játékos karaktere, amely általában 'X'.
  - **int winCondition**: A győzelmi feltétel, amely 5x5-ös tábla esetén 4 egyező karakter sorban.
- Metódusok:
  - **TicTacToeAI5x5(char[][] board, char aiPlayer, char humanPlayer)**: Konstruktor, amely meghívja az ősosztály konstruktorát, beállítva a táblát, az AI és az ember karaktereit, valamint a győzelmi feltételt.
  - **int[] getBestMove():** Meghatározza az AI számára a legjobb lépést. Először győztes lépést keres, majd blokk lépést, és ha egyik sem elérhető, a tábla középső pozícióját választja, ha az üres. Ha ez sem lehetséges, véletlenszerű lépést tesz.
  - **int[] findWinningMove(char player):** Keres egy olyan lépést, amely az adott player számára győzelmet jelentene. Ha talál egy ilyet, akkor visszatér az adott pozícióval; ha nincs, null-t ad vissza.
  - **boolean checkWin(char player):** Ellenőrzi, hogy az adott player számára teljesül-e a győzelmi feltétel a táblán. Sorokat, oszlopokat és átlókat vizsgál.
  - **int[] pickRandomMove():** Létrehoz egy listát az elérhető üres mezőkről, majd ezek közül véletlenszerűen kiválaszt egyet. Ha nincs elérhető mező, {-1, -1}-et ad vissza.

TicTacToeAI osztály

- Leírás:
  - A TicTacToeAI osztály az AIParent osztályból származik, és a klasszikus 3x3-as Tic-Tac-Toe játék AI logikáját valósítja meg.
- Attribútumok: (AIParent osztályét örökli)
  - **char[][] board**: A játéktábla kétdimenziós tömbje, amely a mezők állapotát tárolja.
  - **char aiPlayer**: Az AI játékos karaktere, amely általában 'O'.
  - **char humanPlayer**: A humán játékos karaktere, amely általában 'X'.
  - **int winCondition**: A győzelmi feltétel, amely 3x3-as tábla esetén 3 egyező karakter sorban.
- Metódusok:
  - **TicTacToeAI(char[][] board, char aiPlayer, char humanPlayer):** Konstruktor, amely meghívja az AIParent osztály konstruktorát, beállítva a táblát, az AI és ember karaktereit, valamint a győzelmi feltételt.
  - **int[] getBestMove():** Meghatározza a legjobb lépést a minimax algoritmus segítségével. Minden üres pozíciót megvizsgál, kiszámítja a lépés pontszámát, majd kiválasztja a legjobbat.
  - **int[] findWinningMove(char player):** Megkeresi azt a lépést, amely a megadott player számára győzelmet hozna. Ha talál egy ilyen lépést, visszaadja annak pozícióját; ha nincs, null-t ad vissza.

- **boolean checkWin(char player):** Ellenőrzi, hogy az adott player számára teljesül-e a győzelmi feltétel a táblán. Megvizsgálja a sorokat, oszlopokat és átlókat.
- **int minimax(char[][] board, boolean isMaximizing):** A minimax algoritmus megvalósítása, amely rekurzívan elemzi a játéktábla állapotát és visszatér az adott állapothoz tartozó pontszámmal.
- **List<int[]>getAvailableMoves():** megvizsgálja a lehetséges lépéseket és visszaadja ezeket
- **boolean checkDraw():** Ellenőrzi, hogy a tábla teljesen megtelt-e üres mezők nélkül, azaz döntetlen-e az állás.
- **int[] pickRandomMove():** Nincs implementálva ebben az osztályban (null-t ad vissza), mivel a minimax algoritmus teljes mértékben kezeli a lépés kiválasztását.

Tesztek:

- TicTacToeAI5x5Test osztály
  - **setUp()**: //nem teszt csak inicializálás
    - Inicializál egy 5x5-ös táblát, és létrehoz egy TicTacToeAI5x5 példányt, ahol az AI 'O' jellel játszik az 'X' emberi játékos ellen.
  - **testGetBestMove()**:
    - Ellenőrzi, hogy az getBestMove() metódus érvényes koordinátákat ad vissza (nem null, és a koordináták nem negatívak).
  - **testFindWinningMove()**:
    - A táblán három 'O' jelet helyez el egy sorban, majd ellenőrzi, hogy az findWinningMove('O') metódus az elérhető nyerő lépést adja-e vissza a negyedik pozícióra.
  - **testPickRandomMove()**:
    - Beállítja a táblát úgy, hogy néhány pozíció már foglalt legyen, majd ellenőrzi, hogy a pickRandomMove() metódus érvényes, üres pozíciót választ.
  - **testCheckWin()**:
    - Négy egymás melletti 'X' jelet helyez el egy sorban, és ellenőrzi, hogy a checkWin('X') metódus igaz értéket ad vissza, jelezve, hogy 'X' nyert.
  - **testNoAvailableMove():**
    - Ellenőrzi, hogy az AI hogyan viselkedik, ha a tábla teljesen tele van ('X' karakterekkel), és nincs többé elérhető lépés.
  - **testBlockOpponentWin():**
    - Vizsgálja, hogy az AI megakadályozza-e, hogy az ember nyerjen.
  - **testWinningMoveDiagonal():**
    - Azt ellenőrzi, hogy az AI felismeri-e a saját győzelmi lehetőségét keresztbe. A findWinningMove metódusnak a győzelemhez szükséges lépést kell megtennie.

- TicTacToeAITest osztály
  - **setUp()**: //nem teszt csak inicializálás
    - Inicializál egy 3x3-as táblát előre meghatározott jelekkel, és létrehoz egy TicTacToeAI példányt, ahol az AI 'O' jellel játszik az 'X' emberi játékos ellen.
  - **testGetBestMove()**:
    - Ellenőrzi, hogy az getBestMove() metódus érvényes koordinátákat ad vissza, amelyek nem null és a táblán belül találhatóak.
  - **testMinimax()**:

        - Az AI minimax algoritmusának pontszámát ellenőrzi úgy, hogy a pontszám a lehetséges értéktartományban legyen (-10 és 10 között).
    - **testPreventOpponentWin ():**
      Ez a teszt az AI képességét vizsgálja az ellenfél győzelmének megakadályozására. Vagyis, hogy ne legyen meg a 4 szimbólum.
    - **testEmptyBoardMove ():**
        - Azt vizsgálja, hogy üres táblán érvényes lépést hajt e végre az AI.
    - **testAIWin ():**
        - Azt ellenőrzi, hogy az AI felismeri-e a saját győzelmi lehetőségét. A findWinningMove metódusnak a győzelemhez szükséges lépést kell megtennie.
- GamePanelTest osztály
    - **setUp()**: //nem teszt csak inicializálás
        - Két GamePanel példányt hoz létre különböző méretekkel: egy 3x3-asat és egy 5x5-öset a „Player vs Player” módban.
    - **testHandleButtonClick()**:
        - Új játékot indít, végrehajt egy lépést az (0, 0) pozícióban, és ellenőrzi, hogy az első lépést 'X' teszi.
    - **testCheckWin()**:
        - A 3x3-as táblán egy sorba három 'X' jelet helyez el, majd ellenőrzi, hogy a checkWin() metódus helyesen felismeri a győzelmet.
    - **testCheckDraw()**:
        - Döntetlent okozó jelekkel tölti fel a 3x3-as táblát, majd ellenőrzi, hogy a checkDraw() metódus döntetlent állapít meg.
    - **testStartNewGame():**
        - Ez a teszt azt vizsgálja, hogy a játék megfelelően újraindul-e. A startNewGame() metódus meghívása után ellenőrzi, hogy: A gameEnded állapot false, vagyis a játék még nem ért véget. A kezdő játékos helyesen az 'X'. Az összes cella üres (0), vagyis a tábla teljesen alapállapotba került.
    - **testWinningRowCondition():**
        - Megnézi, hogy a játék felismeri-e, ha valamelyik játékos nyert
    - **testInvalidMoveHandling():**
        - Ez a teszt azt ellenőrzi, hogy a játék megfelelően kezeli-e az érvénytelen lépéseket. Ugyan arra a mezőre próbál valamit tenni, mint amin már van, nem történik semmi.
    - **testResetGameState():**
        - Ez a teszt a játék állapotának alaphelyzetbe állítását ellenőrzi. Lényegében mint a StartNewGame, csak itt már vannak elemek a táblán és azokat is törli.

- **testStartNewGame()**:
  - Új játékot kezd az 5x5-ös táblán, és ellenőrzi, hogy:
    - A játék nem ért véget (gameEnded hamis).
    - Az aktuális játékos 'X'.
    - A tábla minden mezője üres (0).

Felhasznált technológiák:

- **Nyelv**: Java nyelven íródott
- **IDE**: Eclipse IDE és VSCode
- **GUI Könyvtár**: Swing
- **I/O műveletek**: Szerializálás
- **Tesztelés**: JUnit 5