# Sakk

A sakk egy két játékos által játszott stratégiai táblajáték, amely egy 8x8-as négyzetrácsos táblán zajlik. A játék célja az ellenfél királyának „sakk-matt” állapotba juttatása, ami azt jelenti, hogy a király nem tudja elkerülni a megölést.

**Alapvető szabályok:**

1. **Bábuk**:
    - **Király**: Egy négyzetet léphet bármely irányban. Ha sakkban van, lépnie kell, hogy elkerülje a mattot.
    - **Vezér**: Bármely irányban tetszőleges számú négyzetet léphet.
    - **Bástya**: Függőlegesen és vízszintesen léphet, bármennyi négyzetet.
    - **Futó**: Átlósan léphet, szintén bármennyi négyzetet.
    - **Ló**: L-alakban (két négyzet egy irányban, majd egy négyzet merőlegesen) léphet, és ugrik más bábuk felett.
    - **Gyalog**: Előre lép egy négyzetet, de a kezdőlépésnél kettőt is léphet. Átlósan üthet.
2. **Sakk és Matt**:
    - **Sakk**: Ha a király támadás alatt áll, azonnal lépnie kell, hogy megszabaduljon a fenyegetéstől.
    - **Matt**: Ha a király sakkban van, és nincs legális lépés, akkor a játék véget ér.
3. **Játék menete**:
    - A játékosok felváltva lépnek.
    - Ha egy bábu elér egy ellenfél bábuját, azt eltávolítják a tábláról.
4. **Különleges lépések**:
    - **Sánc**: A király és a bástya egyidejű mozgása, amikor a király két négyzetet lép az egyik bástya felé, a bástya pedig a király mellett áll.
    - **En passant**: Gyalog a másik gyalog mellett léphet el, ha az előző lépésben kettőt lépett.
    - **Átalakulás**: Ha egy gyalog eléri az ellenfél alapvonalát, átalakulhat bármilyen bábbá (általában vezérré).
5. **Játék vége**:

A játék véget ér, ha az egyik játékos mattolja a másikat, (patt) döntetlent játszanak, vagy ha egy játékos feladja.

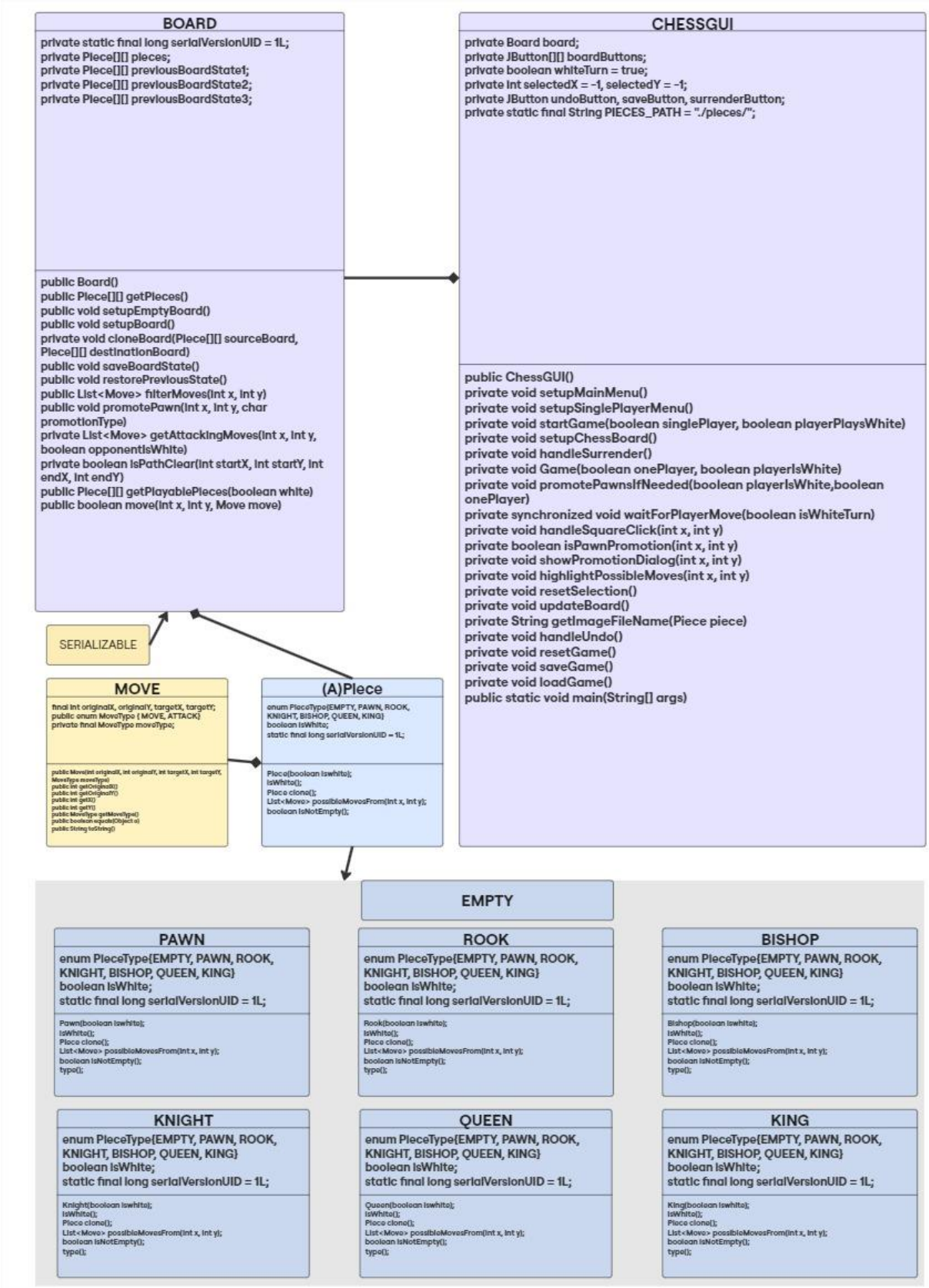
**Feladat rövid szöveges ismertetése**

A feladat célja egy sakkjáték megvalósítása Java nyelven, amely Swing GUI-t használ. A játék támogatja a két személyes és egy személyes játékmódot, ahol a felhasználók a gép ellen játszhatnak, a gép csak a valid lépések közül random választ egyet. Az alkalmazás felhasználói felülete menüvel rendelkezik, amely lehetővé teszi a játék indítását, leállítását, a játék állásának mentését és betöltését, valamint statisztikák megjelenítését. A játék során a bábuk mozgatása csak helyettesítésen keresztül történik, amely alacsony szintű grafikai rutinok segítségével valósul meg. A játékállás fájlba írása és olvasása standard I/O módszerekkel történik.

**Use-case-ek**

1. **Két személyes játék**
    - **Felhasználói akció**: A játékosok felváltva lépnek, kiválasztják a bábukat a grafikus felületen, majd lépnek a mutatott valid lépések közül.
    - **Rendszer válasz**: A rendszer frissíti a táblát, és megjeleníti az aktuális állást, a játék végén a menübe kerül vissza.
2. **Egy személyes játék**
    - **Felhasználói akció**: A felhasználó a program ellen játszik, hasonlóan mint amikor ketten, a játék végén a menübe kerül vissza.
    - **Rendszer válasz**:A program a lehetséges lépések alapján random lép.
3. **Lépés visszavonása**
    - **Felhasználói akció**: A játékos a „undo” gombra kattint.
    - **Rendszer válasz**: A rendszer visszaállítja a játék állását az előző lépésre, ez csak egy adott mennyiségű lépést jelent.
4. **Játék mentése és betöltése**
    - **Felhasználói akció**: A felhasználó elmenti a játék állását vagy betölti az előzőleg elmentett állást a menüből.
    - **Rendszer válasz**: A rendszer serizálási szabály(txt-be) szerint menti le vagy tölti be a játék állását.
5. **Játék feladása**
    - **Felhasználói akció**: A játékos a „surrender” gombra kattint.
    - **Rendszer válasz**: Az ellenség nyer ezután visszakerülünk a főmenübe

**A program megvósításához használt osztályok UML-diagrammja**



**Piece (Abstract Class) implements Serializable**

**Leírás:**

A Piece osztály a sakkbábuk közös tulajdonságait és viselkedéseit definiálja. Ez az alapja az összes konkrét sakkfigura (pl. király, vezér, gyalog) megvalósításának. Serializable-t használ a fájlba mentéshez.

**Attribútumok:**

- boolean isWhite
  Jelzi, hogy a figura fehér-e (true) vagy fekete (false).

**Fontos metódusok:**

1. **isWhite()**
   Visszaadja, hogy a bábu fehér-e.
2. **type()**
   Absztrakt metódus, amely visszaadja a bábu típusát (PieceType enum).
3. **possibleMovesFrom(int x, int y)**
   Absztrakt metódus, amely megadja a bábu összes lehetséges lépését az aktuális pozíciójából, a tábla többi bábújától függetlenül.
4. **clone()**
   Másolatot készít a bábuból (deep copy).
5. **isNotEmpty()**
   Visszaadja, hogy a bábu nem üres-e (hasznos az Empty reprezentálásakor).

---

**Konkrét alosztályok:**

**1. King (Király)**

- **Leírás:**
  A király a legfontosabb figura, amelyet sakkban kell tartani és védeni. A király egy mezőt léphet bármely irányba (vízszintesen, függőlegesen vagy átlósan).
- **Metódusok:**
  - possibleMovesFrom(int x, int y)
    Megadja az összes olyan mezőt, amelyre a király léphet.

**2. Queen (Vezér)**

- **Leírás:**
  A vezér a legerősebb bábu, amely vízszintesen, függőlegesen és átlósan is bármennyi mezőt léphet, amíg nincs akadály.
- **Metódusok:**
  - possibleMovesFrom(int x, int y)
    Az összes olyan mezőt visszaadja, amelyre a vezér szabályosan léphet.

**3. Rook (Bástya)**

- **Leírás:**
  A bástya vízszintesen és függőlegesen léphet bármennyi mezőt.
- **Metódusok:**
  - possibleMovesFrom(int x, int y)
    A bástya lehetséges lépéseinek listáját adja vissza.

**4. Bishop (Futó)**

- **Leírás:**
  A futó csak átlósan léphet, és bármennyi mezőt mozdulhat el akadály nélkül.
- **Metódusok:**
  - possibleMovesFrom(int x, int y)
    Visszaadja a futó átlós lépéseinek listáját.

**5. Knight (Huszár)**

- **Leírás:**
  A huszár egyedi mozgással rendelkezik: „L” alakban lép (két mezőt egyik irányba, egyet a másik irányba).
- **Metódusok:**
  - possibleMovesFrom(int x, int y)
    Visszaadja a huszár lehetséges „L” lépéseit.

**6. Pawn (Gyalog)**

- **Leírás:**
  A gyalog egyenes előre léphet (általában egy mezőt, a kezdőpozícióból két mezőt is). Csak átlósan üthet.
- **Metódusok:**
  - possibleMovesFrom(int x, int y)
    Visszaadja a gyalog lehetséges lépéseit, figyelembe véve az ütések és az előrelépések szabályait.

**7. Empty**

- **Leírás:**
  Speciális osztály, amely az üres mezőket reprezentálja.
- **Metódusok:**
  - isNotEmpty()
    Mindig false-t ad vissza, jelezve, hogy az adott mező üres.
  - type()
    Speciális PieceType.EMPTY típus.

## Board class

A Board osztály a sakktábla működését reprezentálja, beleértve a bábuk elhelyezkedését, a lépések kezelését, a játékállapot mentését/visszaállítását, valamint az ellenőrzések elvégzését. Az osztály tartalmaz privát attribútumokat a bábuállapotok tárolására és számos metódust a sakkjáték logikájának megvalósításához.

### Attribútumok

- Piece[][] pieces: A sakktábla aktuális állapota, ezek mind mátrixok a bábúkkal.
- Piece[][] previousBoardState1: A játékos lépése előtti állapot.
- Piece[][] previousBoardState2: Az AI lépése előtti állapot.
- Piece[][] previousBoardState3: Mindkét lépés előtti állapot (egy körrel korábbi).

### Konstruktor

- **Board()**: Inicializálja a táblát és az előző állapotokat üres 8x8-as mátrixokkal.

### Metódusok

#### Táblakezelés

- **Piece[][] getPieces()**
  Visszaadja az aktuális tábla állapotát.
- **void setupEmptyBoard()**
  Inicializálja a táblát, minden mezőt üresnek (Empty) állítva, tesztekhez.
- **void setupBoard()**
  Beállítja a sakktábla kezdőállását a sakk szabályainak megfelelően (bábuk elhelyezése).

#### Állapotkezelés

- **void cloneBoard(Piece[][] sourceBoard, Piece[][] destinationBoard)**
  Lemásolja a forrás tábla állapotát a cél táblába.
- **void saveBoardState()**
  Elmenti az aktuális táblaállapotot, az előző állapotokat megfelelően eltolva (previousBoardState1, previousBoardState2, previousBoardState3).

- **void restorePreviousState()**
  Visszaállítja a tábla állapotát az utolsó mentett állapotra, miközben az előző állapotokat frissíti.

**Lépések kezelése**

- **List<Move> filterMoves(int x, int y)**
  Megszűri az adott bábu lehetséges lépéseit, hogy csak érvényes mozgásokat tartalmazzon, mindent számításba vesz és ennek a kiszámításához minden lehetséges lépést megnéz és utánna választja ki a jókat.
- **boolean move(int x, int y, Move move)**
  Megkísérli az adott lépést végrehajtani, ellenőrzi annak érvényességét az előző függvénnyel, majd frissíti a táblaállapotot. Visszaadja, hogy a lépés sikeres volt-e.
- **Move.MoveType getMoveType(int startX, int startY, int endX, int endY)**
  Meghatározza, hogy az adott lépés egy támadás vagy egy sima mozgás.

**Ellenőrzések és speciális esetek**

- **boolean isCheckmate(boolean whiteTurn)**
  Ellenőrzi, hogy az aktuális játékos (a paraméter szerint) sakkmattban van-e. Ha a király sakkban van, és nincs érvényes lépése, sakkmatt áll fenn.
- **void promotePawn(int x, int y, char promotionType)**
  Előlépteti a gyalogot a kívánt bábu típusra (Q, R, B, K).
- **List<Move> getAttackingMoves(int x, int y, boolean opponentIsWhite)**
  Visszaadja az összes olyan lépést, amely az adott mezőt támadja, figyelembe véve az ellenfél bábuinak színét, a filterMoves felhasználja ezt.
- **boolean isPathClear(int startX, int startY, int endX, int endY)**
  Ellenőrzi, hogy egy adott mozgási útvonal akadálymentes-e (pl. futó, bástya, vezér mozgásához).

**AI és játékkezelés**

- **void makeComputerMove(boolean isWhite)**
  Az AI végrehajt egy véletlenszerű, de érvényes lépést a saját bábui közül.
- **Piece[][] getPlayablePieces(boolean white)**
  Visszaadja az aktuális játékos színének megfelelő bábukat és azok pozícióit.

Ez a ChessGUI osztály egy sakkjáték GUI (grafikus felhasználói felület), amelyet a Java Swing keretrendszerrel hoztam létre. Az osztály számos funkcióval és lehetőséggel rendelkezik, hogy egy teljes sakkjátékot kínáljon egy- vagy többjátékos módban.

---

**Főbb Jellemzők**

1. **Főmenü:**
    - A játék indításakor egy főmenü jelenik meg, amely lehetőséget ad a következőkre:
        - Egyjátékos mód indítása (választható, hogy fehér vagy fekete színnel játszol).
        - Kétjátékos "hotseat" mód, ahol a játékosok felváltva léphetnek.
        - Játék betöltése (mentett állapot visszatöltése).
        - Kilépés a programból.
2. **Sakkfigura-megjelenítés:**
    - Egy 8x8-as sakkmező gombokból áll, amelyek a sakkbábukat ikonokkal jelzik.
    - A bábuk képeit a PIECES_PATH útvonalon lévő fájlokból tölti be, amelyek a figurák típusának és színének megfelelően vannak elnevezve (pl. "white-queen.png").
3. **Játékmódok:**
    - **Egyjátékos mód:** A játékos a mesterséges intelligencia (AI) ellen játszik.
    - **Kétjátékos mód:** A játékosok egymás után lépnek, ugyanazon az eszközön ("hotseat" mód).
4. **Gyalog átalakulása:**
    - Ha egy gyalog eléri a tábla túlsó végét (1. vagy 8. sor), átalakulás történik.
    - A játékos egy felugró ablakban választhatja ki, hogy mit szeretne (vezér, bástya, futó vagy huszár).
    - Az AI automatikusan vezérré (királynővé) lépteti a gyalogját.

5. **Lépés visszavonása:**
    - Egy "Undo" gomb segítségével a játékos visszavonhatja az utolsó lépést.
    - Ez visszaállítja a tábla előző állapotát, és az ellenfél ismét léphet, mindazonáltal inkább 3 szor nyojuk meg mivel akkor a mi 2 lépésünk és az ellenségé semis.
6. **Játékállás mentése és betöltése:**
    - A mentés során a játékos az aktuális táblaállást és a játékosok sorrendjét is elmentheti, a kiválasztott mappába.
    - A betöltés lehetővé teszi, hogy a korábbi mentésből folytasd a játékot, itt szintén választhat honnan szeretné a fájl betölteni a választás grafikus.
7. **Megadás lehetősége:**
    - A játékos bármikor megadhatja a játékot, amely automatikusan az ellenfél győzelmét eredményezi.
8. **Nyertes kihirdetése:**
    - A program felismeri a sakkmattot, és értesíti a játékosokat a győztesről.
    - A játék ilyenkor automatikusan visszatér a főmenübe.
9. **Dinamikus színkezelés:**
    - Az éppen kiválasztott bábu lépéslehetőségei zöld színnel kerülnek kiemelésre.
    - A tábla alapértelmezett színei váltakozó fehér és szürke mezők.