KIV/UPG

Semestrální práce

Hra šachy v Java

Alimzhan Mukanov A22B0388P
26.3.2023

Část 1:

Zadání:

Část 1: Základní vizualizace (až 10 bodů)

Základní funkční požadavky (7 bodü): Po spuštění programu pomocí alespoň jedním z příkazů Run. cmd nebo . /run. sh (případný nefunkční druhy skript při odevzdaní z projektu smažte), se zobrazí okno o minimální počáteční velikosti 800×600px. V okně se vykresli Čtvercová šachovnice, která bude na středu okna a bude zabírat maximální možný prostor tohoto okna. Velikost okna půjde libovolné měnit a po změně velikosti okna program na tuto akci zareaguje (překreslí okno tak, aby bylo vše korektně zachováno). Na šachovnici se dále vykreslí jednotlivé kameny v zahajovací pozici dle pravidel šachu. Kameny budou vykresleny vektorově a bude rozpoznatelné, který kámen je který.

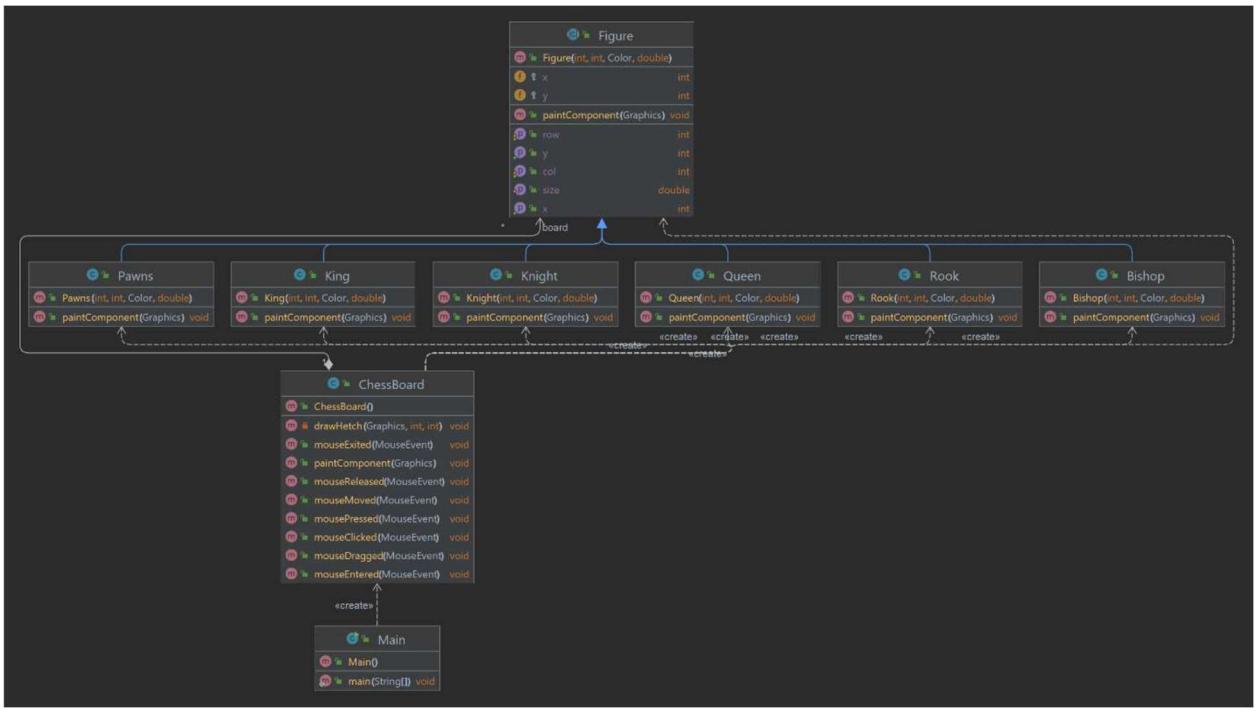
Další požadavky: Kámen půjde pomocí Drag & Drop přesunout na libovolné jiné pole. Pokud dojde k posunu na ji obsazené pole, původní kámen bude odstraněn. Součástí odevzdáni bude kompletní dokumentace dle formátu poskytnutém vyučujícími.

Řešení:

Pro splnění první části práce bylo vytvořeno několik tříd pro různé funkce. První třída, Main.java, je určena pro spuštění programu a nastavení velikosti na 800x600px. Druhou třídu, ChessBoard.java, jsem vytvořil pro vizualizaci šachovnice, figur a implementaci funkce Drag & Drop.
Pro změny velikosti figur a šachovnice se inicializuje componentListener s metodou componentResize(). Tato inicializace reaguje na změnu rozměru okna a změní velikost kamenů a šachovnice.

Třidy:

- **UML Diagram třid:**



- **Main.java**

  Je hlavní třida projektu, která zpustí aplikace. Metoda main nastaví velikost okna, umístí okno do středu obrazovky.

- **ChessBoard.java**

  Je hlavní třida projektu. Dědí od JPanelu a implementuje MouseListener a MouseMotionListener pro práci s Drag and Drop funkcionalitou. Třída kreslí šachovnici a figurky. Šachovnice je bílo-šedá a figurky jsou kresleny v jiných třídách. paintComponent(Graphics g) vykreslí šachovnice a figurky na ní.

- **Figure.java**

  Je abstraktní třida a hlavní třida pro kreslení figur. Dědí od JPanelu.

- **Queen.java, King.java, Knight.java, Bishop.java, Rook.java, Pawns.java**

  Tyto třídy představují implementaci jednotlivých figurek. Každá třída dědí od Figure.

Část 2:

Zadání:

Část 2: Pokročilá vizualizace (až 15 bodů)

Základní funkční požadavky (8 bodů): Program musí splňovat kompletní Část 1 tohoto zadání (včetně Dalších požadavků). Program zvaliduje provedený tah od uživatele a provede ho pouze, pokud je v souladu s pravidly šachu. Budou ošetřena veškerá (i komplexnější) pravidla pohybu (braní mimochodem, rošáda, proměna; stačí v dámu). Pohyb bude animován, tzn. kámen se postupně posune z počáteční pozice do pozice koncové (maximální krok pohybu je lOpx, pohyb bude trvat přesně půl sekundy reálného času). Dojde-li k matu či patové situaci, hra bude ukončena a uživatel bude seznámen o výsledku hry. Poté bude možné hru restartovat od počáteční pozice, aniž by bylo nutné celý program uzavřít. Dokumentace z 1. části odevzdání bude rozšířena o poznatky z této (druhé) části odevzdání. Další požadavky: Program bude obsahovat tlačítko, které zobrazí okno s grafem, jak dlouho trvalo odehrát jednotlivé tahy. Graf bude řádně popsán a bude obsahovat dvě oddělené datové řady pro oba hráče zvlášť). Poslední provedený tah bude označen (např. zvýramením počátečního a koncového pole tahu) tak, aby protihráč věděl, na jaký tah má reagovat. Po označení kamene se zvýrazní pole, kam s tímto kamenem lze táhnout dle kompletních pravidel šachu.

Část 3: Volitelná rozšíření (až 15 bodů)

Hodiny (3 body): Dovolte pomocí argumentů příkazové řádky, aby uživatel zvolil časový interval pro každého z hráčů a časový inkrement po každém tahu. Stav hodin se bude zobrazovat v okně a průběžně měnit (nesmí dojít k tomu, aby čas přeskočil o více než 1 sekundu). Pokud jednomu z hráčů zvolený časový interval nebude stačit, daná hra se musí ukončit a uživatel bude seznámen s výsledkem hry.

Lepší CPU (6 bodů): Naprogramujte černého tak, aby za něj hrál počítač. Algoritmus musí porazit cvičícího alespoň 2x ze tří her, aby tento bod byl splněn. Pro řešení tohoto bodu (pouze) lze použit externích knihoven (např. Stockfish apod.) Bude umožněno černého přepnout zpět na lidského hráče.

Rastrový export (1 bod): Umožněte (např. tlačítkem, rozbalovacím menu) export aktuálního stavu hry do PNG formátu. Veškerý povinný obsah okna z části 1 a 2 musí být exportován.

Zobrazení sebraných kamenů (1 bod): U obou hráčů bode zobrazeno, jaké kameny mu byly vzaty. Také bude vykreslen celkový rozdíl v cenách kamenů

Řešení:

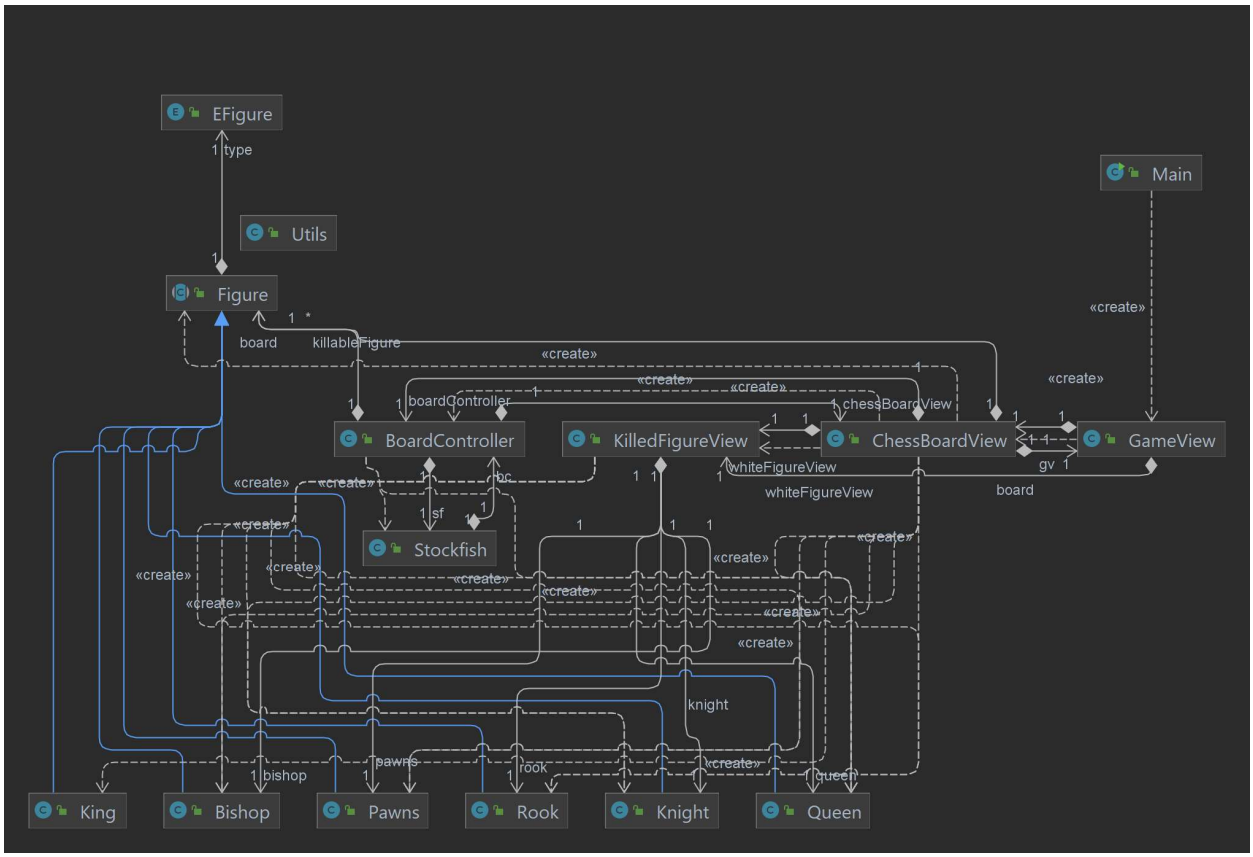
Po splnění první části práce byla zmíněná architektura projektu. Vykreslovací třídy byly oddělený od logiky, čímž připomíná architekturu MVC. Hlavní třída,

Main.java, je určena pro spuštění programu a nastavení GUI šachovnice. Projekt obsahuje několik balíků:

- **`Controllers`**
  - BoardController.java, který má funkci tahání figur, pravidla šachů a časovači.
- **`View`**
  - GameView.java vytvoří okno aplikace.
  - ChessBoardView.java vykreslí šachovnice s figurami.
- **`Figures`**
  - Třídy figur. Každá figura má logiku pohybu a zobrazení figury.
- **`Engine`**
  - StockFish.java. Engine pro předpověď tahu.
- **`Utils`**
  - Utils.java. Obsahuje pomocné funkce.

Třídy:

- **UML Diagram tříd:**



- **Main.java**

Je hlavní třida projektu, která zpustí aplikace. Metoda `main` vytvoří hru, která zatím vytvoří okno

- **ChessBoardView.java**

  Je hlavní třida vykreslování. Dědí od JPanelu. Třída kreslí šachovnici a figury na pozicích. Také kreslí všechny možné tahy a poslední tah. Také vytváří grafy a PNG obrázek šachovnici.

  - **paintComponent(Graphics g)**
    vykreslí šachovnice a figurky na ní.

- **GameView.java**

  Třida dědí od **JFrame** a tvoří okno 800x600 px, a volá **ChessBoardView**. Má menu, v které jsou dvě tlačítka **"Game"** a **"Help".** Help zatím prazdý, Game obsahuje ještě tři tlačítka:

  - `Restart` – začne novou hru bez restartování aplikace.
  - `Pause` – zastaví hru a časovač a vypne **drag & drop.**
  - `Export graf` – udělá PNG obrázek sloupcového grafu a uloží do cesty, kterou uživatel ukázal.
  - `Export PNG` – vytváří PNG obrázek šachovnice a figure na ní.

- **KilledFigureView.java**

  Tato třida dědí od **JPanel** a vytváří seznam zabitých figur.

- **Figure.java**

  Je abstraktní třida a hlavní třida pro kreslení figur. Dědí od `JPanelu`.

- **Queen.java, King.java, Knight.java, Bishop.java, Rook.java, Pawns.java**

  Tyto třídy představují implementaci jednotlivých figurek. Každá třída dědí od `Figure`. Figury Pawns májí logiku **„braní mimochodem"**. A figury King májí logiku **„rošady"**.

- **BoardController.java**
  BoardController obsahuje logiku hry, šach, mat, pat a proměnu. Tato třida implementuje MouseListener, který je používán pro Drag and drop.
- **StockFish.java**

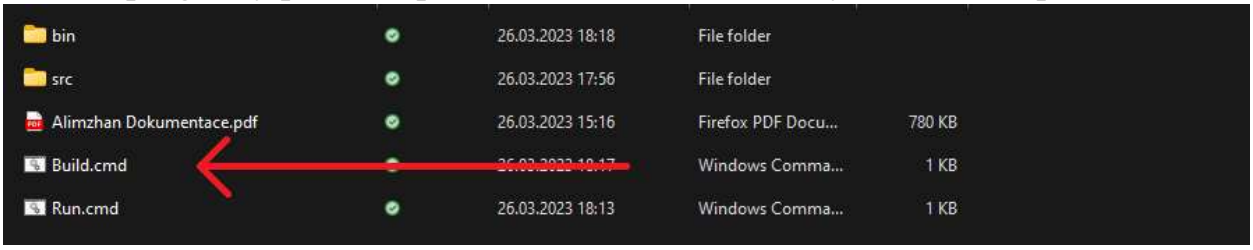
  Tato třida posila příkazy enginu **StockFish** a přijímá odpovědi.

- **Utils.java**
  Děla převod souřadnic od **StockFishu** do tvaru x, y souřadnic.

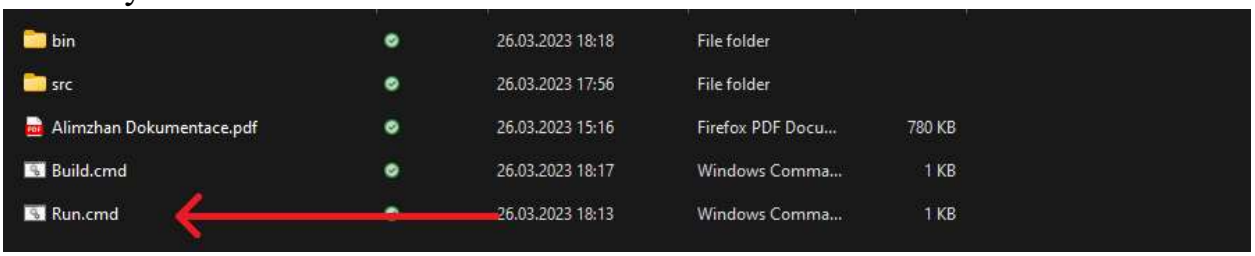
Uživatelská dokumentace:

- **Instalace:**

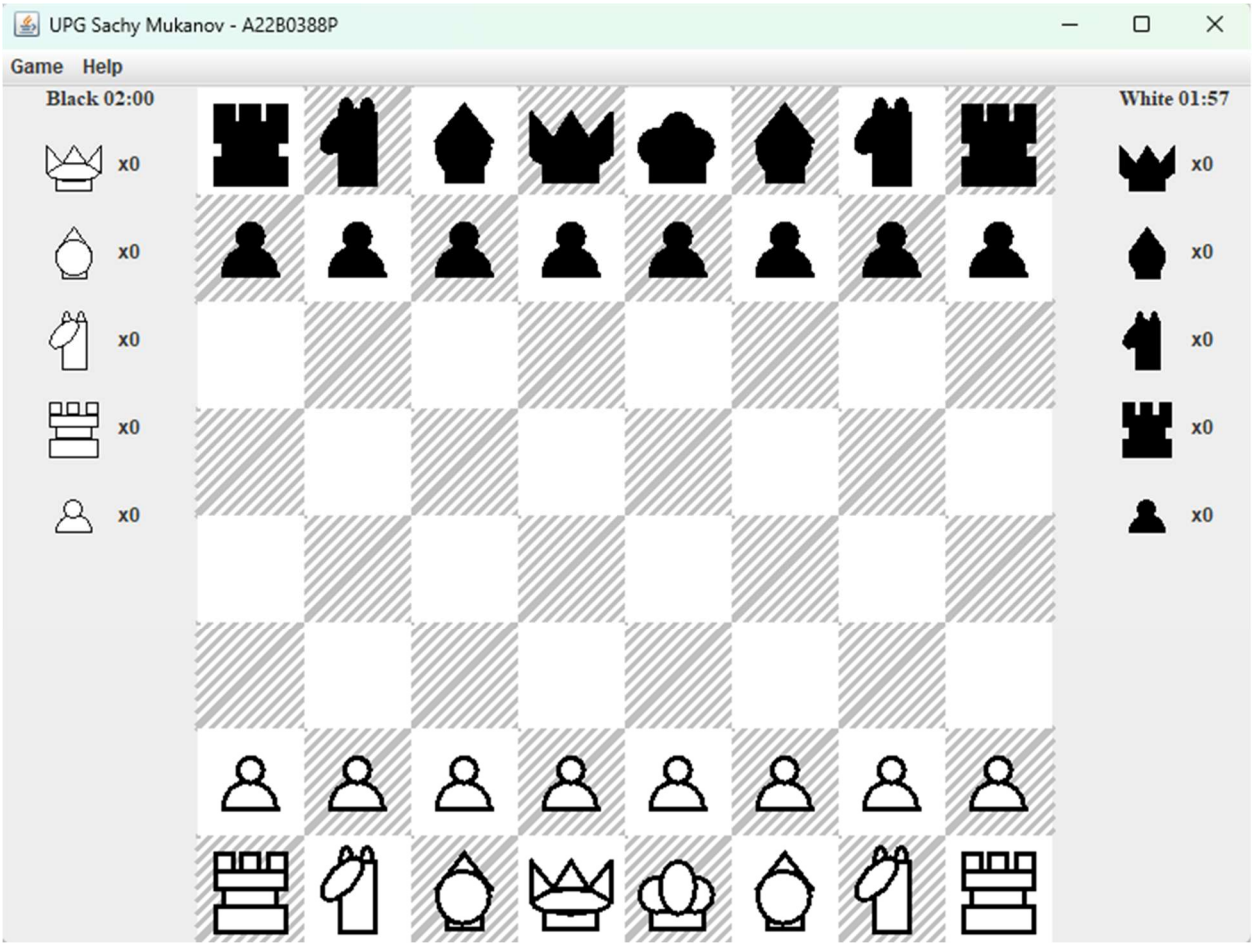
1. Překlad programy pomoci spuštění **`Bild.cmd`** ze složky semestrální práce



2. Spuštění aplikace pomoci spuštění **`Run.cmd`**
ze složky



- **Příklad GUI:**



Závěr:

Tento projekt je program pro hraní šachů s využitím knihovny **`Java Swing`** a principů **`OOP`**. Program představuje vizuální simulaci herní desky, na které jsou umístěny figury, které se mohou pohybovat.

Funkce programu zahrnují možnost změny velikosti okna, přičemž šachovnice a figurky se přizpůsobují velikosti okna.

Program obsahuje úplná pravidla šachů: braní mimochodem, rošáda a proměnná. Každý tah je animován a zprava od šachovnice je časovač pro bílého a černého hráče.

Program byl doplněn hlavním menu z výběrem hry proti bota, s použitím StockFish, a nebo proti člověka.

Při práce jsem se naučil používat knihovnu **`Jawa Swing`** a **`JFreeChart`**.