A programozás alapjai 2.

Házi Feladat Dokumentáció

**SAKK TÁBLAJÁTÉK**

Készítette: Kocsis László, IG72OK

Kocsislaci95@gmail.com

Készítés féléve: 2016/17/2

Tartalomjegyzék

[Felhasználói dokumentáció 4](#_Toc9243757)

[Osztályok statikus leírása 4](#_Toc9243758)

[Game 4](#_Toc9243759)

[Felelőssége 4](#_Toc9243760)

[Ősosztályok 4](#_Toc9243761)

[Attribútumok 4](#_Toc9243762)

[Metódusok 4](#_Toc9243763)

[Player 5](#_Toc9243764)

[Felelőssége 5](#_Toc9243765)

[Ősosztályok 5](#_Toc9243766)

[Attribútumok 5](#_Toc9243767)

[Metódusok 5](#_Toc9243768)

[Piece 5](#_Toc9243769)

[Felelőssége 5](#_Toc9243770)

[Ősosztályok 5](#_Toc9243771)

[Attribútumok 5](#_Toc9243772)

[Metódusok 6](#_Toc9243773)

[Pawn 6](#_Toc9243774)

[Felelőssége 6](#_Toc9243775)

[Ősosztályok 6](#_Toc9243776)

[Attribútumok 6](#_Toc9243777)

[Metódusok 6](#_Toc9243778)

[Rook 6](#_Toc9243779)

[Felelőssége 6](#_Toc9243780)

[Ősosztályok 6](#_Toc9243781)

[Attribútumok 7](#_Toc9243782)

[Metódusok 7](#_Toc9243783)

[Knight 7](#_Toc9243784)

[Felelőssége 7](#_Toc9243785)

[Ősosztályok 7](#_Toc9243786)

[Attribútumok 7](#_Toc9243787)

[Metódusok 7](#_Toc9243788)

[Bishop 7](#_Toc9243789)

[Felelőssége 7](#_Toc9243790)

[Ősosztályok 7](#_Toc9243791)

[Attribútumok 8](#_Toc9243792)

[Metódusok 8](#_Toc9243793)

[Queen 8](#_Toc9243794)

[Felelőssége 8](#_Toc9243795)

[Ősosztályok 8](#_Toc9243796)

[Attribútumok 8](#_Toc9243797)

[Metódusok 8](#_Toc9243798)

[King 8](#_Toc9243799)

[Felelőssége 8](#_Toc9243800)

[Ősosztályok 8](#_Toc9243801)

[Attribútumok 9](#_Toc9243802)

[Metódusok 9](#_Toc9243803)

[UML osztálydiagramm 10](#_Toc9243804)

[Összegzés 10](#_Toc9243805)

[Mit sikerült és mit nem sikerült megvalósítani a specifikációból? 11](#_Toc9243806)

[Mit tanultál a megvalósítás során? 11](#_Toc9243807)

[Továbbfejlesztési lehetőségek 11](#_Toc9243808)

[Képernyőképek a futó alkalmazásról 12](#_Toc9243809)

# Felhasználói dokumentáció

A házi feladatomban implementáltam egy sakk táblajátékot. Mivel grafikus környezetet nem hoztam létre ezért a játék konzolon játszható. A program indítása és egyedi azonosítók megadása után már lehet is látni a felrajzolt táblán a rajta elhelyezkedő bábukkal. A kis betűk a sötét, a nagy betűk a világos figurákra utalnak. A főmenüben elindíthatunk egy lépést vagy kiléphetünk a játékból. Az utóbbi választásával a program leáll. Ha a lépés indítását választjuk utána meg kell adni, hogy mely pozícióról mely másik pozícióra szeretnénk lépni. A pozíciót alakja egy kisbetű és egy szám egymás után, minden egyéb karaktert mellőzve. A lehetséges lépéseket ismerni kell ahhoz, hogy megengedett lépést tudjunk beadni a programnak, mivel alapból nem ajánlja fel a lehetséges eseteket. A program a beadott koordináták után leellenőrzi, hogy a lépés lehetséges-e és ha igen akkor megteszi azt, ha nem akkor hibát dob és újra kell kezdeni a lépés megadását.

# Osztályok statikus leírása

## Game

### Felelőssége

Ez az osztály felel a felhasználói interakciókért, valamint egységbe foglalja a játékhoz szükséges elemeket.

### Ősosztályok

Ez az osztály nem öröklődik.

### Attribútumok

#### Privát

* blackPlayer – Player típusú játékos
* whitePlayer – Player típusú játékos
* board – 8x8-as char tömb. A tábla.
* isThereAWinner – logikai érték, a játék végét jelzi
* isWhitesTurn – logikai érték, a játékosok körét jelzi

#### Védett

#### Publikus

### Metódusok

#### Privát

#### Védett

#### Publikus

* Konstruktor
* Destruktor
* playGame – a játék felhasználói interakcióit valósítja meg
* print – kirajzolja a táblát
* move – egy lépést valósít meg a bemenetek alapján, valamint vizsgálja, hogy lehetséges-e a lépés
* getPieceOnPosition – egy tábla pozícióról lekérdezi, hogy mi helyezkedik ott el
* setPieceOnPosition – elhelyez egy figurát a tábla egy adott pontjára
* getIsWhitesTurn – a kimenete, hogy éppen a világos vagy a fekete jön

## Player

### Felelőssége

Ez az osztály felel azért, hogy egy játékos és annak bábui azonosíthatók legyenek, valamint gondoskodik, hogy a felhasználó által megadott lépések szerint a bábuk a helyükre kerüljenek.

### Ősosztályok

Ez az osztály nem öröklődik.

### Attribútumok

#### Privát

* name – std::string típusú név, azonosító
* isWhite – logikai érték, a játékos színét adja meg
* isItsTurn – logikai érték, azt adja meg, hogy éppen ez a játékos jön-e
* pieces – Piece típusú pointereket tároló dinamikus tömb pointere

#### Védett

#### Publikus

### Metódusok

#### Privát

#### Védett

#### Publikus

* Konstruktor
* Destruktor
* getIsWhite – visszaadja az isWhite adattagot
* getName – visszaadja az name adattagot
* getPieces – visszaadja a pieces adattagot
* setTaken – egyet a játékos bábui közül levesz az asztalról
* takeMove – lép egyet az egyik bábuval

## Piece

### Felelőssége

Egy bábu általános tulajdonságait és metódusait zárja egységbe. Absztrakt osztály ezért nem példányosítható. A ténylegesen példányosodó bábuk az ebből származtatott különböző figura osztályokból jönnek létre.

### Ősosztályok

Ez az osztály nem öröklődik.

### Attribútumok

#### Privát

#### Védett

* isTaken – logikai érték, ami megmondja, hogy adott bábu játékban van-e
* isWhite – logikai érték, ami megmondja, hogy adott bábunak milyen a színe
* character – char típusú változó, ami a bábu jelölését tartalmazza
* xpos – char típusú változó, ami az vízszintes tengely szerinti pozícióját tartalmazza a figurának
* ypos - char típusú változó, ami az függőleges tengely szerinti pozícióját tartalmazza a figurának

#### Publikus

### Metódusok

#### Privát

#### Védett

#### Publikus

* Konstruktor
* Destruktor
* getCharacter – visszaadja a character adattagot
* getXPos – visszaadja az xpos adattagot
* getYPos – vissaadja az ypos adattagot
* setPos – megadott paraméterek alapján beállítja a pozíciót leíró adattagokat
* setTaken – a figura pozícióját átállítja, úgy, hogy ne legyen a pályán, valamint az isTaken adattagot igazzá teszi
* move – virtuális tagfüggvény, minden figura esetében más logikai vizsgálatok alapján működik

## Pawn

### Felelőssége

Egy speciális figura, gyalog.

### Ősosztályok

Piece

### Attribútumok

#### Privát

* isInstartPos – logikai érték, ami megmondja, hogy a figura a kezdeti pozíciójában van-e

#### Védett

#### Publikus

### Metódusok

#### Privát

#### Védett

#### Publikus

* Konstruktor
* Destruktor
* move – adott paraméterek alapján lép a gyalog és ellenőrzi, hogy ezt megteheti-e

## Rook

### Felelőssége

Egy speciális figura, bástya.

### Ősosztályok

Piece

### Attribútumok

#### Privát

#### Védett

#### Publikus

### Metódusok

#### Privát

#### Védett

#### Publikus

* Konstruktor
* Destruktor
* move – adott paraméterek alapján lép a gyalog és ellenőrzi, hogy ezt megteheti-e

## Knight

### Felelőssége

Egy speciális figura, ló.

### Ősosztályok

Piece

### Attribútumok

#### Privát

#### Védett

#### Publikus

### Metódusok

#### Privát

#### Védett

#### Publikus

* Konstruktor
* Destruktor
* move – adott paraméterek alapján lép a gyalog és ellenőrzi, hogy ezt megteheti-e

## Bishop

### Felelőssége

Egy speciális figura, futó.

### Ősosztályok

Piece

### Attribútumok

#### Privát

#### Védett

#### Publikus

### Metódusok

#### Privát

#### Védett

#### Publikus

* Konstruktor
* Destruktor
* move – adott paraméterek alapján lép a gyalog és ellenőrzi, hogy ezt megteheti-e

## Queen

### Felelőssége

Egy speciális figura, vezér.

### Ősosztályok

Piece

### Attribútumok

#### Privát

#### Védett

#### Publikus

### Metódusok

#### Privát

#### Védett

#### Publikus

* Konstruktor
* Destruktor
* move – adott paraméterek alapján lép a gyalog és ellenőrzi, hogy ezt megteheti-e

## King

### Felelőssége

Egy speciális figura, király.

### Ősosztályok

Piece

### Attribútumok

#### Privát

#### Védett

#### Publikus

### Metódusok

#### Privát

#### Védett

#### Publikus

* Konstruktor
* Destruktor
* move – adott paraméterek alapján lép a gyalog és ellenőrzi, hogy ezt megteheti-e

# UML osztálydiagramm

# Összegzés

Ábra - UML diagram

## Mit sikerült és mit nem sikerült megvalósítani a specifikációból?

Sikerült egy objektum orientált irányelvekkel működő sakktáblát és rajta mozgatható figurákat létrehoznom, úgy, hogy a figuráknak van „gazdája”, vagyis létre kell hoznunk játékosokat a játék működéséhez. Nem sikerült tökéletesre befejeznem a lépések helyességét figyelő függvényeket, mesterséges intelligenciát és a játék végén az eredmények kiíratását, azonban ezeket a továbbiakban be fogom fejezni.

A házifeladat készítésekor megismertem pontosabban a heterogén kollekció, kompozíció és aggregáció működését.

## Mit tanultál a megvalósítás során?

A játék sokkal komplexebb, mint elsőre gondoltam, még úgy is, hogy előtte papíron összeírtam milyen elemek lesznek szükségesek. A lépések vizsgálata önmagában derék feladat, és azok után a még a speciális „lépések” nincsenek megemlítve se.

## Továbbfejlesztési lehetőségek

Jelen állapotában még nem tökéletes játékként funkciónál, ezt szeretném kijavítani, hogy teljesértékű legyen. Valamint egy „minimax” elven működő IA-t is szeretnék bevezetni a programba, hogy egyedül is lehessen játszani.

# Képernyőképek a futó alkalmazásról



 

