TEHENEK ÉS LEOPÁRDOK

Dokumentáció

Sándor Balázs – W1SRF8

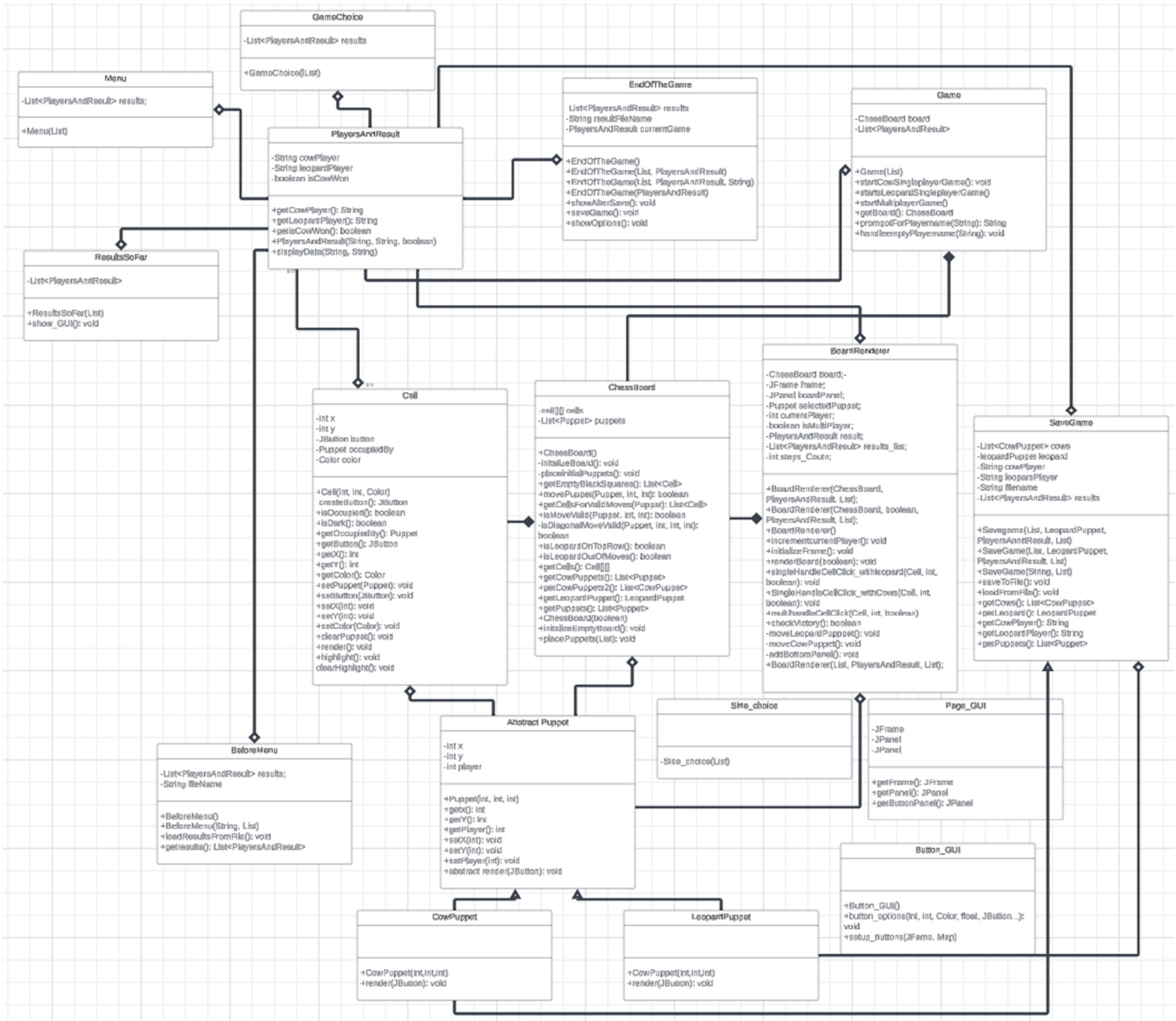
2024.november 26.

Tartalom

1.Bevezetés:

A Tehenek és Leopárdok játékot megvalósító osztályokat és azok kapcsolatát, metódusaikat mutatom be ebben a dokumentációban. A rendszer képes tárolni és megjeleníteni a játékosok neveit és a játék eredményeit, lehetővé téve a felhasználók számára, hogy visszajelzéseket kapjanak a játék állapotáról. Az alkalmazás funkcióit Java Swing grafikus felülettel valósítottuk meg, és a fájlkezelést a mentett adatok betöltésére és tárolására használjuk.

2.UML Diagramm:



3.Csomagok – osztályok felépítése:

3.1. Default Package:

Ebben a csomagban található a program main metódusa, amely a szokásos String args[]-ot kapja paraméterül és void a visszatérési értéke. A függvény létrehoz egy BeforeMenu objektumot, melynek lekéri a List tagváltozóját, ezen kívül pedig, létrehoz egy Menu objektumot melynek át adja a Listet, és ettől kezdve elindul maga a játék.

3.2. Menü Package:

3.2.1.: BeforaMenu Class:

Attribútimok:

- **private List<PlayersAndResult> results:** az összes eddig elmentett játék eredményét tartalmazza (csak azt, h□gy ki-kivel játsz□tt és, h□gy kinyert).
- **private String fileName:** annak a fájlnak a neve, ahol az eredmények tárolva vannak, a konstruktorban ennek az értéke „Results.txt”.

Konstruktor:

- **BeforeMenu():** Beállítja az alapértelmezett fájlnevet, üresre inicializálja a results listát, és meghívja a loadResultsFromFile() függvényt.
- **BeforeMenu(String fileName, List<PlayersAndResult> results):** teszthez szükséges

Metódusok:

- **private void loadResultsFromFile():** megynitja a Results.txt fájlt olvasásra, soronként olvassa be az adatokat, majd a ’;’ karakter alpaján feldarabolja őket, és z adatok alapján egy új PlayersAndResult objektumot hozz létre, amit hozzá ad a resutls listához.
- **public List<PlayersAndResult> getResults():** getter függvény a results attribútumhoz

3.2.2.: Menu Class:

Attribútumok:

- **private List<PlayersAndResult> results:** Az összes eddig elmentett játék eredményét t□rt□lm□zz□, □melyeket □ menü megjelenítéséhez szükséges funkciók h□szná1n□k (pl. eredmények megtekintése).

Konstruktor:

- **Menu(List<PlayersAndResult> results):** beállítja a főmenü ablak (JFrame) tulajdonságait
  - beállítja a főmenü ablak (JFrame) tulajdonságait
  - létrehozza és hozzáadja a címsor panelt
  - létrehoz egy középső panelt, a gomboknak (GAME, RESULTS SO FAR, EXIT)
  - a gombok eseménykezelését egy Map segítségével definiálja, amelyben a gombokhoz műveletek (Runnable) vannak társítva
  - a gombok grafikus tulajdonságait a Button_GUI segédosztály kezeli

Metódusok:

- Az osztály nem definiál nyilvános metódusokat, mivel az osztály fő célja a főmenü létrehozása és megjelenítése az ablak létrehozásakor. A konstruktor tartalmazza a teljes logikát.

3.3. GUI Package:

3.3.1.: Graphics Interface:

Az interface eredeti célja az volt, hogy implementálja, majd valamennyi grafikus felülettel rendelkező osztály azonban ez végül nem következett be, szerepe elenyésző a program felépítésében, azonban van, ahol mégis felhasználásra kerül, ezért nem töröltem.

Metódus:

- **void show_GUI**(): implementáló osztályokban definiálja a grafikus felhasználói felület megjelenítését

3.3.2.: Button_GUI Class:

A Button_GUI osztály segédosztályként szolgál a gombok beállításának és eseménykezelésének egyszerűsítésére a grafikus felhasználói felületeken belül. Két fő funkciója van: a gombok vizuális tulajdonságainak konfigurálása, valamint a gombokhoz események hozzárendelése.

Attribútumok:

- Az osztály nem tartalmaz attribútumokat, mivel minden szükséges adatot a metódusokon keresztül adunk meg.

Konstruktor:

- **Button_GUI():** Alapértelmezett konstruktor, amely inicializálja az osztályt. Nem végez külön műveletet.

Metódus:

- **v□id butt□n_□pti□ns(int height, int width, C□l□r c□l□ur, fl□at c□mp□nent, JButt□n... butt□ns):** Beállítja a megadott gombok vizuális tulajdonságait, váltakozó hosszú paraméter listán, bárhány gombét betudja állítani
- **void setup_buttons(JFrame frame, Map<JButton, Runnable> buttonActions**): több gombhoz rendel eseménykezelőt, amely megadott műveleteket hajt végre a gombok megnyomásakor, a Map paraméter a gombokhoz társított műveleteket tárolja, a kulcsok JButtonok objektumok, az értékek pedig a végrehajtandó műveletek. Hozzáad minden megadott gombhoz egy eseménykezelőt, amely a hozzá tartozó Runnable objektumot futtatja.

3.3.3.: pageGUI Class:

Az A pageGUI osztály segédosztályként szolgál a grafikus felhasználói felületek (GUI) alapstruktúrájának létrehozására. osztály egy ablakot (JFrame), valamint két alapvető panelt hoz létre: egy címképernyőt (titlePanel) és egy gombok számára fenntartott panelt (buttonPanel). Ezeket a komponenseket egységesen kezeli, és getter metódusokat biztosít az elérésükhöz.

Attribútumok:

- **private JFrame frame:** az osztály által létrehozott főablak, amely a GUI szerkezetének alapjául szolgál
- **private JPanel titlePanel:** az ablak tetején elhelyezkedő panel, amely a title megjelenítésére szolgál
- **private JPanel buttonPanel:** az ablak középső részén található panel, amely gombok és egyéb vezérlők elhelyezésére alkalmas

Konstruktor:

- **pageGUI(String s1, String s2):** létrehozza a GUI alapstruktúráját, beállítja az ablak tulajdonságait, és inicializálja a paneleket, s1 megadja az ablak címét, s2 pedig a címpanelen megjelenő szöveget.
- **JFrame getFrame():** visszaadja az osztály által létrehozott JFrame objektumot.
- **JPanel getPanel():** visszaadja a címképernyőt (titlePanel).
- **JPanel getButtonPanel():** visszaadja a gombok elhelyezésére szolgáló panelt (buttonPanel).

3.4. Results Package:

3.4.1.: ResultsSoFar Class:

A ResultsSoFar osztály a játék során mentett eredmények megjelenítésére szolgál grafikus felhasználói felületen keresztül. Az osztály a korábbi eredményeket listázza, és két fő lehetőséget biztosít: visszatérés a főmenübe vagy az alkalmazás bezárása. Az osztály a Graphics interfészt valósítja meg, amely megköveteli a show_GUI metódus implementálását.

Attribútumok:

- **private List<PlayersAndResult> results:** a korábban mentett eredményeket tartalmazó lista.

Konstruktor:

- **ResultsSoFar(List<PlayersAndResult> results):** Inicializálja az osztályt a megadott eredményekkel, majd meghívja a GUI megjelenítéséért felelős metódust.

Metódus:

- **void show_GUI():** az osztály grafikus felületét jeleníti meg. Ez a metódus a Graphics interfész által előírt funkció. Létrehozza az ablakot (JFrame) és beállítja annak tulajdonságait. Egy panelen megjeleníti az eredményeket, amelyeket az results listából olvas ki. Görgethető nézetet (JScrollPane) használ, hogy sok eredmény esetén is kezelhető legyen a megjelenítés. Hozzáad egy alsó panelt, amely gombokat tartalmaz a visszalépéshez vagy a program bezárásához.

3.5. Puppets Package:

3.5.1.: Puppet Abstract Class:

A Puppet egy absztrakt osztály, amely a játékban részt vevő bábuk közös tulajdonságait és viselkedését definiálja. Ez az osztály szolgál alapul a konkrét bábukhoz, mint például a tehenek és a leopárd. Tartalmazza a bábuk pozíciójának, tulajdonosának kezelésére szolgáló attribútumokat, valamint egy absztrakt metódust a megjelenítéshez.

Attribútumok:

- **protected int x:** a korong x-koordinátája a játéktáblán.
- **protected int y:** a korong y-koordinátája a játéktáblán.
- **protected int player:** a korong tulajdonosának azonosítója (értéke 1 ha tehén, 2 ha leopárd).
-

Konstruktor:

- **public Puppet(int x, int y, int player): inicializálja a báb pozícióját és tulajdonosát.**

Metódusok:

Getterek**:**

- **public int getX():**
- **public int getY():**
- **public int getPlayer():**

Setterek**:**

- **public void setX(int x):**
- **public void setY(int y):**
- **public void setPlayer(int player):**

Absztrakt metódus:

- **public abstract void render(JButton button):** Absztrakt metódus, amely felelős a korongok megjelenítéséért a grafikus felületen. Minden konkrét korong (például tehén vagy leopárd) saját implementációt biztosít ehhez a metódushoz

3.5.2.: CowPuppet Class

A CowPuppet osztály a Puppet absztrakt osztály egy konkrét megvalósítása, amely a játékban szereplő tehén bábokat reprezentálja. Az osztály biztosítja a tehén speciális megjelenítési logikáját a grafikus felületen.

Attribútumok:

- A CowPuppet osztály nem tartalmaz új attribútumokat, azokat a Puppet absztrakt osztályból örökli:

Konstruktor:

- **public CowPuppet(int x, int y, int player):** Inicializálja a tehén korong pozícióját és tulajdonosát a Puppet szülőosztály konstruktorának meghívásával.

Metódusok:

Örökölt metódusok**:**

- A CowPuppet az összes gettert és settert örökli a Puppet osztályból.

Megjelenítés:

- **@Override public void render(JButton targetSquare):** A tehén korong megjelenítéséért felel a grafikus felületen. A megadott JButton négyzetet frissíti úgy, hogy az egy tehén korong megjelenését szimulálja.

3.5.3.: LeopardPuppet Class

A LeopardPuppet osztály a Puppet absztrakt osztály egy konkrét megvalósítása, amely a játékban szereplő leopárd korongokat reprezentálja. Az osztály biztosítja a leopárd speciális megjelenítési logikáját a grafikus felületen.

Attribútumok:

- A LeopardPuppet osztály nem tartalmaz új attribútumokat, azokat a Puppet absztrakt osztályból örökli:

Konstruktor:

- **public LeopardPuppet(int x, int y, int player):** Inicializálja a leopárd korong pozícióját és tulajdonosát a Puppet szülőosztály konstruktorának meghívásával.

Metódusok:

Örökölt metódusok**:**

- A LeopardPuppet az összes gettert és settert örökli a Puppet osztályból.

Megjelenítés:

- **@Override public void render(JButton targetSquare):** A leopárd korong megjelenítéséért felel gr fikus felületen. A meg dott JButton négyzetet frissíti úgy, hogy z egy leopárd korong megjelenését szimulálj.

3.6. Board Package:

3.6.1.: Cell Class:

A Cell osztály a játéktábla egy-egy mezőjét reprezentálja. Ez az osztály kezeli a mező pozícióját, vizuális megjelenését (JButton), színét, és hogy van-e rajta korong. Az osztály felelős a mező állapotának frissítéséért és megjelenítéséért a grafikus felületen.

Attribútumok:

- **private int x:** A cella x-koordinátája a játéktáblán
- **private int y:** A cella y-koordinátája a játéktáblán.
- **private JButton button**: A cellát megjelenítő gomb.
- **private Puppet occupiedBy**: A cellát elfoglaló korong (ha van), vagy null, ha a cella üres.
- **private Color color:** A cella színe (világos vagy sötét)

Konstruktor:

- **public Cell(int x, int y, Color color):** Inicializálja a cella pozícióját, színét, és létrehozza a gombot a megjelenítéshez.

Metódusok:

Privát metódusok**:**

- **private JButton createButton():** Létrehoz és konfigurál egy új JButton objektumot, amely a cella megjelenítéséhez szükséges. Beállítja a méretet, betűstílust, igazítást és az alapértelmezett háttérszínt.

Állapotellenőrzés:

- **public boolean isOccupied():** Ellenőrzi, hogy a cella foglalt-e (occupiedBy nem null).
- **public boolean isDark():** Ellenőrzi, hogy a cella sötét színű-e.

Getterek:

- **public Puppet getOccupiedBy()**: Visszaadja a cellát elfoglaló korongot.
- **public JButton getButton()**: Visszaadja a cellát megjelenítő gombot.
- **public int getX()**: Visszaadja a cella x-koordinátáját.
- **public int getY()**: Visszaadja a cella y-koordinátáját.
- **public Color getColor()**: Visszaadja a cella színét.

Setterek:

- **public void setPuppet(Puppet puppet)**: Beállítja a cellát elfoglaló korongot.
- **public void setButton(JButton button)**: Beállítja a cella gombját.
- **public void setX(int x)**: Beállítja a cella x-koordinátáját.
- **public void setY(int y)**: Beállítja a cella y-koordinátáját.
- **public void setColor(Color color)**: Beállítja a cella színét.

Állapotmódosítás:

- **public void clearPuppet():** Eltávolítja a cellából a korongot (occupiedBy null értékre állítása).
- **public void render():** Frissíti a cella megjelenítését a gombon. Ha van korong a cellán, azt megjeleníti, különben a cella színe alapján frissít.
- **public void highlight():** Kiemeli a cellát sárga háttérszínnel, jelölve, hogy melyik korongot jelöltük ki.
- **public void clearHighlight():** Eltávolítja a kiemelést, visszaállítja a cella alapértelmezett színét. Nem kerül végül felhasználásra, de bent hagytam a programomban**.**

3.6.2.: ChessBoard Class:

A **ChessBoard** osztály a játék logikájának központi eleme, amely a játéktábla állapotát és az azon elhelyezkedő korongok mozgását kezeli. Továbbá lehetővé teszi a játék kezdeti állapotának beállítását, és jelentős szerepet játszik a játékállapotának mentésében és visszatöltésében.

Attribútumok:

- **private Cell[][] cells:** Egy 8x8-as mátrix, amely a játéktábla mezőit reprezentálja. Minden mező egy Cell objektum.
- **private List<Puppet> puppets:** Egy lista, amely a táblán jelenleg elhelyezett korongokat tárolja.

Konstruktorok:

- **public ChessBoard()**:Inicializálja a játéktáblát és elhelyezi a korongokat az alapértelmezett kezdőpozíciókban.
- **public ChessBoard(boolean empty)**: Egy üres tábla inicializálására szolgál, amely mentett játékállapotok importálásakor használható (saved game visszatöltése). Nem helyez el korongokat a táblán.

Metódusok:

Táblával kapcsolatos metódusok:

- **private void initalizeBoard():** A játéktábla alapértelmezett állapotát hozza létre, váltakozó világos és sötét mezőkkel. Meghívja a kezdőkorongok elhelyezéséért felelős metódust is.
- **private void initalizeEmptyBoard()**: Üres tábla inicializálása váltakozó világos és sötét mezőkkel, korongok nélkül.
- **public Cell[][] getCells()**: Visszaadja a táblát reprezentáló Cell mátrixot.

Korongok elhelyezése és mozgatása:

- **private void placeInitilaPuppets()**:Elhelyezi a tehén korongokat a tábla megfelelő mezőire, illetve véletlenszerűen elhelyez egy leopárd korongot egy szabad fekete mezőre.
- **public void placePuppets(List<Puppet> puppetsToPlace)**: Egy adott lista alapján helyezi el a korongokat a táblán, ellenőrizve, hogy a célmezők üresek-e.

Korongok mozgatása:

- **public boolean movePuppet(Puppet puppet, int targetX, int targetY)**: Egy korongot mozgat az aktuális pozíciójáról a megadott célmezőre, ha a lépés érvényes. Frissíti a korong pozícióját és a mezők állapotát.
- **public List<Cell> getCellsForValidMoves(Puppet puppet)**: Visszaadja azokat a mezőket, ahová az adott korong szabályosan léphet.
- **public boolean isMoveValid(Puppet puppet, int targetX, int targetY)**: Ellenőrzi, hogy az adott korong számára a megadott lépés szabályos-e.
- **private boolean isDiagonalMoveValid(Puppet puppet, int targetX, int targetY, int maxSteps)**: Megvizsgálja, hogy az átlós lépés érvényes-e, figyelembe véve a lépéshosszt és az ugrási szabályokat.

Játék végének az ellenőrzése:

- **public boolean isLeopardOnTopRow()**: Ellenőrzi, hogy a leopárd korong elérte-e a tábla legfelső sorát.
- **public boolean isLeopardOutOfMoves()**: Ellenőrzi, hogy a leopárd korong számára van-e érvényes lépés.

Getterek a korongokhoz:

- **public List<Puppet> getPuppets()**: Visszaadja az összes korongot tartalmazó listát.

- **public List<Puppet> getCowPuppets()**: Visszaadja a tehén korongokat általános Puppet típusú listaként.

- **public List<CowPuppet> getCowPuppets2()**: Visszaadja a tehén korongokat CowPuppet típusú listaként.

- **public LeopardPuppet getLeopardPuppet()**: Visszaadja a leopárd korongot, ha az megtalálható a táblán.

3.6.3.: BoardRenderer Class:

A BoardRenderer osztály felelős egy sakkhoz hasonló játék grafikus felületének megjelenítéséért, a játékosok interakcióinak kezeléséért, valamint a játékmenet irányításáért. Az osztály biztosítja a játék grafikus felületét, kezeli az állapotfrissítéseket, és a játékkal kapcsolatos eseményeket. Ez az osztály a program legösszetetteb metódusait foglalja magában.

Attribútumok:

- **ChessBoard board:** A játék tábla, amely cellákból és korongokból áll.
- **JFrame frame**: Az alkalmazás fő ablaka, amely a játékot megjeleníti.
- **JPanel boardPanel:** A tábla grafikus megjelenítéséért felelős panel**.**
- **Puppet selectedPuppet:** Az aktuálisan kiválasztott korong, amelyet a játékos mozgat.
- **int currentPlayer**: Az aktuális játékos száma (0 a teheneknek, 1 a leopárdnak).
- **boolean isMultiPlayer:** Annak jelzése, hogy többjátékos módban fut-e a játék.
- **PlayersAndResult result:** A játékosokat és az eredményt nyomon követő objektum.
- **List<PlayersAndResult> results_list:** A játék korábbi eredményeinek listája.
- **int stepsCount:** A lépések száma a játék során.

Konstruktorok:

- **public BoardRenderer(ChessBoard board, PlayersAndResult result, List<PlayersAndResult> results):** többjátékos módhoz.
- **public BoardRenderer(ChessBoard board, boolean startingWithLeopard, PlayersAndResult result, List<PlayersAndResult> results):** egyjátékos módhoz
- **public BoardRenderer(List<Puppet> puppets, PlayersAndResult result, List<PlayersAndResult> results):** mentett játék folytatása módhoz
- Mindegyik konstruktor inicializálja a tagváltozókat, illetve meghívja az initalizeFrame() és renderBoard(boolean) függvényeket

Metódusok:

Megjelenítés és inicializálás:

- **initializeFrame():** Létrehozza a fő ablakot, beállítja az elrendezést, és hozzáadja a grafikus elemeket, például a táblát és a vezérlőgombokat.
- **renderBoard(boolean isSaved):** Újrarajzolja a táblát és frissíti az állapotot, z a függvény kzli magát a játékot is, gondoskodik róla, hogy a flhasználó csak érvénysn léphtnk és csak addig tartson a játék, amíg valamlyik fél nm nyr. Ezkt másik függvénykmghívásával éri l.

Játékmnt logikája

- **incrementCurrentPlayer():** Növeli az aktuális játékos számlálóját.
- **singleHandleCellClick_withLeopard(Cell cell, int stepsCount, boolean isSaved)**: Kezeli a cellák kattintását egyjátékos módban, amikor a leopárd a soron következő játékos. Ellenőrzi a lépések érvényességét, frissíti a megjelenítést, és kezeli a tehén AI-t.
- **singleHandleCellClick_withCows(Cell cell, int stepsCount, boolean isSaved):** Kezeli a cellák kattintását egyjátékos módban, amikor a tehén a soron következő játékos. Ellenőrzi a lépések érvényességét, frissíti a megjelenítést, és kezeli a leopárd AI-t.

- **multihandleCellClick(Cell cell, int stepsCount, boolean isSaved):** Kezeli a cellák kattintását többjátékos módban, váltakozva a tehén és a leopárd játékos között.
- **moveLeopardPuppet():** AI által vezérelt lépést hajt végre a leopárd bábura.
- **moveCowPuppet():** Véletlenszerű tehénbábut választ és AI által vezérelt lépést hajt végre.

Győzelem ellenőrzése:

- checkVictory(): Meghatározza, hogy a játéknak vége van-e, azaz olyan állapotban van, hogy valaki nyert-e.

Felhasználói felület elemei:

- addBottomPanel(): Hozzáad egy alsó panelt az alábbi gombokkal: Menu, Restart, Save, Exit.

3.7. Game Package:

3.7.1.PlayersAndResult Class

A PlayersAndResult osztály a játékosok adatait és a játék eredményét tárolja. Ez az osztály felelős a játékosok nevének, valamint a győztes játékos azonosításának kezeléséért. Ezenkívül lehetőséget nyújt az eredmények megjelenítésére konzolon vagy fájlba való mentésére.

Attribútumok:

- **String cowPlayer: A tehén játékos neve.**
  String leopardPlayer: A leopárd játékos neve.
- **boolean isCowWon: Azt jelzi, hogy a tehenek nyertek-e (true esetén igen, false esetén a leopárd nyert).**

Konstruktor:

- **PlayersAndResult(String cowPlayer, String leopardPlayer, boolean isCowWon):** Létrehozza az osztály egy példányát a játékosok nevével és a győztes meghatározásával. Inicializálja az osztály attribútumait az átadott értékekkel.

Metódusok:

Getterek**:**

- **String getCowPlayer():** Visszaadja a tehén játékos nevét.
- **String getLeopardPlayer():** Visszaadja a leopárd játékos nevét.
- **boolean getisCowWon():** Visszaadja, hogy a tehenek nyertek-e.

Setter:

- **void setisCowWon(boolean isCowWon):** Beállítja, hogy a tehenek nyertek-e.

Megjelenítés:

- **void displayData(String output, String filename**): Megjeleníti vagy elmenti a játékosok adatait és az eredményt. Az output értéke „console” vagy „file”, ettől függ, hogy hova íródik ki az adat.

3.7.2 GameChoice Class

A GameChoice osztály a felhasználó számára biztosít lehetőséget arra, hogy válasszon a játék típusai közül, mint például "Singleplayer", "Multiplayer" vagy "Continue Saved Game". Az osztály a grafikus felhasználói felületet (GUI) biztosít, amely lehetővé teszi a játékosok számára, hogy az egyszerűen kiválaszthatják a kívánt játékformátumot.

Attribútumok:

- **List<PlayersAndResult> results:** A játékosok és az eredmények tárolására szolgáló lista. Ezen keresztül az osztály elérheti az eddigi játékok eredményeit, például a mentett játékok betöltésére. Igazából csak továbbadja, másik függvénynek.

Konstruktor:

- **GameChoice(List<PlayersAndResult> results):** Inicializálja az ablakot (JFrame), amely a játékmód választó felületet tartalmazza.Beállítja a cím panelt, amely tartalmazza a játék típ□sválasztást: "Choose a Game Type". Gombokat hoz létre, amelyek lehetőséget adnak a felhasználónak a különböző játékmódok közötti választásra. Eseménykezelők rendelése a gombokhoz, hogy végrehajtják a megfelelő akciókat a választás □tán.

3.7.3 Side_choice Class

A Side_choice osztály lehetőséget biztosít a felhasználóknak arra, hogy válasszanak egy oldalt a játékban (tehén vagy leopárd) egyjátékos módban. A felhasználó az oldalak közül választhat, és az osztály elindítja az adott játékmódot. A grafik□s felhasználói felület biztosítja az interakciót és a gombok kezelését.

Konstruktor:

- **Side_choice(List<PlayersAndResult> results):** Létrehozza a Side_choice osztály egy példányát és inicializálja a grafikus felhasználói felületet (GUI).Az ablak tartalmazza a két gombot, amelyek közül a felhasználó választhat: "Play with cows" és "Play with the leopard". Emellett a felhasználó választhat vissza gombot a játék típusának kiválasztásához és kiléphet az alkalmazásból. Minden gombhoz egy eseménykezelőt rendel, amely az adott választásnak megfelelő akciót hajtja végre.

3.7.4 SaveGame Class

A SaveGame osztály a játék állapotának mentésére és betöltésére szolgál. Ez az osztály lehetővé teszi a játékosok, a tehenek és a leopárd állapotának fájlba való mentését, valamint a korábban mentett játék állapotának visszaállítását. A játék állapotát egy szöveges fájlban tárolja, amelyet később újra betölthetünk, hogy folytathassuk a játékot.

Attribútumok:

- **List<CowPuppet> cows**: A teheneket reprezentáló bábuk listája. Minden bábu tartalmazza annak pozícióját (X, Y koordináták) és a hozzá tartozó játékost.
- **LeopardPuppet leopard**: A leopárd bábuját reprezentáló objektum, amely tartalmazza a leopárd pozícióját és a hozzá tartozó játékost.
- **String cowPlayer**: A tehén játékos nevét tartalmazza.
- **String leopardPlayer**: A leopárd játékos nevét tartalmazza.
- **String filename**: A fájl neve, amelyben a játék állapotát mentjük vagy onnan betöltjük (alapértelmezett: SavedGame.txt).

Konstruktorok:

- **SaveGame(List<CowPuppet> cows, LeopardPuppet leopard, PlayersAndResult players):** Inicializálja az osztály attribútumait az átadott értékek alapján. Beállítja az alapértelmezett fájlnevet SavedGame.txt-re.
- **SaveGame(String filename):** A fájlnevet beállítja az osztályban. Inicializálja az üres tehenek listáját (cows), majd meghívja a loadFromFile() metódust a fájlban tárolt adatok betöltésére.
- **SaveGame(List<CowPuppet> cows, LeopardPuppet leopard, PlayersAndResult players, String fileName):** teszthez szükséges

Metódusok:

File kezeléshez:

- **saveToFile():** Az aktuális játék állapotát fájlba menti (SavedGame.txt vagy a megadott fájlneve). Minden tehenet (legfeljebb négyet) ment, beleértve azok koordinátáit (X, Y) és a hozzájuk tartozó játékost. A leopárdot is elmenti, a koordinátáival és a játékosával együtt. Végül elmenti a tehén és a leopárd játékosainak nevét.
- **loadFromFile():** Betölti a játék állapotát a fájlból (például SavedGame.txt). Beolvassa az első négy sort, amelyek a tehenek pozícióit és játékosait tartalmazzák. Ha van mentett leopárd, akkor annak pozícióját és játékosát is betölti (egyébként úgy működik a játék, hogy mindenképpen van mentett leopárd, ha van mentett játék). Végül betölti a tehén és leopárd játékosainak neveit

Getterek:

- **getCows()**: Visszaadja a tehén bábuk listáját.
- **getLeopard()**: Visszaadja a leopárd bábut.
- **getCowPlayer()**: Visszaadja a tehén játékos nevét.
- **getLeopardPlayer()**: Visszaadja a leopárd játékos nevét.
- **getPuppets()**: Visszaadja a játékban lévő összes bábút (teheneket és a leopárdot).
-

3.7.5 Game Class

A Game osztály a játék logikáját kezeli, beleértve a különböző játékmódok (egyjátékos és többjátékos) elindítását. A játékosok adatait (név, győztes) és a játéktáblát tartalmazza, és lehetőséget ad arra, hogy a játékosok különböző szerepeket válasszanak (tehén vagy leopárd) egyjátékos mód esetén, illetve két játékos esetén többjátékos módot biztosít. Az osztály felelős a megfelelő mód elindításáért és a játéktábla megjelenítéséért.

Attribútumok:

- **ChessBoard board**: A játéktábla, amely a játék aktuális állapotát tartalmazza, beleértve a korongok elhelyezkedését.
- **List<PlayersAndResult> results**: A játékosok adatait és az eredményeket tartalmazó lista, amely segít nyomon követni a játékmenetet és az eredményeket.

Konstruktor:

- **Game(List<PlayersAndResult> results):** Inicializálja a results listát és létrehozza az üres ChessBoard objektumot.

Metódusok:

- **startCowSingleplayerGame**(): Elindít egy egyjátékos módot, ahol a játékos a tehén oldalon játszik, míg a leopárd szerepét az AI (mesterséges intelligencia) tölti be. Kéri a játékostól a nevét, majd a PlayersAndResult objektumot hozza létre a megadott névvel és az AI leopárd játékosával. A játékos neve és a győztes eredmény kiírásra kerül a konzolra. A BoardRenderer segítségével megjeleníti a játéktáblát. A konzolra írás csak fejlesztői segítség.
- **startLeopardSingleplayerGame(): Elindít egy egyjátékos módot, ahol a játékos a leopárd oldalon játszik, míg a tehén szerepét az AI tölti be. Kéri a játékostól a nevét, majd a** PlayersAndResult **objektumot hozza létre az AI tehén játékosával és a megadott leopárd játékossal. A játékos neve és a győztes eredmény kiírásra kerül a konzolra. A** BoardRenderer **segítségével megjeleníti a játéktáblát.**
- **startMultiplayerGame():** Elindít egy többjátékos módot, ahol két ember játszik egymás ellen, egy tehén és egy leopárd szerepében. Kéri mindkét játékostól a nevét, majd a PlayersAndResult objektumot hozza létre a két játékos nevével. A játékosok nevei és a győztes eredmény kiírásra kerül a konzolra. A BoardRenderer segítségével megjeleníti a játéktáblát.
- **getBoard():** Visszaadja a játéktáblát (ChessBoard), amely a játék aktuális állapotát tartalmazza.
- **promptForPlayerName(String whichName):** Megjelenít egy párbeszédpanelt, amely kéri a játékostól a nevét a megadott szerephez. Ha a játékos megadja a nevét és rákattint az OK gombra, akkor azt visszaadja. Ha a játékos lemond, null-t ad vissza.
- **handleEmptyPlayerName(String playerName):** visszalép a programban a megfelelő helyre, abban az eseteben ha felhasználó érvénytelen nevet / neveket ad meg.

3.7.6 EndOfTheGame Osztály Dokumentáció:

Az EndOfTheGame osztály a játék végén megjelenő választási lehetőségeket és a játék eredményeinek mentését kezeli. Az osztály biztosítja a felhasználói interakciókat, például azt, hogy a játékos mentse-e az eredményt, visszatérjen a menübe, vagy kilépjen az alkalmazásból. Az eredmények fájlba mentésére is képes, és az új játék eredményeit a fájlba írja.

Attribútumok

- **List<PlayersAndResult> results**: A játék összes eddigi eredményeit tartalmazó lista. Minden egyes játék eredményét PlayersAndResult objektumok reprezentálják.
- **String resultsFilename**: A fájl neve, amelybe a játék eredményeit mentjük. Alapértelmezett értéke: "Res□lts.txt".
- **PlayersAndResult currentGame**: A játék aktuális eredménye. Az osztály ezt az adatot használja a mentés előtt.

Konstruktorok:

- **EndOfTheGame():** Inicializálja az results listát, és beállítja az alapértelmezett fájlnevet ("Results.txt"). Meghívja a showOptions() metódust a játék vége után megjelenítendő opciókhoz.
- **EndOfTheGame(List<PlayersAndResult> results, PlayersAndResult currentgame):** Inicializálja az results listát a megadott adatokkal, beállítja az alapértelmezett fájlnevet, és a showOptions() metódust hívja meg a választási lehetőségek megjelenítéséhez.
- **EndOfTheGame(PlayersAndResult currentgame):** Inicializálja az results listát üresen, beállítja az alapértelmezett fájlnevet, és a showOptions() metódust hívja meg.
- **EndOfTheGame(List<PlayersAndResult> results, PlayersAndResult currentgame, String fileName):** teszthez szükséges

Metódusok:

- **showAfterSave():** A játék mentése után egy új GUI ablakot jelenít meg, amely két lehetőséget kínál: visszatérés a főmenübe vagy kilépés a játékból. Ha a játékos a "Back to menu" gombra kattint, a főmenü (Menu) jelenik meg. Ha a játékos az "Exit" gombra kattint, a játék leáll.
- **saveGame():** A currentGame objektumot menti az eredmények fájlba (resultsFilename). Mielőtt mentené, ellenőrzi, hogy létezik-e a currentGame. Ha nem, hibaüzenetet ad ki. Az eredményeket a fájlba a következő formátumban írja CowPlayerName;LeopardPlayerName;CowWon(0 or 1);.
- **showOptions():** A játék vége után egy GUI ablakot jelenít meg, amely három lehetőséget kínál a játékosnak: „Save the result”. „Back to menu”, „Exit”. Az ablak gombjainak kattintására meghívja a megfelelő metódust: menti az eredményt, visszatér a menübe, vagy kilép a játékból.

4.Felhasználói dokumentáció

A játék használata a felhasználó számára igen egyszerű és szinte minden esetben egyértelmű. A program indítása után a felhasználó a menüben találja magát, ahol lehetősége van választani, három opció közül:

1. Game
2. Result So Far
3. Exit

Az *Exit* szerepe egyértelmű, csak úgy, mint a program későbbi szakaszain előjövő Back szerepe is ezeket nem részletezem tovább (back hatására mindig egyel visszább lép a program). A *Result So Far* lenyomása után a felhasználó megtekintheti az eddigi mentett játékokok résztvevőit és azok kimenetelét. A Game hatására egy új oldalon találja magát, ahol, a következő opciók közül választhat:

1. Singleplayer
2. Multiplayer
3. Continue Saved Game
4. Back
5. Exit

A *Continue Saved Game* hatására, elindul az elmentett játék, a *Multiplayer* hatására pedig elindul egy ilyen típusú menet. A *Singleplayer* hatására megint ez új ablak jelenik, meg:

1. Play with cows
2. Play with the leopard
3. Back
4. Exit

A felhasználó itt a közül tud választani, hogy melyik féllel szeretne lenni. Miután ezt kiválasztotta a játékos elindul a játék, ekkor meg kell adni az egy (multi esetén kettő) felhasználónevet, amennyiben ez nem történik meg vagy semmit nem adnak meg felhasználó névnek a program vissza ugrik a játék választáshoz. *Singleplayer* esetben a program a felhasználót csak a választott fél valamelyik korongjával engedi lépni. A kiválasztás úgy történik, hogy a felhasználónak rá kell kattintani, egy olyan mezőre, ahol a neki megfelelő korong található, majd ezután egy olyan mezőre, ahova lépni szeretne és ezt meg is teheti. A program addig nem megy tovább ó, amíg helyesen meg nem történik mind egy korong kiválasztása, mind annak a mezőnek a helyes kiválasztása, ahova lépni szeretne. Multiplaye esetben is igaz mindez, csak akkor mindegy egyes lépést egy külső felhasználónak kell elvégezni. Játék közben a felhasználónak lehetősége van a sakktábla alján található gombok lenyomásával: kilépni, a *menube* lépni vagy menteni, a mentés a *Save* gomb lenyomása után automatikuson megtörténik ekkor már ki lehet lépni nyugodtan.
A játék végeztével (mután valamelyik fél nyert) a felhasználónak lehetősége van a következőkre:

1. Save the result
2. Back to Menu
3. Exitű

A *Save the result* lenyomásával el tudjam menteni a játék kimenetét és meg tudja nézni a későbbiekben a *Result So* Far menü pont alatt.
Ezeken kívül az ablak szélén kattintva a felhasználónak lehetősége van még az ablak méretének változtatására, illetve a program azonnal leállítására