**Sakk dokumentáció**

Osztálydiagram:

(bele kell nagyítani, hogy látható legyen)



Osztályok és metódusok rövid ismertetése:

**Ablak:** a játék közbeni megjelenítésért felel

public Ablak(String): Létrehoz egy JFrame-et és hozzáadja a komponenseket, illetve formázza az elrendezést. A String azt mondja meg, hogy a mentett vagy új játékot kell-e betölteni.

private JPanel gombok(): Az ablakon található Kilépés és Feladás gombokat hozza létre, majd egy panelben adja őket vissza. Az ablak konstruktora hívja meg, és adja hozzá saját elemeihez.

public void Vege(char): Paraméterként megkapja, melyik szín nyert, és ez alapján kiírja a győztest egy felugró ablakban.

**Menu:** A játékablakba lépés előtt menüből lehet választani

public void run(): A menü megvalósítja a Runnable interfészt, run függvényét pedig felülírja, hogy lehessen invoke-olni. Létrehozza az ablakot és a szükséges gombokat.

**Main:** Innen indul a program

public void run(): A menühöz hasonlóan megvalósítja a Runnable interfészt, és run függvényét felülírja, hogy lehessen invoke-olni.

public static void main(String[]): Elindít egy új menüt.

**MattCheck:** A sakkot, a mattot és az ezekhez tartozó speciális helyzeteket ellenőrzi

public MattCheck(Tabla, LinkedList<Babu>, LinkedList<Babu>, Kiraly, Kiraly):

A konstruktor beállítja a paraméterként kapott értékeket. Tudnia kell a két fél lehetséges lépéseit, illetve a királyok elhelyezkedését.

public void update(): Az objektum állapotát frissíti, hogy az új állást tárolja. Ehhez a lehetséges lépéseket újra kell számolni.

public boolean fekSakk()/public boolean fehSakk(): Ellenőrzi, hogy a király sakkban van-e

public boolean fehMatt()/public boolean fekMatt(): Ellenőrzi, hogy a király mattban van-e. Amennyibben sakkban van, lefuttatja a Feloldhato, Utheto és Blokkolhato függvényeket, melyekből ha valamelyik igaz, akkor védhető a sakk, így nincs matt.

public boolean Feloldhato(Map<Mezo, List<Babu>>, Kiraly): Leteszteli, hogy a sakkban lévő királynak van-e olyan lépése, ami után nem lesz sakkban. A paraméterek az ellenfél lépései és a saját király, hogy ellenőrizni lehessen a lépést.

public boolean Utheto(Map<Mezo, List<Babu>>, List<Babu>, Kiraly): Leteszteli, hogy valamelyik szövetséges bábu leütheti-e a sakkot adót úgy, hogy utána legyen sakk. A Ralephet függvény meghívásával biztosítja, hogy az ütés után nem lesz sakkban a király. Megkapja a szövetségesek lépéseit, illetve a támadó bábukat, amiből kideríthető, hogy a támadó mezőjére kik tudnak lépni.

public boolean Blokkolhato(List<Babu>, Map<Mezo, List<Babu>>, Kiraly): Végigmegy a támadó és a király közti mezőkön. Ha valamelyik köztes mezőre tud szövetséges lépni úgy, hogy utána nem lesz sakk(Ralephet függvény alapján), akkor igazzal tér vissza. A futó és vezér átlóban üt, ezért őket külön ciklusokban teszteljük.

public List<Mezo> HovaLephet(): a lepesek attribútum tárolja az általánosságban léphető mezőket. Ha nincs sakk, minden mezőre lehet lépni, ha viszont ha sakk van, csak az ezt feloldó lépések engedélyezettek. Meghívja a sakkteszteléseket(ezeken belül töltődik a lista), majd ha valamelyik sakk igaz, akkor a mattot is(itt is töltődik a lista). Végül visszaadja az így kapott összes léphető mezőt.

public boolean Ralephet(Babu, Mezo): A kapott bábuval és célmezővel eljátssza a lépést, leteszteli, hogy ebben a pozícióban sakkban van-e a saját király, majd visszaállít mindent az eredeti pozícióba.

**Babu:** A bábuk absztrakt ősosztálya

public Babu(char, Mezo, String): Konstruktorában egy színt, kezdőmezőt és egy képet kap, melyeket beállít magának

public boolean lep(Mezo): megpróbál a célmezőre lépni. Ha ott áll egy azonos színű, visszatér esemény nélkül, ha más szín, ütést hajt végre, ha üres, csak átállítja a mezővel való ismertségét.

public void kirajzol(Graphics): kirajzolja a bábu képét.

public int[] SzabalyosHossz(Mezo[][], int, int): Leteszteli, hogy a saját sorában vagy oszlopában melyik mezőkre tud lépni. Ha a sorban vagy oszlopban másik bábuhoz érkezik, ezen túl már nem tud lépni. Visszaad egy négyelemű tömböt, ami a négy irány hosszát tartalmazza.

public List<Mezo> SzabalyosAtlo(Mezo[][], int, int): Leteszteli, hogy az átlókban melyik mezőkre tud lépni a SzabalyosHossz-hoz hasonlóan. Visszaadja ezeket a mezőket.

public abstract List<Mezo> SzabalyosLepesek(Tabla): Minden bábutípus különböző módon kiszámolja a lehetséges lépéseit.

A bábu alosztályai mind meghívják a Babu konstruktorát saját konstruktorukban és megvalósítják a SzabalyosLepesek függvényt.

**Lo:**

A lépéseket úgy keresi, hogy -2 – 2 irányokban ellenőrzi az eggyel jobbra és balra lévő mezőket.

**Kiraly:**

Szabályos lépései az őt körülvevő lépések, amennyiben nem áll rajta más

**Gyalog:**

A gyalognak van egy plusz attribútuma, ami azt ellenőrzi, hogy ez az első lépése-e a játékban, mivel akkor kettőt léphet. Ehhez tartozik a boolean lep(Mezo) metódus, ami meghívja az ős lep fgv-ét, majd beállítja az attribútumot igazra.

Lépései fekete és fehér esetben máshogy működnek, mert a fekete csak lefelé tud lépni, a fehér csak felfelé. Hozzáadja az előtte álló mezőt, ha nincs rajta más, és az eggyel átlósan jobbra vagy balra lévőt, ha van rajta ellenfél.

**Futo:**

A SzabalyosAtlo függvény megadja a lépéseit

**Bastya:**

A SzabalyosHossz függvény megadja a lépéseit

**Shrek:**

Kitalált bábu, ami ugyanúgy lép, mint a gyalog, viszont ha átlósan többen is állnak előtte az ütés irányában, akkor akár ugorhat is, és leütheti a hátsó bábut.

**Vezer:**

A SzabalyosHossz és a SzabalyosAtlo kombinációjából kijönnek a lépései.

**Mezo:** A mezők tartalmazzák a rajtuk álló bábut, illetve saját elhelyezkedésüket a táblán

public Mezo(Tabla, char, int, int): Konstruktorban kap egy színt, a táblát, amihez tartozik, és a koordinátáit.

public boolean vanBabu(): Visszaadja, hogy áll-e rajta valaki

public void utes(Babu b): a jelenleg rajta álló bábut kitörli magáról és a bábukat tároló listából is, majd a paraméterként kapott bábuval állítja be az ismertséget.

public void paintComponent(Graphics): Override, ami a kirajzolásért felel. Beállítja a konkrét színeket és kitölti a téglalapot

public Babu torles(): Elfelejti az eddig rajta álló bábut

**Tabla:** A tábla megvalósítása, az egér érzékelése

public Tabla(Ablak, String): Inicializálja a tábla attribútumait. A mezőket létrehozza és a bábukat a kapott fájlnév alapján felállítja.

public int [][] beolvas(String fajl): a kapott fájlból beolvassa a pályát.

public void paintComponent(Graphics): Override, ami meghívja a mezők színezését és a bábuk kirajzolását.

public void Mentes(): Az aktuális pályát elmenti egy fájlba olyan formátumban, ahogyan azt ki is lehet olvasni.

public void mousePressed(MouseEvent): A kattintás helyén lévő bábut eltárolja.

public void mouseReleased(MouseEvent): Ha az egérgombot elengedjük, ellenőrzi, hogy szabályos-e a lépés az elengedés helyén lévő mezőre. Ha igen, ez alapján frissíti a táblát.

Felhasználói útmutató:

Indításkor egy menüt látunk. Itt három lehetőség közül választhatunk kattintással: új játékot kezdhetünk, egy mentettet betölthetünk, vagy kiléphetünk a játékból.

Ha betöltünk egy játékot, a sakktábla felett két gomb van: a kilépés, mely feldobja a mentés lehetőségét majd bezárja az alkalmazást, és a feladás, ami visszadob a főmenübe.

A bábuk egérrel húzhatók, azaz a gomb megnyomásával veszünk fel egy bábut és elengedéssel tesszük az új pozíciójára. Matt esetén kiíródik a nyertes.

Saját pályát is létrehozhatunk. Ehhez a „palya” nevű txt fájlt kell módosítanunk. A szövegfájl felépítésére a következő megkötések vannak: Az első két sorban a pálya szélessége és hossza szerepel egy számmal kifejezve. A következő sorban a pálya „rajzolható” le számokkal, a sorok egymás alatt, az oszlopok szóközzel elválasztva(a sorok és oszlopok számának meg kell egyeznie a fentebb megadott értékeknek).

A bábuk számkódjai a következők:

fekete:

bástya(1)

futó(2)

huszár(3)

vezér(4)

király(5)

gyalog(6)

Shrek(7)

A fehér bábuk ugyanezen számok negatív előjeles változatai.

A játékban egy új bábu is van, Shrek: hasonlít a gyaloghoz, de első körében is csak egyet tud lépni, és ha előtte átlósan két ellenfél is van egymás után, akkor a hátrébb lévőt is le tudja ütni, az elsőt átugorva.