Dokumentáció

Működés:

A programot megnyitva egy főmenüvel találkozunk, ahol meg kell adni a két játékos nevét. Ha a két játékos neve nincs megadva, vagy ugyanaz a nevük, akkor a játékot nem tudjuk elindítani.

Ha megadtuk mindkét fél nevét, akkor indulhat a játék. 2 játék indítási funkció közül tudunk választani, az első, hogy új játékot kezdünk, a második pedig, hogy az előző be nem fejezett játékot visszatöltjük, ha van ilyen, ha nincs akkor ez az opció is új táblát hoz létre.

Amint bekerülünk a játékba, egy egyszerű sakktáblát látunk, amin az egyszerű sakk szabályokat betartva tudunk sakkozni. A játékot a fehér játékos kezdi alapból, de ha a korábbi betöltése opciót választjuk, akkor ott ez megváltozhat, mivel mindig a mentés pillanatában soron következő játékost menti el a rendszer.

Minden bábúnak zölddel jelöli ki a program, hogy melyik mezőre tudunk lépni ütés nélkül, pirossal pedig azokat a mezőket emeli ki, amelyekre ütéssel tudunk lépni.
A játék sárga színnel emeli ki a királyt, ha sakkban van. A játékban bal oldalt a leütött bábúkat tudjuk megtekinteni.

A játékban található egy menüsor, ahol a „File” menüpontot kiválasztva meg tudjuk tekinteni a toplistát, valamint ki tudunk lépni a játékból. Játék közben csak a File menü kilépés gombjára kattintva tanácsos kilépni, a program így menti el a félbehagyott pályát.

A toplistára nyomva megjelenik az első három ember, akik a legkevesebb lépésből adtak mattot. Megjelenik a nevük és a lépéseik száma is. Ezeken felül alul megjelenik egy vissza gomb, amivel ki tudunk lépni a toplista nézetből.

Ha vége van a játéknak, tehát matt- vagy patthelyzet alakul ki, akkor megjelenik egy panel, amely matt esetén kiírja a győztes nevét, valamint van rajta 2 gomb, egy kilépés és egy új játék gomb. Az új játékkal új játékot tudunk indítani, a kilépés gombbal pedig egyszerűen ki tudunk lépni a programból. A kilépés gombbal való kilépés nullra teszi az előző játék értékét, mivel nincs befejezetlen játék.

Főbb osztályok és metódusaik:

public abstract class Babu:

Ez egy absztrakt osztály, vagyis nem példányosítható, mivel sakkban sima bábú nem létezik, ezért is van neki egy BabuFajta enum-ja, amit csak azért hoztam létre. hogy a bábúfajtákat meg tudjam különböztetni.
Minden Babu-nak van egy BabuFajta babuFajta(a bábú fajtája), egy boolean elsoLepesE(megadja, hogy az elsőt fogjuk-e lépni a bábúval), egy int babuPos(megadja a Bábú koordinátáját) és egy Szin babuSzin(megadja, hogy milyen színű a bábú) és ezek mindegyikéhez tartozik egy getter függvény.
A BabuFajta enum, mindegyik leszármazottja vissza tudja adni, hogy ő bizonyos bábú-e egy boolean metódussal. Pl.: Király vissza tudja adni a boolean isKiraly() metódussal a true-t, de a gyalog a false-t.
A Babu osztályban megtalálhatók abstract metódusok, ezekről leljebb a konkrét bábúk osztályainál lesz szó.

Bábúk osztályai(Gyalog, Csiko, Futo, Bastya, Kiraly, Kiralyno):

Ezeket az osztályokat nem szedem külön, mivel mindegyiknek ugyanaz a lényege, csak mindenhol más-más értékekkel.
Mindegyik fajta bábúnak megvan előre adva, hogy a jelenlegi pozíciójuktól számítva, melyik irányokba tudnak lépni, ezt az int[] lehetsLepes adja, meg, ezek alól kivétel a Kiralyno, Bastya és Futo osztály, mivel az ő lépéseik lehetőségét vektorokban fogtam fel ezért nekik csak a lehetséges irányok vannak megadva egy int[] lepVektor tömbben.
A List<Lepes> szabLepesek metódus visszaadja azokat a lépéseket, amelyeket megtehetünk egy bábúval, figyelve azokra az esetekre, amikor a bábú a tábla valamely szélén helyezkedik el, valamint a királynál figyelembe veszi a sáncolás lehetőség, illetve a gyalognál az első lépés esetén a dupla lépést, illetve azt, hogy a gyalog csak keresztbe tud ütni.
Mindegyik bábúnak vagy egy Babu babuLep(Lepes lepes) metódusa, amely visszaad egy új bábút a jelenlegi bábú színével és fajtájával megegyező bábút, a lépés célkoordinátájának helyén, azonban az első lépés értékét

ebben az esetben false-ra állítja.
Minden bábúnak van egy értéke(ezt a leütött bábúk panelján való elhelyezés miatt csináltam és ehhez tartozik egy getter függvény.
Ahol szükséges volt ott létrehoztam több boolean metódust amely visszaadja, hogy a jelenlegi pozíciója és a lépés célkoordinátája, nem helyezkednek-e el, úgy, hogy az a lépés ne lehessen végrehajtható.

public abstract class Jatekos

A Jatekos szintén egy abstract osztály, mivel csak „színes” játékosok léteznek.
Minden játékosnak, van egy Tabla tabla(A Tabla amin a játékos létezik), egy Kiraly jelenJatekosKiralya(Az a király, amelyik ehhez a játékoshoz tartozik), egy boolean isSakk(megadja, hogy sakkban van-e a játékos), illetve egy List<Lepes> lehetsLepesek(megadja, hogy melyek ennek a játékosnak a lehetséges lépései.
Minden játékosnak, van egy Collection<Lepes> tamadasSzamol(int babuPos, Collection<Lepes> ellenfelLepesek) metódusa, amely visszadja, hogy a megadott collection-ból van e, olyan lépés, amelynek célja a megadott pozíció.
Definiáltam egy boolean vanUtMashova() függvényt, amely megadja azt, hogy van -e még a játékosnak végrehajtható lépése.
A játékosokhoz tartozó Babu kiralyLetrehoz, visszaadja a játékos táblájáról azt a királyt, amely hozzá tartozik.
Létezik egy boolean isLegalLepes(Lepes lepes) metódus, amely megadja a megadott lépésről, hogy legális-e.
Létrehoztam több boolean metódust(isSakkMatt, isSakk, isPatt), amelyek visszaadják, hogy a játékos az előbbi helyzetek valamelyikében van-e.
A LepesValosit lep(Lepes lepes) metódus egy úgymond virtuális táblán megvalósítja a lépést és egy LepesValosit változóban eltárolja, hogy ez a lépés lehetséges-e. A LepesValosit osztályt, azért hoztam létre, hogy el tudjam tárolni, hogy az adott lépés megvalósítható-e.
Emellett a játékosoknak van több abstarct metódusa is van, de ezekről, majd a definíciójuk helyen ejtek szót.

FeherJatekos/FeketeJatekos:

A Jatekos osztály leszármazottjai és az összes bennük definiált metódus a Jatekos osztály metódusainak felülírása. A gettereken kívül egy fontos metódusuk van a public Collection<Lepes> CastleSzamol(Collection<Lepes>, Collection<Lepes>), amely mindkét leszármazott esetében ugyanazt csinálja, csak más Collection-okat kap meg paraméterként. A CastleSzamol visszaadja az adott játékos esetében a lehetséges lépéseket, amelyek egy sáncolást valósítanak, meg.

public enum LepesAllapot:

Egy enum, amelyet a lépések állapotának, megadására hoztam létre.
Egy lépés lehet, NemLehetsLepes, ha a lépést nem lehet végrehajtani a táblán, SakkbanHagy, ha a lépés után továbbra is sakkban leszünk és lehet Vegrehajtva, ha a lépés végrehajtható.

public class LepesValosit

A lépésekhez eltárol egy táblát és el tudjuk az adott lépéshez az adott táblán tárolni, a lépés tulajdonságát.

public abstract class Lepes

A lépések abstract ősosztálya, mivel egyszerű lépés nem létezik.
A lépéseknek van egy táblájul(mTabla), amelyen a lépés értelmezve van, van egy bábúja, amelyet mozgatunk(mBabu) és van egy cél koordinátája(celkoord), ahová helyezni szeretnénk a bábút.
A legfontosabb metódusa a Lepes osztálynak a Tabla megvalosit(), ez lényegében végrehajtja a lépést és visszaadja a módosított táblát.

public static class SimaLepes:

Semmit nem változtatunk benne a Lepes osztályon, mivel ezt csak a származtatás és a könnyebb olvashatóság miatt hoztam létre.

public static class TamadoLepes:

Ennél a lépésfajtánál létrehozunk egy támadott bábút(tamadott), amely azt a

bábút adja meg, amelyik a cél mezőn van, de ezen felül nem változtatunk semmit az eredeti Lepes osztály metódusain, kivéve a különböző setterek.

public static abstract CastleLepes:

A sáncolást valósítja meg, azzal, hogy felülírja a megvalosit() metódust, mivel sáncolásnál a bástyát és a királyt is mozgatni kell.
Kettő leszármazottja van(KiralyOldalCastle, KiralynoOldalCastle), de őket csak a könnyebb olvashatóság miatt hoztam létre.

public static class RosszLepes:

Ha 2 koordináta között nem hozható létre lépés semmilyen módon akkor egy Rosszlépést hozunk létre, amelynek a megvalosit() metódusa RuntimeException-t dob így nem tudunk, ha esetleg valahol a kódban mehívjuk egy RosszLepes megvalosit metódusát, akkor nem kapunk NullPointerException-t, illetve a metódusok tesztelésére is tökéletesen alkalmazható.

public abstract class Mezo:

A mezők abstarct osztálya. Minden mezőnek van egy koordinátája és a Map-ja a koordinátákról és a hozzájuk tartozó mezőkből.
A createAllUresMezo() metódus a visszaad egy Map-ot, amelynek minden Integer tagjához(0-63) felvesz egy üres mezőt.
A mezoLetrehoz(int, Babu) visszaad egy mezőt annak függvényében, hogy a Babu null-e mert, ha igen akkor létrehoz egy teli mezőt, amelyen a megadott bábú van, ha null, akkor a Map adott int-hez tartozó Babu-t adja vissza.

public static class UresMezo

Semmi érdemi különbsége nincs az ősosztálytól, csak az öröklődés és a jobb olvashatóság miatt hoztam létre.

public static class TeliMezo:

A teli mezőknek van egy Babu változójuk, amely eltárolja azt, hogy milyen bábú áll a mezőn, de egyéb különbsége nincs az ősosztálytól.

public class Tabla:

Minden táblának van egy List<Mezo> (jatekTabla) változója, amely eltárolja a táblán található mezőket, van kettő Collection<Babu> változója, amelyekkel a fehér és a fekete játékoshoz tartozó bábúkat tárolja el, van egy Jatekos változója, amely azt a játékost adja meg aki éppen soron van és van egy FeketeJatekos és egy FeherJatekos változója, amelyek eltárolják a 2 játékost.

A Collection<Lepes> lepesSzamit(Collection<Babu>) metódus visszaadja a megadott bábúkhoz tartozó összes lehetséges lépés kollekcióját, lényegében az egy játékoshoz tartozó összes lehetséges lépést és az összes lehetséges lépést a táblán lehet vele megadni.

A Collecntion<Babu> jelenlegiBabuk(List<Mezo>, Szin) metódus visszaadja a megadott színhez tartozó összes jelenleg elérhető bábút a táblán.

A List<Mezo> jatekTablaGen(Generator) létrehozza a mezőket az alapján, hogy a megadott generátor Map-jében az adott int-hez tartozik-e bábú.

A Tabla alapTabla() metódus visszaad egy táblát, amelyen „fel vannak állítva” a bábúk alaphelyzetben.

A Tabla korabbiBetolt(String[], Szin) a megadott String[] alapján felállít egy táblát, amelynek a kezdőszínét a megadott Szin alapján beállítja és visszaadja a táblát.

public static class Generator:

A Generator osztály eltárolja egy Map-ban az összes lehetséges integer koordinátához a hozzájuk tartozó bábúkat és van egy Szin változója, amiben az éppen soron lévő játékos színét tároljuk el.

A Generator setBabu(Babu) beállítja a Map azon részére a bábút, amelyet a bábú pozíciója ad meg.

A Tabla generate() metódus visszaad egy táblát, amelynek a konstruktorának odaadja saját magát.

public class TablaUtil:

A kivSor(int) és kivOszlop(int) megadja, hogy az adott koordinátától indulva mely koordináták vannak az adott sorban/oszlopban és amelyek ott vannak oda true-t tesz és emellé definiáltam a boolean[] változókat, amelyek eltárolják a koordinátákról, hogy az adott sorban/oszlopban vannak-e.

public class SakkTabla:

Ebben az osztályban van létrehozva az egész GUI.
Itt lényegében legtöbbször csak a java.swing leszármazottakat hozom létre és módosítom ahogyan szeretném, ezért ezekre nem térnék ki részletesen, ezért a továbbiakban a fontosabb saját osztályaimról és a fontosabb saját függvényeimről lesz szó.
A tablaToFile() metódus elmenti a tábla jelenlegi állását egy fájlba.
A tablaFromFile() metódus beolvas és létrehoz egy táblát egy fájlból.
A szinFromFile beolvassa a soron következő játékos színét egy fájlból.
A JMenu fileMenuLetrehoz() létrehoz egy felső menüsávot és rajta egy lenyíló menüt.

public TablaPanel:

Ez a panel jeleníti meg magát a sakktáblát és tablaRajzol(Tabla) metódusa a megadott tábla alapján „megrajzol” egy táblát.

public static class StartPanel:

Ez a Panel jelenik meg, amikor belépünk a játékba, ez lényegében a főmenü.

public static class EndPanel:

Ez a Panel jelenik meg a játék végén matt és patt esetén ez lényegében szintén egy főmenü csak a játék végén.

public static class TopListPanel:

Ez a panel valósítja meg a toplistának a megjelenítését.

public static class MezoPanel:

Ez a panel egyenként a mezők megjelenítéséért felel.
A gyoztesToFile() elmenti a győztes lépésszámát file-ba.
A kiralyKiemel() beszínezi a király mezőjének színét sárgára-ha sakkban vagy, pirosra, ha matt van és narancssárgára, ha patt van.
A mezoSzinBeallit() beállítja a mezők alapszínét.
A lehetsKiemel() kiemeli zölddel az adott bábúhoz tartozó lehetséges sima lépéseket és pirossal a támadó lépéseket.
A Collection<Lepes> babuLehetsLepesek(Tabla) metódus visszaadja a mozgatni kívánt bábúhoz tartozó lehetséges lépéseket.
A babuIconBeallit(Tabla) beállítja a tábla elérhető bábúhoz tartozó ikonokat.

public static class LepesNyilvantarto:

Eltárolja az összes végrehajtott lépést.
A nyilvantartoToFile() metódus elmenti a nyilvántartás támadó lépéseit file-ba, mivel így, ha az előző játékot szeretnénk vissza tölteni, akkor a már leütött bábúkat is vissza tudjuk tölteni.
A nyilvantartoFromFile() metódus betölti az elmentett támadólépések támadott bábúit file-ból és létrehoz támadó lépéseket, amiknek a támadott bábún kívüli tulajdonságai nem számítanak, mivel csak a támadott bábú jelenik majd meg az eddig leütött bábúk paneljén.

public class UtottBabukPanel:

Ez a panel jeleníti meg az eddig leütött bábúkat, méghozzá az értékük szerint növekvő sorrendben és minden hozzáadásnál újra rendezi a kiírást.
A hozzaad(LepesNyilvantarto) végigmegy a nyilvántartón és először sorba rendezi a leütött bábúkat, majd az ikonjaikat hozzáadják a panelhez.