A programozás alapjai 2.

Házi Feladat Dokumentáció

**SAKK TÁBLAJÁTÉK**

Készítette: Kocsis László, IG72OK

Kocsislaci95@gmail.com

Készítés féléve: 2016/17/2

Tartalomjegyzék

[Dokumentációval kapcsolatos teendők 3](#_Toc8863111)

[Felhasználói dokumentáció 4](#_Toc8863112)

[Osztályok statikus leírása 4](#_Toc8863113)

[Game 4](#_Toc8863114)

[Felelőssége 4](#_Toc8863115)

[Ősosztályok 4](#_Toc8863116)

[Attribútumok 4](#_Toc8863117)

[Metódusok 4](#_Toc8863118)

[Player 4](#_Toc8863119)

[Felelőssége 4](#_Toc8863120)

[Ősosztályok 5](#_Toc8863121)

[Attribútumok 5](#_Toc8863122)

[Metódusok 5](#_Toc8863123)

[Piece 5](#_Toc8863124)

[Felelőssége 5](#_Toc8863125)

[Ősosztályok 5](#_Toc8863126)

[Attribútumok 5](#_Toc8863127)

[Metódusok 6](#_Toc8863128)

[UML osztálydiagramm 7](#_Toc8863129)

[Összegzés 7](#_Toc8863130)

[Mit sikerült és mit nem sikerült megvalósítani a specifikációból? 8](#_Toc8863131)

[Mit tanultál a megvalósítás során? 8](#_Toc8863132)

[Továbbfejlesztési lehetőségek 8](#_Toc8863133)

[Képernyőképek a futó alkalmazásról 9](#_Toc8863134)

# Dokumentációval kapcsolatos teendők

# Felhasználói dokumentáció

A házi feladatomban implementáltam egy sakk táblajátékot. Mivel grafikus környezetet nem hoztam létre ezért a játék konzolon játszható. A program indítása után már lehet is látni a felrajzolt táblát a rajta elhelyezkedő bábukkal. A kis betűk a sötét, a nagy betűk a világos figurákra utalnak. A kirajzolás után már lehet is kezdeni a játékot. Az 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.

# 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 a, hogy mi helyezkedik ott el
* setPieceOnPosition – elhelyez egy figurát a egy adott pontjára

## Player

### 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 a, hogy mi helyezkedik ott el
* setPieceOnPosition – elhelyez egy figurát a egy adott pontjára

## Piece

### 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 a, hogy mi helyezkedik ott el
* setPieceOnPosition – elhelyez egy figurát a egy adott pontjára

# 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 továbbá az AI megvalósítása ezért az egyik osztály örököltetés még nincs kihasználva, viszont ezt a továbbiakban be szeretném fejezni, ezért nincs kiszedve a programból.

## 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