Gymnázium, Praha 6, Arabská 14
Arabská, Praha 6, 160 00

.

ROČNÍKOVÝ PROJEKT

Předmět: Programování

Téma: Šachy

Autor: Nikola Jankovič

Třída: 2.E

Školní rok: 2022/2023

Vyučující: Mgr. Jan Lána

Třídní učitel: Mgr. Blanka Hniličková

Čestné prohlášení:

Prohlašuji, že jsem jediným autorem tohoto projektu, všechny citace jsou řádně označené a všechna použitá literatura a další zdroje jsou v práci uvedené. Tímto dle zákona 121/2000 Sb. (tzv. Autorský zákon) ve znění pozdějších předpisů uděluji bezúplatně škole Gymnázium, Praha 6, Arabská 14 oprávnění k výkonu práva na rozmnožování díla (§ 13) a práva na sdělování díla veřejnosti (§ 18) na dobu časově neomezenou a bez omezení územního rozsahu.

V Praze dne: **……**

Nikola Jankovič

Anotace

Tato dokumentace popisuje program pro hraní šachu, který byl naprogramován v technologii JavaFX. Popisuje základní prvky uživatelského rozhraní, pravidla hry a způsob jakým je program implementován. Dále se zaměřuje na jednotlivé třídy a jejich funkce, včetně řešení logiky šachového tahu a zobrazení stavu šachovnice.

Annotation

This documentation describes a chess playing program that has been programmed using JavaFX technology. It describes the basic elements of the user interface, the rules of the game, and the way in which the program is implemented. It also focuses on the individual classes and their functions, including the solution to the logic of the chess move and the display of the state of the chessboard.

Zadání ročníkového projektu

Naprogramovat plaformu pro hraní šachu.

Úvod

Tato dokumentace popisuje program pro hraní šachu, který byl naprogramován v technologii JavaFX. Šachy jsou tradiční desková hra, která má své kořeny v Indii zhruba před 1500 lety a dodnes se těší velké oblibě po celém světě. Digitální podoba této hry umožňuje hráčům zahrát si šachy na počítači, bez nutnosti fyzické šachové sady.

Cílem této dokumentace je poskytnout ucelený přehled o funkcích a možnostech tohoto programu, který umožňuje uživatelům hrát šachy v digitální podobě.

Dokumentace popisuje jednotlivé třídy, funkce a metody kódu.

1. Použité technologie

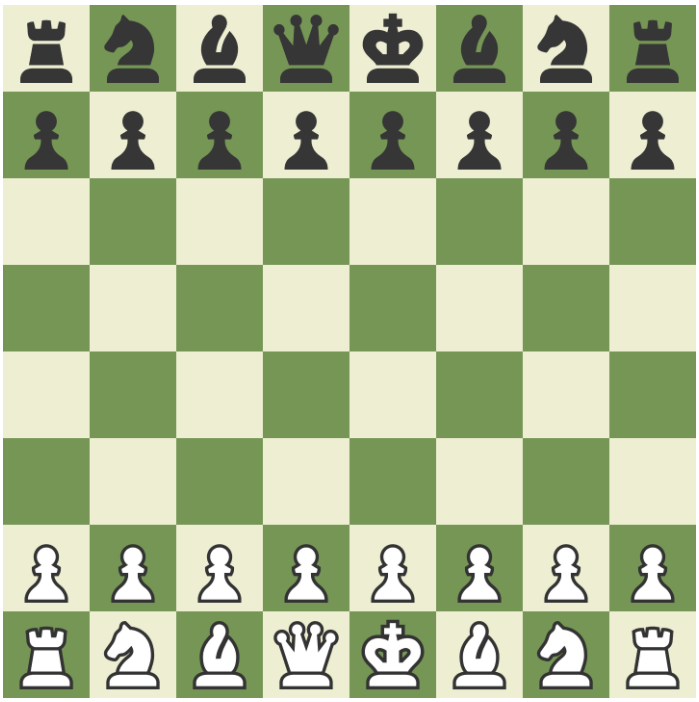
Pro vypracování této ročníkové práce jsem používal programovací jazyk Java a jeho rozšíření JavaFX. Program byl napsán ve vývojovém prostředí (IDE) IntelliJ IDEA.

2. Šachy

Šachy jsou desková hra pro dva hráče. Moderní forma šachu vznikla v 15. století v jižní Evropě a v posledních letech nabývá na popularitě obzvláště mezi mladšími generacemi.

Šachy se hrají na čtvercové desce rozdělené na 8x8 polí se střídajícími se barvami. Této desce se říká šachovnice. Každý hráč má 16 kamenů: + krále, 1 dámu/královnu, 2 věže, 2 střelce 2 jezdce a 8 pěšáků. Cílem hry je dát soupeřovi takzvaný mat. Mat nastane, když na krále útočí nějaká figurka protivníka a král nemá, jak tomuto útoku zabránit.

Na začátku hry jsou na šachovnici umístěny podle předem daných pravidel. Bílé kameny jsou umístěny dole na šachovnici tedy na 1. a 2. řadě a černé kameny na 7. a 8. V každém rohu je umístěna věž přímo vedle těchto věží jsou umístěny jezdci, vedle nich střelci a vedle střelců král a dáma. Pro odlišení rozmístění dámy a krále se často používá pravidlo: “Dáma ctí barvu.” To znamená, že bílá dáma bude umístěna na bílé/světlé pole a černá dáma na černé/tmavé pole.



Obrázek 1 - Šachovnice z chess.com

3. Program

Program je rozdělen do 10 tříd. Nejdůležitější a nejdelší třída se jmenuje GameEngine. V této třídě jsou funkce a metody, které řeší jak zobrazení šachovnice, tak i pravidla hry a její popis bude nejobsáhlejší. Mezi další třídy patří constants, Tiles a Piece. Poté jsou vytvořeny třída pro každý druh figurky, které jsou všechny potomkem třídy Piece.

3.1. Třída Tiles

Třída Tile je velmi jednoduchá třída, která vytvoří rectangle pro GUI a přiřadí mu barvu podle souřadnice. Tato třída zároveň obsahuje metody setPiece, getPiece a hasPiece, které slouží k pokládání a získávání kamenů z určitého pole.

3.2. Třída Piece

Třída Piece je třída, která vytvoří objekt pro GUI, přídá tomuto objektu atribut piece type, který bude definovaný v potomcích této třídy. Mezi další atributy patří oldX a oldY, mouseX a mouseY. OldX a oldY určuje souřadnice na kterých žeton původně ležel a mouseX a mouseY určuje souřadnice myši. Zároveň je v této třídě vytvořen základ pohybu pomocí setOnMousePressed, která získá původní souřadnice figurky a setOnMouseDragged, která zařizuje, že se žetonu při držení levého tlačítka pohybuje.

Součástí této třídy jsou také void funkce move a abortMove. Move změní atribut oldX a oldY a pak relokuje figurku na tyto souřadnice. AbortMove nezmění tyto atributy a relokuje žeton na původní oldX a oldY. Toto slouží k tomu, aby se figurky pohybovali na GUI.

```
56          setOnMousePressed(e -> {
57              mouseX = e.getSceneX();
58              mouseY = e.getSceneY();
59          });
60
61          setOnMouseDragged(e -> {
62              relocate(e.getSceneX() - mouseX + oldX, e.getSceneY() - mouseY + oldY);
63          });
64
65
66      }
67
68      public void move(int x, int y) {
69          oldX = x * tileSize + 0.25 * tileSize;
70          oldY = y * tileSize + 0.25 * tileSize;
71          relocate(oldX, oldY);
72      }
73
74      public void abortMove() {
75          relocate(oldX, oldY);
76      }
77
```

Obrázek 2 - lambda expressions, funkce move a abortMove ze třídy piece

Z této třídy poté máme potomky, pro každý druh figurky. Tyto potomci definují atribut pieceType a budou z nich ve třídě GameEngine tvořeny pravidla pro pohyb.

3.3. Třída GameEngine

Tato třída je zodpovědná za veškerou logiku hry. Na začátku třídy jsou vytvořeny proměnné, které budu poté používat v celé třídě. Asi nejvíc důležitá je proměnná board, která reprezentuje v kódu to, co se zobrazuje na obrazovce. V této třídě jsou veškeré funkce, které řeší původní umístění žetonů, zda je pohyb, který se hráč snaží udělat legální a zároveň řeší, zda je král v šachu nebo matu. V následujících kapitolách budou popsány jednotlivé metody.

3.3.1. Metoda placeTiles

Metoda placeTiles vytvoří šachovnici a na příslušná pole přidá žetony, které budou pomocí metod pohyb umístěny do středu odpovídajícího pole a budou pohyblivé.

3.3.2. Funkce pro kontrolu legálního pohybu

Veškeré následující funkce společně dohlíží na to, že je pohyb legální. Funkce commonRules slučuje funkce OB, která zabraňuje hráčům v přetáhnutí figurky mimo šachovnici a occupiedByMyPiece, která zkontroluje souřadnice kam chce hráč žeton přetáhnout a pokud je tam žeton stejné barvy, tak je tah nelegální. Tyto pravidla jsou společná pro všechny druhy žetonů a pro oba hráče.

```
662     private boolean queenLegalMove(int startX, int startY, int endX, int endY){
663
664         //spojení pravidel pro věž a střelce (diagonílní, horizontální nebo vertikální pohyb
665         if (startX != endX && startY != endY && Math.abs(startX - endX) != Math.abs(startY - endY)) {
666             System.out.println("pohyb nesplňuje podmínku směru ");
667             return false;
668         }
669
670         //vrací true pouze když jsou splněny funkce lineIsEmpty() a commonRules()
671         return lineIsEmpty(startX, startY, endX, endY) && commonRules(startX, startY, endX, endY);
672     }
673
674     private boolean rookLegalMove(int startX, int startY, int endX, int endY){
675
676         //kontroluje, že je pohyb vertikální nebo horizontální
677         if (startX != endX && startY != endY){
678             System.out.println("nesplňuje směr");
679             return false;
680         }
```

Obrázek 3 - příklad queenLegalMove a rookLegalMove

Poté jsou v kódu funkce legalMove pro každý druh figurky zvlášť a pokud je pohyb legální vrací true a jinak vrací false.

Například funkce kingLegalMove vezme kromě funkce common rules i původní hodnotu x a y a odečte jí od nové hodnoty x a y. Pokud toto čístlo v absolutní hodnotě je větší než 1, tak je vzdálenost moc velká pro krále a funkce vrací false.

3.3.3. Metody pohyb

Pro každý druh žetonu existuje metoda pohyb, která vytvoří objekty tříd King, Queen atd. Tato metoda obsahuje lambda expression setOnMouseReleased, která inicializuje proměnné startX, startY, endX a endY. Tyto proměnné se pak využívají pro kontrolu, zda je pohyb legální nebo ne. V metodě pro krále se ukládá zároveň pozice bílého a černého krále. Toto pak využívám ke kontrole toho, zda je král v šachu. Pokaždé, když se král pohne, tato hodnota se aktualizuje. Při každém legálním pohybu se v if podmínce zavolají boolean funkce checkIfKingAttacked, která vrací true, pokud nějaká figurka útočí na krále a moveDoesNotStopCheck, která vrací true pokud po pohybu je král pořád v šachu. Funkce moveDoesnNotStopCheck využívá proměnné whiteKingInCheck a blackKingInCheck, které aktualizují při zavolání checkIfKingAttacked. Proto je důležité volat funkci checkIfKingAttacked před moveDoesNotStopCheck aby se tato hodnota aktualizovala.

V tomto jsou všechny tyto metody stejné, akorát vytváří jiné objekty a volají funkci legalMove pro odpovídající figurku. Jediná rozdílná je metoda pro krále. Která zároveň aktualizuje královi souřadnice v proměnných whiteKingX, whiteKingY a blackKingX, blackKingY. Tyto proměnné jsou dostupné pro celou třídu a jsou volané v předtím zmíněné funkci checkIfKingAttacked pro získání pozice krále v jakékoliv metodě pohyb.

3.3.4. Funkce checkIfKingAttacked

Implementace útočení na krále byla asi nejkomplikovanější část celého programu. Pro zkontrolování šachu program získá lokaci krále a odpovídající objekt třídy Piece – tedy krále. Poté podle těchto souřadnic volá funkci isPieceAttacked. Tato funkce volá další 4 funkce, které postupně zkontrolují všechny možné pozice, ze kterých by mohla na krále vidět nějaká nepřátelská figurka.

Prvně se volá funkce pieceAttackedHorizontallyOrVertically, která projede všechny pole ve stejném řádku nebo sloupci. Pokud najde v nějaké nepřerušované linii nějakou nepřátelskou figurky, tak zkontroluje pieceType této figurky. Pokud je pieceType rook nebo queen, funkce vrací true.

Poté se volá funkce pieceAttackedDiagonally, která dělá to samé ale pro diagonály a vrací true v případě, že pieceType je bishop nebo queen.

Třetí v pořadí se volá funkce pieceAttackedByPawn. Tato funkce zjistí barvu krále, který je zrovna volán. Podle toho určí, jestli má kontrolovat pěšáky, které jsou na souřadnici y dále

```
172         if (piece.getColor().equals("white")){
173             checkY = checkY - 1;
174             try {
175                 if (board[checkX + 1][checkY].getPiece() != null &&
176                         board[checkX + 1][checkY].getPiece().getPieceType().equals("pawn") &&
177                         board[checkX + 1][checkY].getPiece().getColor().equals("black")) {
178                     System.out.println(piece.getColor() + " " + piece.getPieceType() + " is attacked by: "
179                             + board[checkX + 1][checkY].getPiece().getColor() + " "
180                             + board[checkX + 1][checkY].getPiece().getPieceType() + "from: "
181                             + (checkX + 1) + " " + checkY);
182                     return true;
183                 }
184                 if (board[checkX - 1][checkY].getPiece() != null &&
185                         board[checkX - 1][checkY].getPiece().getPieceType().equals("pawn") &&
186                         board[checkX - 1][checkY].getPiece().getColor().equals("black")) {
187                     System.out.println(piece.getColor() + " " + piece.getPieceType() + " is attacked by: "
188                             + board[checkX - 1][checkY].getPiece().getColor() + " "
189                             + board[checkX - 1][checkY].getPiece().getPieceType() + "from: "
190                             + (checkX - 1) + " " + checkY);
191                     return true;
192                 }
193             }catch (ArrayIndexOutOfBoundsException e){
194                 System.out.println("checked cordinates are OB");
```

Obrázek 4 - příklad funkce pieceAttackedByPawn pro bílého krále

nebo blíže hodnotě 0. Vzhledem k tomu, že 0 se nachází nahoře, tak pro bílého krále kontroluje pěšáky o 1 pole blíže k 0 a pro černého o 2 pole dále. Poté na této hodnotě y zjistí, jestli se nalevo nebo napravo od krále nachází pěšák nepřátelské barvy. Pokud ano, tak vrací true.

Jako poslední se volá funkce, které postupně kontroluje všech 8 polí ze kterých by na krále mohl vidět jezdec. Tyto podmínky jsou v blocích try-catch. Vzhledem k tomu, že na rozdíl od ostatních funkci nevyužívá for cyklů ale postupné kontroly každého možného pole. Takový způsob volání by mohl skončit s ArrayIndexOutOfBounds vyjímkou. Tyto bloky zařídí, že pokud se funkce bude snažit zkontrolovat pole co neexistuje tak to daný blok kódu přeskočí a neskončí chybou.

Závěr

V tomto ročníkovém projektu jsem vytvořil program, který umožňuje hrát šachy na jednom počítači. Nepodařilo se mi naprogramovat, aby hra automaticky skončila, pokud dojde k matu nebo speciální pravidlo rošádu. Největší problém bylo opravování chyb a přepisů v takto dlouhém kódu, kde se občas podmínky liší pouze jedním znaménkem. Další možný rozvoj by byla implementace těchto funkcionalit, aby hra fungovala doopravdy stejně jako na internetu.

Seznam Obrázků