VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY

FAKULTA INFORMAČNÍCH TECHNOLOGIÍ
ÚSTAV INTELIGENTNÍCH SYSTÉMŮ

FACULTY OF INFORMATION TECHNOLOGY
DEPARTMENT OF INTELLIGENT SYSTEMS

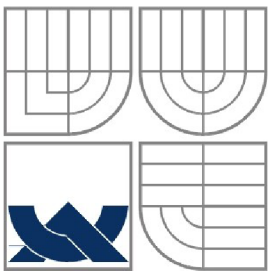
RÔZNE VARIANTY ŠACHOV

BAKALÁŘSKÁ PRÁCE
BACHELOR'S THESIS

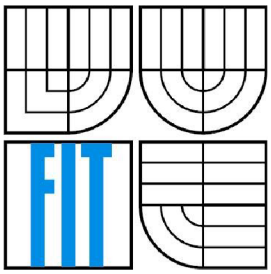
AUTOR PRÁCE
AUTHOR

MICHAL ĎURIŠ

BRNO 2010



VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY



FAKULTA INFORMAČNÍCH TECHNOLOGIÍ
ÚSTAV INTELIGENTNÍCH SYSTÉMŮ

FACULTY OF INFORMATION TECHNOLOGY
DEPARTMENT OF INTELLIGENT SYSTEMS

RŮZNÉ VARIANTY ŠACHŮ

VARIOUS CHESS VARIATIONS

BAKALÁŘSKÁ PRÁCE
BACHELOR'S THESIS

AUTOR PRÁCE
AUTHOR

MICHAL ĎURIŠ

VEDOUCÍ PRÁCE
SUPERVISOR

Ing. JAROSLAV ROZMAN

BRNO 2010

Abstrakt

Hlavním úkolem bakalářské práce bylo vytvoření aplikace, umožňujíci hraní člověka proti počítači s možností volby vybraných variant šachů. V úvodu práce klasifikuje existujíci šachové varianty do skupin a popisuje případné nutné změny ohodnocovací funkce oproti klasickým šachům. Následuje část popisujíci návrh a realizaci implementace aplikace a opis hlavních grafických užívateľských komponent. V závěru se nachází zhodnocení práce s návrhy umožňujícími zlepšení samotné aplikace.

Abstract

The main goal of this bachelor's thesis was to create an application, which offer an opportunity to play a game between human and computer with possibillity of choosing from various chess variations. The introduction classifies existing chess variations into the groups and describes eventuelly required changes of valuation function in comparison with classic chess variation. Next part is describes a suggestion and a realization of implementation and also the main graphical user compotents. The last part evaluates the thesis with suggestions how to improve the application even more.

Klíčová slova

Šach, šachové variace, umělá inteligence, Java , grafické užitelské rozhraní, ohodnocovací funkce

Keywords

Chess, chess variations, artificial intelligence, Java, graphical user interface, valuation function

Citace

Michal Ďuriš: Rôzne varianty šachov, bakalářská práce, Brno, FIT VUT v Brně, 2010

Rôzne varianty šachov

Prohlášení

Prohlašuji, že jsem tuto bakalářskou práci vypracoval samostatně pod vedením Ing. Jaroslava Rozmana.
Uvedl jsem všechny literární prameny a publikace, ze kterých jsem čerpal.

……
Michal Ďuriš
19. května 2010

Poděkování

Chtěl bych poděkovat Ing. Jaroslavovi Rozmanovi, za rady, návrhy rešení práce a ochotu při spolupráci.

© Michal Ďuriš, 2010.
Tato práce vznikla jako školní dílo na Vysokém učení technickém v Brně, Fakultě informačních technologií. Práce je chráněna autorským zákonem a její užití bez udělení oprávnění autorem je nezákonné, s výjimkou zákonem definovaných případů.

Obsah

1 Úvod

Šach ako hra má za sebou pomerne dlhú históriu. Počas tejto doby si ľudia pravidlá rôzne prispôsobovali, čím sa vytvoril priestor pre vznik alternatívnych šachových variácií, ktorých zmeny pramenili z potreby ľudí si šachové partie určitým spôsobom vylepšiť, ozvláštniť alebo spraviť atraktívnejšími. V súčastnosti sa zaradujú šachové varianty podľa ich hlavných špecifických znakov, líšiacich sa od šachov klasických, do troch skupín. Týmito znakmi sú:

- rôzny počet alebo tvar hracích polí na šachovnici (šachovnice iného ako štvorcového tvaru, napríklad trojuholníka alebo šesťuholníka, prípadne šachovnice štvorcového tvaru formátu 10 x 10, 8 x 10 a podobne)
- zmena šachovej sady – pridanie nových figúrok, prípadne zámena figúrky z pôvodnej sady figúrkou špeciálnou alebo odobratie určitého typu figúrky
- zmena pravidiel – vzájomného vyhadzovania a možnosti pohybu figúrok, prípadne rozdielne podmienky víťazstva a podobne

Konečný počet všetkých možných šachových variantov nie je možné určiť. Pričom v knihe [1] autor D. B. Pritchard, zaoberajúci sa šachom, odhaduje, že celkový počet existujúcich variantov presahuje číslo dvetisíc. Medzi najznámejšie šachové varianty patria Fisherove šachy, od bývalého majstra sveta v šachu Roberta Fishera, žravé šachy, význačné nutnosťou neustáleho vyhadzovania súperových figúrok a maharadžov šach, známy špeciálnou figúrkou maharadža. Podrobnejšie sa budem jednotlivými variantami zaoberať v kapitolách 2.1.
Kapitola je voľne prevzatá z [2]

2 Klasifikácia šachových variantov

V tejto kapitole sa bude pojednávať o základnom rozdelení rôznych šachových variantov do skupín podľa podobnostných znakov. Pri každej variante zaradenej do určitej skupiny bude napísaný krátky popis vystihujúci špecifické pravidlá pre daný variant a odlišnosť voči ostatným šachovým variantom a variantom klasického šachu.

Šachové varianty sú rozdelené do troch hlavných skupín, ktoré vyjadrujú základné rozdelenie týkajúce sa rozostavenia figúrok a prípadných zmien v šachových sadách. V hlavných skupinách nasleduje rozdelenie do špecifických podskupín zoskupujúcich varianty, ktorých nosné pravidlá sú si podobnostne najbližšie.

Skupinové rozdelenie som vytvoril z dôvodu systematického usporiadania a pre lepšiu prehľadnosť mnou vybraných šachových variantov.
Pravidlá sú prevzaté z [3].

2.1 Rozdelenie šachových variantov

1. Šachy so zachovanou základnou pozíciou uloženia a sadou figúrok

a) Šachy so zmenou pravidiel

- **Cyklický šach :**
  - Figúrky, ktoré hráč zajme svojmu súperovi, sa objavia v poli Extra figúrky a v opačnej farbe – stanú sa teda novými figúrkami hráča, ktorý ich zajal. Takto získané figúrky môžu byť umiestnené na ľubovoľné prázdne pole na šachovnici a síce jedna figúrka za jeden ťah.

- **Žravý šach :**
  - Branie súperových figúrok je povinné. Hráč musí hrať len takou figúrkou, ktorá môže zajať súperovu figúrku. Iba v prípade, že súper nemá takúto možnosť, môže ťahať kamkoľvek.

- **Vymretý šach :**
  - Vyhráva ten, kto zajme všetky súperove figúrky jedného druhu, to znamená zajatím všetkých 8 pešiakov hra konči a taktiež hra končí pri zajatí obidvoch strelcov alebo jazdcov.

- **Trojšachový šach :**
  - Hráč vyhráva okrem štandardných pravidiel aj pri trojnásobnom šachu súperovi v jednej partií.

- **Atómový šach :**
  - Ak je figúrka zajatá, vybuchne v ničivej explózii a zničí figúrku, ktorá vykonala zajatie, zajatú figúrku a všetky figúrky nachádzajúce sa v bezprostrednom okolí figúrky.
- **Berlínsky šach :**
  - Pešiaci sa pohybujú diagonálne a zajímajú súperove figúrky vertikálne vpred.
- **Jazdcový šach :**
  - Ľubovoľná figúrka, ktorá je krytá jazdcom rovnakej farby, môže k vlastným možnostiam pohybu pridať aj pohyby špecifické pre jazdca
- **Recyklovaný šach :**
  - Hráč smie zajať svoju vlastnú figúrku, okrem kráľa, a vrátiť ich späť na šachovnicu v nasledujúcich ťahoch.
- **Valcový šach :**
  - Stĺpce A a H sú spojené, takže hráč ich môže používať, ako keby stĺpec A bol hneď vedľa stĺpca H a naopak.

b) Šachy s novou figúrkou

- **Maharadžov šach :**
  - Maharadža je nová figúrka spájajúca vlastnosti kráľa, dámy a jazdca. To znamená, že sa pohybuje ako dáma a jazdec dohromady, nesmie vstúpiť na pole ohrozené súperovými figúrkami a ak je napadnutá, musí ustúpiť do bezpečnej pozície alebo napádajúcu figúrku zajať.
- **Amazonský šach :**
  - Dáma je nahradená novou figúrkou nazývanou amazonka. Amazonka kombinuje ťahy dámy a jazdca.

c) Šachy ovplyvňované náhodným činiteľom

- **Kockové šachy :**
  - Pred každým ťahom sa hádže kockou a číslo hodené na kocke určuje, ktorou figúrkou sa bude tiahnuť. V prípade premeny pešiaka, kocka rozhoduje na akú figúrku sa pešiak môže premeniť.
- **Behemotské šachy :**
  - Behemoth sa po každom ťahu hráča premiestňuje. Môže sa posunúť o 1- 4 polia, kde počet polí je náhodne určený, v ľubovoľnom smere, ktorý je taktiež určený náhodne. Behemoth zničí všetky figúrky, na ktoré narazí po ceste.

2. Šachy s rozdielnym základným pozičným postavením

a) Šachy s rovnakou sadou figúrok

- **Rohový šach :**
  - Králi sú umiestnení do pravých dolných rohov šachovníc vždy z pohľadu hráča. Ostatné figúrky okrem pešiakov sa náhodne rozmiestnia na prvom riadku. Jeden strelec sa musí postaviť na biele pole a druhý na čierne pole. Základné pozície oboch hráčov musia byť symetrické podľa stredu šachovnice.
- **Fisherov šach :**
  - Pre každú hru je vytvorená nová náhodná počiatočná pozícia rozostavenia figúrok, avšak musí splňovať nasledujúce podmienky. Kráľ musí stáť medzi oboma vežami, jeden strelec musí byť na bielom poli a druhý na poli čiernom a biela a čierna základná pozícia sú symetrické.
- **Leganov šach :**
  - Figúrky oboch hráčov sa rozmiestnia do špecifickej začiatočnej polohy na šachovnici s centrom v pravých dolných rohoch z pohľadu každého hráča.
- **Voľne pozičný šach a pozičný šach :**
  - Každý hráč si pred začiatkom hry sám zvolí svoje počiatočné rozmiestnenie figúrok na šachovnici.

b) Šachy s rozdielnou sadou figúrok

- **Prepadový šach :**
  - Biely hrá s obvyklou sadou šachových figúrok, zatiaľ čo čierny hrá iba s armádou 32 pešiakov rozmiestnených v špecifickom základnom rozostavení.
- **Závod kráľov :**
  - Každý hráč začína hru s 1 kráľom, 1 dámou, 2 vežami, 2 strelcami a 2 jazdcami. Figúrky obidvoch farieb sú umiestnené na prvé dva riadky rovnakej strany. Cieľom hry je dosiahnuť vlastným kráľom poslednú radu pomocou bežných pravidiel hry šach pre pohyb figúrok.
- **Masakrové šachy :**
  - Hráč začína s 8 dámami, 8 vežami, 8 strelcami a 8 jazdcami. Z toho vyplýva že celá šachovnica je zaplnená figúrkami. Branie figúrok je v hre povinné a hráč, ktorý nemôže vykonať v hre platný ťah, čiže zajať súperovu figúrku, prehráva hru.

- **Opevnený šach :**
  - Každý kráľ je umiestnený do pravého dolného rohu z pohľadu každého hráča a pred každého kráľa sú ešte naviac umiestnení traja pešiaci.

3. Šachy na rozdielnej šachovnici

a) Šachy s novou figúrkou

- **Veľký šach :**
  - Hrá sa na šachovnici o veľkosti 10 x 10 polí. Obsahuje dve nové figúrky a to maršala, ktorý je kombináciou veže a jazdca a kardinála, ktorý je kombináciou strelca a jazdca.
- **Casablancove šachy :**
  - Hrá sa na šachovnici o veľkosti 10 x 8 polí. Obsahuje dve nové figúrky a to arcibiskupa, ktorý je kombináciou strelca a jazdca a kancelára, ktorý je kombináciou veže a jazdca.

b) Šachy so zmenou klasickej sady

- **Los Alamos šachy :**
  - Hrá sa na šachovnici o veľkosti 6 x 6 polí so všetkými klasickými šachovými figúrkami okrem strelcov.
- **Kockové šachy 10 x 10 :**
  - Hrá sa na šachovnici o veľkosti 10 x 10 polí a každý hráč začína s tromi kráľmi namiesto jedného.

3 Ohodnocovacia funkcia a jej zmeny v závislosti na šachovom variante

Kapitola sa zaoberá rozdielmi ohodnocovacích funkcií šachových variantov oproti ohodnocovacej funkcií šachov klasických.

Využívanie ohodnocovacích funkcií pri metódach hrania hier je dôsledkom toho, že úplné prehľadanie stavového priestoru pri zložitých hrách, ku ktorým šach jednoznačne patrí, je nemožné, preto sa prehľadáva stavový priestor len do určitej hĺbky, čo v praxi znamená prehľadávanie vopred daného počtu ťahov. A keďže nie je možné zaručiť, že v danej prehľadávanej hĺbke je možné rozhodnúť o riešiteľnosti, poprípade neriešiteľnosti daného uzla, je nutné tieto uzly určitým spôsobom ohodnotiť, k čomu slúžia práve ohodnocovacie funkcie.

Ohodnocovacie funkcie sú v podstate celočíselné hodnotiace funkcie, pri ktorých kladné ohodnotenie vnútorného uzla znamená smer k výhre a naopak záporné ohodnotenie vypovedá o smerovaní k prehre. Samotná výhra alebo prehra sú hodnotené maximálnymi hodnotami uvažovaného intervalu. Jedným z algoritmov využívajúcich ohodnocovaciu funkciu je algoritmus MiniMax.

V nasledujúcej časti sú popísané východiskové pravidlá ohodnocovacej funkcie klasických šachov a na záver sú určené potrebné zmeny klasickej ohodnocovacej funkcie jednotlivo pre každú šachovú variantu.

Pri tvorbe úvodnej časti tejto kapitoly bolo čerpané z [4]

3.1 Návrh zmien ohodnocovacej funkcie

Veľká rozmanitosť šachových variant je dôsledkom toho, že na ich implementáciu nebude postačujúca jedna ohodnocovacia funkcia. Ale keďže každý variant má určité spoločné znaky s klasickým šachom, rozhodol som sa, že budem vychádzať z ohodnocovacej funkcie šachov klasických. Keďže moja práca je pokračovaním minuloročnej bakalárskej práce [5], ktorej témou boli práve klasické šachy, dovolil som si vychádzať z kolegovej ohodnocovacej funkcie, keďže jej výsledky boli dostačujúce.

3.1.1 Ohodnocovacie pravidlá klasických šachov:

- ➔ Záporne sa hodnotí skoré rozvinutie kráľovnej z dôvodu zlej strategickej pozície ľahkých figúrok.
- ➔ Kladne sa hodnotia veže na súperovom predposlednom riadku. Predpokladá sa, že na poslednom stojí súperov kráľ.
- ➔ Kladne sa hodnotia veže na otvorených a polootvorených stĺpcoch.
- ➔ Záporne sa hodnotia strelci a kone na zadných pozíciach svedčiacich o zlej strategickej rozvinutosti figúr.
- ➔ Kladne sa hodnotia kone na centrálnych poliach (D4, D4, F4, F5) a strelci na hlavných diagonálach.
- ➔ Hodnotí sa miera zablokovanosti strelcov.
- ➔ Kladne sa hodnotí rozvíjanie pešiakov na centrálnych pozíciach a veľmi významne je hodnotené pokiaľ pešiak dôjde na druhú stranu šachovnice.

3.1.2 Zmeny ohodnocovacej funkcie

Vykonané zmeny vychádzajú z ohodnocovacej funkcie klasického šachového variantu, ktorá je popísaná v **kapitole 3.1.1** a sú zoradené podľa zaradenia do skupín určených v **kapitole 2.1**. Ide o predbežný návrh zmien ohodnocovacích funkcií na základe pravidiel jednotlivých šachových variantov.

Skupina 1.a

Vymretý šach – pridanie pravidla kladného ohodnotenia možnosti vyhodenia súperových párových figúrok (jazdec, veža, strelec).

Ostatným variantom v tejto skupine je postačujúca ohodnocovacia funkcia klasických šachov.

Skupina 1.b

Maharadžov šach – pridanie pravidla hodnotenia hĺbky figúrky maharadža v poli šachovnice
Amazonský šach – pridanie pravidla záporného ohodnotenia skorého rozvinutia amazonky

Skupina 1.c

Na varianty tejto skupiny dostatočne postačuje ohodnocovacia funkcia klasických šachov.

Skupina 2.a

Na varianty tejto skupiny dostatočne postačuje ohodnocovacia funkcia klasických šachov.

Skupina 2.b

Prepadový šach – pridanie pravidla kladného hodnotenia pozície pešiaka na súperovom predposlednom riadku

Závod kráľov – pridanie pravidla kladného hodnotenia pozície kráľa blížiacej sa cieľovej strane. Čím bližšie k cieľovému riadku, tým vyššie kladné hodnotenie.
– kladné ohodnotenie blokovania kráľa vežou, strelcom, jazdcom a dámou
– záporné ohodnotenie neblokovania kráľa vežou, strelcom, jazdcom a dámou
– zrušenie všetkých východiskových pravidiel klasického šachu

Masakrové šachy – zrušenie pravidla týkajúceho sa pešiakov
– zrušenie pravidla záporného hodnotenia skorého rozvinutia kráľovnej
– zrušenie pravidla kladného ohodnotenia pozície veže na predposlednom súperovom riadku
– zrušenia pravidla záporného ohodnotenia pozície strelca a jazdca na zadných radoch.

Pre **ostatné varianty** tejto skupiny je postačujúca ohodnocovacia funkcia klasických šachov.

Skupina 3.a

Veľký šach – úprava pravidla o kladnom hodnotení veže na súperovom predposlednom riadku na kladné hodnotenie veže na pozícií tretieho riadka z pohľadu súpera. Predpokladá sa, že súperov kráľ je na predposlednom riadku.
– úprava pravidla o kladnom hodnotení jazdca na centrálnych poliach, ktoré sú v danej variante na D4,D5,E4,E5 a F4,F5.

Pre **ostatné varianty** je postačujúca ohodnocovacia funkcia klasických šachov.

Skupina 3.b

Los Alamos šach – zrušenie pravidiel týkajúcich sa strelcov

Pre **ostatné varianty** tejto skupiny je postačujúca ohodnocovacia funkcia klasických šachov.

4 Návrh a implementácia aplikácie šachových variantov

Kapitola pojednáva o jednotlivých krokoch postupu návrhu a následnej implementácií. Prvým krokom bude výber šachových variantov k implementácií, následne bude popísaný hlavný riadiaci činiteľ. Taktiež budú ozrejmené možné herné módy aplikácie a spomenuté doplnkové triedy využívané v aplikácií.

Pre implementáciu aplikácie som zvolil objektovo orientovaný programovací jazyk Java™, o ktorom je možné sa viac dozvedieť na [6]. Pri výbere programovacieho jazyka boli mojimi hlavnými kritériami jednoduchosť, nezávislosť na architektúre, dobrý výkon a prenositeľnosť, ktoré práve Java™, spĺňa. Podpora viacvláknového programovania sa mi stala taktiež dobrou pomôckou. Ako vývojové prostredie som používal NetBeans kvôli editoru grafického užívateľského rozhrania, podporujúceho volný design. Viac o NetBeans je možné sa dozvedieť na [7].

Využíval som štandardne poskytované knižnice, taktiež externú knižnicu pre prácu s databázou MySQL a taktiež mnou navrhnutú knižnicu obsahujúcu nové prvky užívateľského rozhrania, ktoré sú spomenuté v kapitole 5.3.

Hlavnou riadiacou triedou aplikácie je trieda BaseFrameWindow.java, ktorá poskytuje navigáciu medzi ponúkanými funkciami aplikácie. Hlavnou triedou šachových hier je trieda ChessGame.java, ktorá zabezpečuje inicializáciu riadiacej jednotky a všetkých potrebných subsystémov a následne interakciu s užívateľom pre samotné hranie.

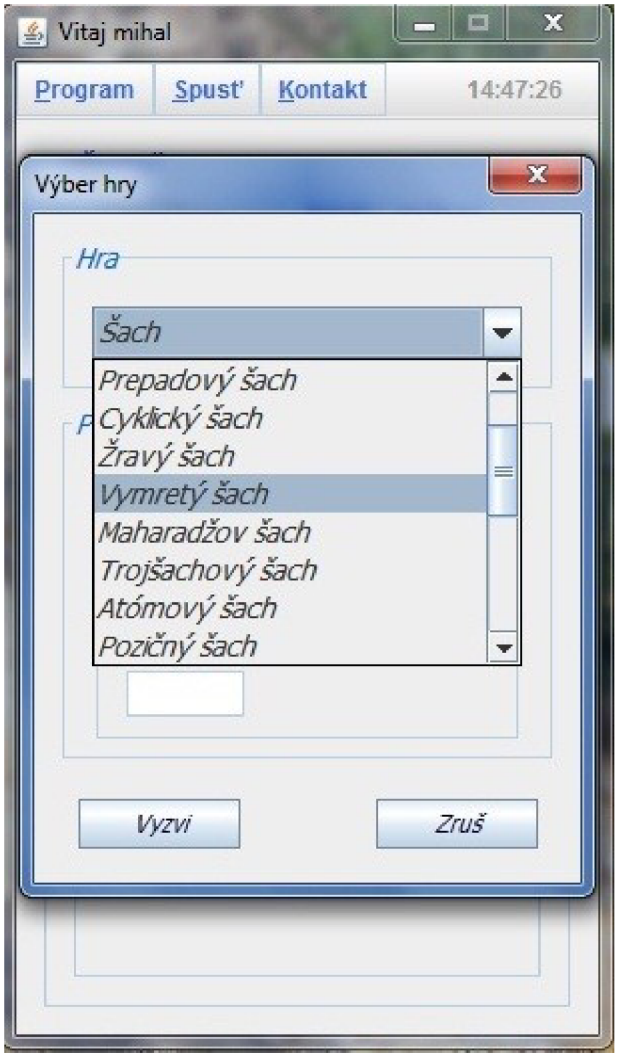
4.1 Výber šachových variantov

Pri výbere šachových variantov pre implementáciu som vychádzal z variantov popísaných v kapitole 2.1. Určil som si, že naimplementujem varianty hrané na štandardných rozmeroch šachovnice predstavujúcich veľkosť 8 x 8 polí. Snažil som sa vybrať hry tak, aby každá jedna reprezentovala unikátnu ideu šachovej varianty a to použitie špeciálnej figúrky, zmien štandardných pravidiel, pridanie ovplyvnenia hry náhodným činiteľom alebo prípadnou zmenou celého princípu šachov. Celkový počet zvolených šachových variantov je rovný počtu deväťnásť, ktoré spolu s implementáciou štandardných šachov tvoria dvadsať možných hier.

Pre zlepšenie prehľadnosti som pre reprezentáciu vybraných šachových variantov vytvoril rozhranie *ChessVariationsInteface.java,* nachádzajúceho sa v balíčku *client.transport*, kde každý jeden variant predstavuje celočíselnú statickú konštantu. Rozhranie je využívané pri výbere varianty a na to nadväzujúcej správnej inicializácii riadiaceho činiteľa.
Obsahuje nasledujúce varianty:

- variant – názov konštanty = hodnota
- **šach** – STANDARD_CHESS = 100
- **rohový šach** – CORNER_CHESS = 101
- **opevnený šach** – FORTRESS_CHESS = 102
- **prepadový šach** – HORDE_CHESS = 103
- **cyklický šach** – LOOP_CHESS = 104
- **žravý šach** – ANTI_CHESS = 105
- **vymretý šach** – EXTINCTION_CHESS = 106
- **maharadžov šach** – MAHARAJAH_CHESS = 107
- **trojšachový šach** – THREE_CHECKS_CHESS = 108
- **atómový šach –** ATOMIC_CHESS = 109
- **pozičný šach** – SCREEN_CHESS = 110

- **volne pozičný šach** – CRAZY_SCREEN_CHESS = 111
- **amazonský šach** – AMAZON_CHESS = 112
- **berlínsky šach** – BEROLINE_CHESS = 113
- **fisherov šach** – FISHER_RANDOM_CHESS = 114
- **leganov šach** – LEGAN_CHESS = 115
- **kockové šachy** – DICE_CHESS = 116
- **behemotský šach** – BEHEMOTH_CHESS = 117
- **závod kráľov** – RACING_KINGS = 118
- **masakrový šach** – MASSACRE_CHESS = 119



Obrázok 4.1: Výberový box šachových variantov

4.2 Návrh riadiaceho činiteľa

Pri veľkej rozmanitosti šachových variantov som musel vymyslieť riadiaceho činiteľa, ktorý by riadil všetky ostatné triedy zabezpečujúce ovládanie virtuálnej šachovnice, generovanie pohybu figúrok po šachovnici, kontrolu šachu, matu a patu a v neposlednom rade triedu zabezpečujúcu chod umelej inteligencie.

Na riešenie daného problému som využil vlastností objektovo orientovaného programovania a to dedičnosť a polymorfizmus, pričom som si vytvoril abstraktnú triedu *TransportAbstractClass.java*, nachádzajúcu sa v balíčku *client.transport*, ktorá poskytuje množinu riadiacich inštančných premenných a metód, vďaka ktorým je možné dynamicky meniť šachové pravidlá, sady a pohyb figúrok.

4.2.1 Riadiace inštančné premenné

Jednotlivé triedy šachových variantov dedia od abstraktnej triedy *TransportAbstractClass.java*, čím získajú množinu riadiacich prvkov, ktoré sú počiatočne nastavené na boolean hodnotu false, čo znamená vypnutie daného riadiaceho prvku. Pri implementácii varianty je teda postačujúce zapnúť len tie riadiace prvky, ktoré potrebujeme. Výber je nasledujúci:

Prvky riadenia pre kontrolu rošády a pre triedu *ŠachMatPat.java*, zabezpečujúcej kontrolu šachu, matu a patu

- ***sachEnabled*** – kontrola šachu
- ***matEnabled*** – kontrola matu
- ***patEnabled*** – kontrola patu
- ***rosadeEnabled*** – kontrola vykonania rošády

Pri ponechaní vypnutého jedného z vyššie spomínaných prvkov, je nutné určiť, pre ktorého hráča dané obmedzenie platí a to priradením do prvku ***controlDisabledForPlayer***, ktorý je typu String, nasledujúcou hodnotou:

- ***N*** – pre obidvoch hráčov
- ***B*** – pre hráča bielych figúrok
- ***C*** – pre hráča čiernych figúrok

Prvky riadenia pre integrovanie špeciálnej figúrky do štandardnej šachovej sady

Nastavením ***specialFigureEnabled***, ktorý je typu String, nasledujúcou hodnotou:

- ***N*** – pre žiadneho hráča
- ***B*** – pre hráča bielych figúrok
- ***C*** – pre hráča čiernych figúrok
- ***O*** – pre obidvoch hráčov

Pri určení, ktorý hráč, prípadne hráči, využívajú špeciálnu figúrku je nutné následne určiť aký typ vlastností bude daná figúrka reprezentovať. Nastavenie sa vykoná priradením príslušnej boolean hodnoty do nasledujúcich inštančných premenných:

- ***kingPowerOfSpecialFigure*** – špeciálna figúrka s vlastnosťou kráľa
- ***horsePowerOfSpecialFigure*** – špeciálna figúrka s vlastnosťou jazdca
- ***shooterPowerOfSpecialFigure*** – špeciálna figúrka s vlastnosťou strelca
- ***towerPowerOfSpecialFigure*** – špeciálna figúrka s vlastnosťou veže

Vlastnosti je možné medzi sebou kombinovať, čím vzniká variabilita tvorby špeciálnych figúrok.

Prvky riadenia pre ohodnocovaciu funkciu umelej inteligencie

Jedná sa o ohodnocovaciu funkciu štandardných šachov, ktorá je spomínaná v kapitole 3.1.1.

- ➔ ***firstRuleEnabled*** – zapnutie/vypnutie záporného ohodnotenia kráľovnej
- ➔ ***secondRuleEnabled*** – zapnutie/vypnutie kladného ohodnotenia veže na druhom súperovom riadku
- ➔ ***thirdRuleEnabled*** – zapnutie/vypnutie kladného ohodnotenia veže na otvorených a polootvorených stĺpcoch
- ➔ ***fourthRuleEnabled*** – zapnutie/vypnutie kladného ohodnotenia koní na centrálnych pozíciach
- ➔ ***fifthRuleEnabled*** – zapnutie/vypnutie záporného ohodnotenia koní na zadných pozíciach
- ➔ ***sixthRuleEnabled*** – zapnutie/vypnutie kladného ohodnotenia strelcov na hlavných diagonálach
- ➔ ***seventhRuleEnabled*** – zapnutie/vypnutie záporného ohodnotenia strelcov na zadných pozíciach
- ➔ ***eigthRuleEnabled*** – zapnutie/vypnutie určenia miery zablokovania strelcov
- ➔ ***ninthRuleEnabled*** – zapnutie/vypnutie kladného ohodnotenia pešiakov na centrálnych pozíciach
- ➔ ***tenthRuleEnabled*** – zapnutie/vypnutie kladného ohodnotenia dovŕšenia pešiakom opačnej strany

Prvok kontroly a počítania počtu obdržaných šachov od súpera

Pri zapnutí danej kontroly, nastavením príslušnej hodnoty do inštančnej premennej ***sachCounterEnabled***, je nutné uviesť celým číslom maximálny možný počet obdržaných šachov od súpera, nastavením premennej ***sachMaxCount***.

Po dosiahnutí maximálneho počtu šachov, daný hráč, u ktorého bol počet presiahnutý, hru prehráva.

Prvky riadenia zmien šachových pravidiel a možnosti pohybu šachových figúrok

- ➔ ***masacreEnabled*** – reprezentuje prioritu vyhadzovania súperových figúrok v triede UI.java
- ➔ ***returnKilledFigureEnabled*** – možnosť vrátiť vyhodenú figúrku späť do hry
- ➔ ***diceEnables*** – ovplyvňovanie hry náhodným činiteľom
- ➔ ***bonusFigureEnabled*** – integrovanie nezávislej figúrky

4.2.2 Riadiace inštančné metódy

Okrem riadiacich premenných sa dedia z abstraktnej triedy *TransportAbstractClass.java* riadiace metódy, ktoré zabezpečujú kontrolu zmien pravidiel na šachovnici, platnosť a generovanie zmeny pohybu figúrok a výkon neštandardných udalostí v priebehu hry.

Metódy kontroly platnosti vykonaného ťahu sú implicitne nastavené na boolean hodnotu true, ktorá reprezentuje platný vykonaný ťah. Metódy zastrešujúce pešiakov majú predvolené správanie klasického šachového variantu. Zostávajúce metódy implicitne nevykonávajú žiadnu činnosť. Pri implementácií šachového variantu je teda možné využiť polymorfizmu, čo znamená požadovanú metódu prepísať a doplniť vhodnou implementáciou. Poskytované sú:

- **Kontrolné metódy**
  - ***isGoodLimitFigureMove*** – kontrola správneho vykonania ťahu. Sprístupnenie metódy je možné priradením boolean hodnoty **true** a to pre umelú inteligenciu do inštančnej premennej ***figureMoveLimitUIEnabled*** a pre užívateľskú kontrolu do premennej ***figureMoveLimitUserEnabled***

  - ***isSoldierOnEnd*** – kontrola dosiahnutia pešiakom opačnej strany šachovnice.

- ***checkEndOfGame*** – kontrola dovŕšenia konca hry. Sprístupnenie metódy je pomocou inštančnej premennej ***possibleContinueEnabled***

- **Metódy generujúce zoznam ťahov**
  - ***soldierMoves*** – zoznam možných ťahov pešiakom
  - ***soldierTakeMovesList*** – zoznam možných braní súperových figúrok pešiakom

- Metódy obsluhujúce udalosti
  - ***setNewFigureForSoldier*** – výber a vloženie novej figúrky na šachovnicu po dosiahnutí opačnej strany pešiakom
  - e***xecuteMultiKill*** – vykonanie viacnásobného vyhodenia figúrok. Sprístupnenie metódy je pomocou inštančnej premennej ***multiKillEnabled***
  - ***throwDice*** – vykonanie hodu kockou. Metóda je prístupná v prípade, ak je spustené ovplyvnenia hry náhodným činiteľom

4.3 Módy hrania šachových partií

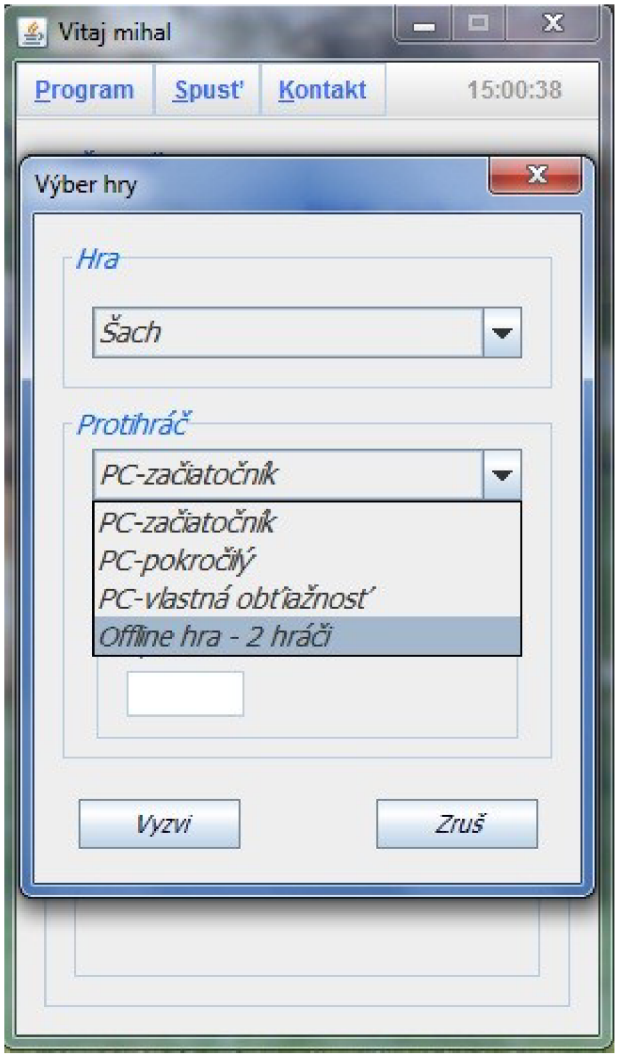
Pre aplikáciu som zvolil architektúru klient – server, čo mi umožnilo vytvoriť možnosť hrania šachových variantov po sieti medzi užívateľmi. Takže k módu hrania hier medzi dvomi hráčmi a medzi hráčom a počítačom, mi pribudla možnosť hry po sieti. Celá aplikácia pracuje v dvoch režimoch.

Prvým režimom je režim online, kedy užívateľ má prístup k serveru. V tomto režime má server na starosti všetky činnosti s fungovaním aplikácie a to registráciu užívateľov, ich následné prihlasovanie, odosielanie online užívateľov pre možnosť chatu alebo práve hrania šachových partií. Viac o možnosti hrania po internete je popísané v kapitole 4.3.3. Okrem režimu hrania online je užívateľovi samozrejme umožnené hrať aj v móde dvoch hráčov na rovnakom PC a taktiež proti počítaču. Server všetky informácie získava a zapisuje do MySQL databázy.

Druhým režimom je režim offline, ktorý nastáva v prípade neprístupnosti serveru. Aplikácia automaticky rozpozná, že došlo k tomuto prípadu a prepne sa do režimu offline. Všetky online funkcie sú neprístupné. Registrácia a prihlasovanie sú riešené lokálne, pomocou databázového xml súboru. V tomto prípade je možné hrať šachové partie len v móde dvoch hráčov na rovnakom PC a proti počítaču.

4.3.1 Mód dvoch hráčov v offline režime

Pozostáva z hry dvoch hráčov hrajúcich na jednej šachovnici. Ako je možné vidieť na **obrázku 4.2** daný mód zvolíme výberom možnosti z ponukového boxu oponentov ako variantu ***Offline hra – 2 hráči***. Po vybraní šachovej varianty a potvrdení výberu tlačítkom *Vyzvi*, sa začne šachová partia.

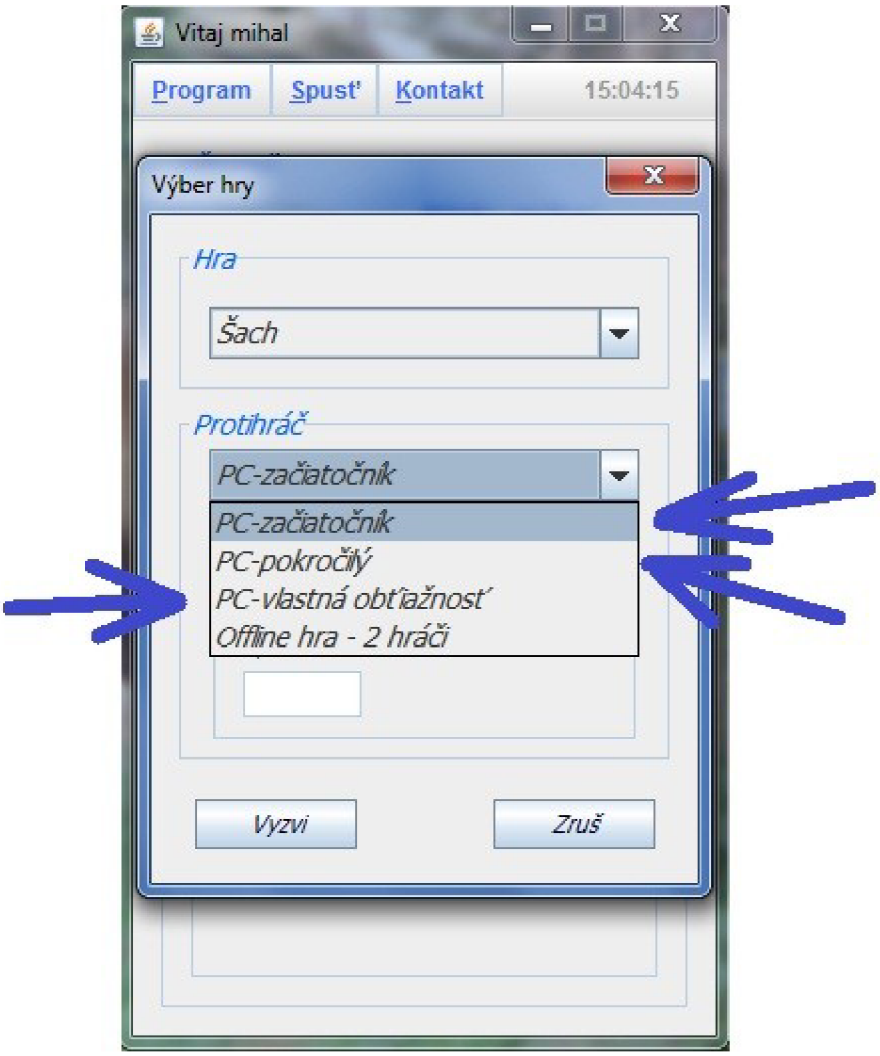


Obrázok 4.2

4.3.2 Mód hráča proti počítaču

Hra proti počítaču je štandardom, ktorý ani v mojej aplikácií nechýba. Užívateľovi je umožnené vybrať si z výberového boxu oponenta, ako je vidieť na **obrázku 4.3**, z dvoch predvolených obtiažností a z možnosti, kde si bude môcť obtiažnosť určiť sám. Táto obtiažnosť reprezentuje hĺbku zanorenia pre algoritmus *MiniMax* s *AlfaBeta* rezmi, ktorý využíva umelá inteligencia v triede *UI.java*.

Obtiažnosť začiatočník predstavuje hĺbku zanorenia dva a obtiažnosť pokročilý predstavuje hĺbku zanorenia tri. Pri zvolení týchto variant je možnosť zvolenia vlastnej obtiažnosti, čiže vlastnej hĺbky zanorenia neprístupná.



Obrázok 4.3

Po zvolení možnosti *PC – vlastná obtiažnosť* z výberového boxu oponenta je sprístupnená možnosť zadania vlastnej obtiažnosti ako je možné vidieť na **obrázku 4.4**. Povolený rozsah je nastavený na hodnoty od jedna do päť. Tento rozsah som zvolil z dôvodu časovej efektivity, keďže pri zanoreniach vyšších ako číslo päť sa časová náročnosť neprípustne zvyšovala. Pri zadaní hodnoty mimo rozsah je užívateľ upozornený na prekročenie rozsahu. Pokiaľ nieje zadaná korektná hodnota nie je povolené užívateľovi spustiť šachovú partiu.

Obrázok 4.4

4.3.3 Mód dvoch hráčov po sieti

Ako je už v úvode spomínané, aplikácia je architektúry *klient – server*. O hru po sieti sa v tomto prípade stará server. Zabezpečuje komunikáciu medzi klientskými stanicami, registráciu nových užívateľov, prihlasovanie užívateľov, v prípade online hrania preposiela výzvy na hranie šachových partií a následne zabezpečuje zasielanie ťahov vykonaných hráčmi na svojich klientoch. Komunikácia prebieha pomocou mnou navrhnutého protokolu.

Ide o textový protokol, typu otázka – odpoveď, kde jeden riadok oddelený koncom riadka tvorí jeden príkaz, ktorý je zaslaný z klienta na server. Server odpovedá špecifickou odpoveďou alebo potvrdením úspešného vykonania, či v prípade neúspechu chybovou správou. Z tohto dôvodu je možné si protokol rozdeliť na časť klientskú a časť serveru.

Protokol klientskej časti

Schéma protokolu je nasledujúca:

CHESS *OPERÁCIA* DÁTA

Na začiatku každého príkazu je vždy identifikačná hlavička správy vo forme názvu *CHESS*. Za ňou nasleduje medzera a názov operácie, ktorú chce aby server vykonal. V prípade, že príkaz využíva položku dáta, tak za názvom príkazu a následnou medzerou pripojí potrebné dáta a správu odošle na server, ktorý správu spracuje a odošle späť patričnú odpoveď.
Typy možných operácií:

- ***REGISTER*** – registrácia užívateľa. Využíva položku dáta pre uvedenie užívateľského mena a hesla v poradí meno medzera heslo
- ***LOGIN*** – prihlasovanie užívateľa. Využíva položku dáta pre uvedenie užívateľského mena a hesla v poradí "meno heslo"
- ***LOGOUT*** – ohlásenie užívateľa. Vyžíva položku dáta pre uvedenie užívateľského mena, ktorý sa chce odhlásiť.
- ***SAVE_CONFIG*** – uloženie užívateľských nastavení do databázy. Využíva položku dáta a to zadaním jedného z nasledujúcich identifikátorov:
  - ***A*** – nasleduje hostname
  - ***B*** – nasleduje číslo portu
  - ***C*** – nasleduje číslo skinu
- ***SET_STATUS*** – prepnutie statusu na serveri. Využíva položku dáta a to pre prepnutie do režimu užívateľa v hre zadaním *INGAME*, pripadne pre prepnutie do režimu online zadaním *ONLINE*
- ***CONTACTLIST*** – žiadosť o odoslanie kontakt listu
- ***ADDCONTACT*** – pridanie užívateľa do kontakt listu
- ***CHAT*** – odoslanie chatovej správy. Využíva položky dáta vo formáte "meno text", kde meno predstavuje meno užívateľa ktorému bola zaslaná správa v položke text.
- ***REQUEST*** – žiadosť o hranie hry. Využíva položku dáta vo formáte "šachovaVarianta užívateľ", kde šachový variant je reprezentovaný názvom šachového variantu, ku ktorému je vyzvaný užívateľ s menom v položke užívateľ.
- ***ACCEPT*** – potvrdenie prijatia výzvy na hru. Využíva položku dáta na správne preposlanie potvrdenia vyzývateľovi
- ***REFUSE*** – odmietnutie výzvy na hru. Využíva položku dáta na správne preposlanie zamietnutia vyzývateľovi
- ***MOVE_FIGURE*** – odoslanie jednotlivých ťahov vykonaných užívateľmi. Využíva položku dáta vo formáte "startX startY endX endY", kde startX a startY reprezentujú počiatočnú pozíciu figúrky a endX a endY cieľovú pozíciu figúrky
- ***EXIT_GAME*** – potvrdenie konca hry

Protokol serverovej časti

Schéma protokolu je obdobná protokolu klientskej časti:

CHESS *ODPOVEĎ* DÁTA

Na začiatku každej odpovedi je vždy identifikačná hlavička správy vo forme názvu *CHESS*. Za ňou po medzere nasleduje typ odpovedi. V prípade nutnosti využitia dátovej časti protokolu, pripojí dáta za medzeru za odpoveďou. Po skompletizovaní správy ju odosiela na klienta a server prechádza do stavu čakania na ďalšiu požiadavku na vykonanie operácie od klienta.
Typy možných odpovedí:

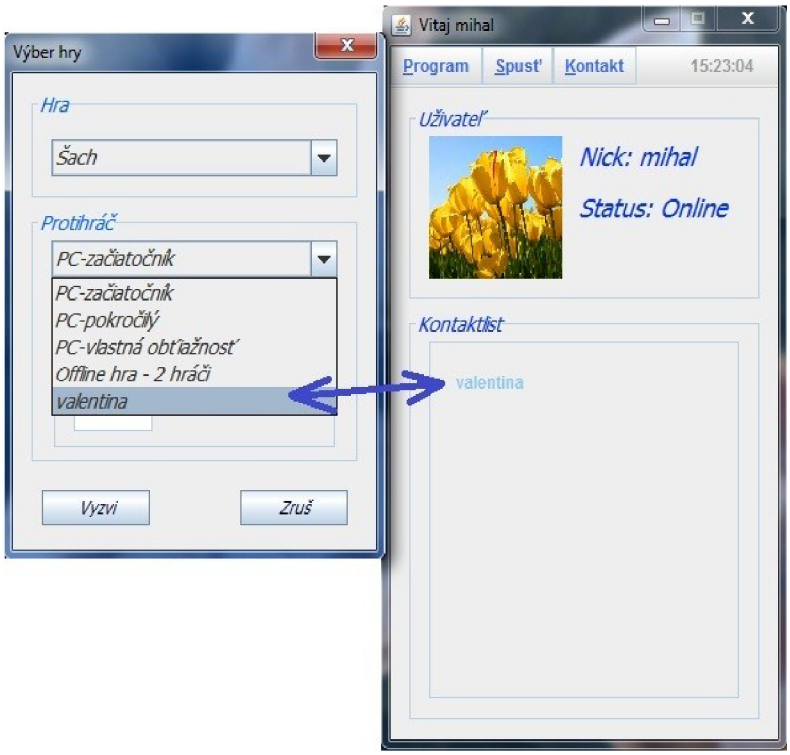
- ***EOK*** – potvrdenia úspešného vykonania operácie
- ***ACCEPT*** – potvrdenia výzvy na hru
- ***REFUSE*** – odmietnutie výzvy na hru

- ***ALLREADY_PLAY*** – automatické odmietnutie hry v prípade, ak vyzývaný hráč už hrá šachovú partiu
- ***MOVE_FIGURE*** – odoslanie jednotlivých ťahov vykonaných užívateľmi. Využíva položku dáta vo formáte "startX startY endX endY", kde startX a startY reprezentujú počiatočnú pozíciu figúrky a endX a endY cieľovú pozíciu figúrky
- ***EXIT*** – ukončenie hry
- ***CONTACTLIST*** – odoslanie kontakt listu. Využíva položku dáta na odoslanie menného zoznamu užívateľovho kontakt listu.
- ***ERR*** – chybové hlásenia. Využíva položku dáta, v ktorej sa môžu vyskytovať tieto varianty:
  - ***NOT_REGISTRED*** – neregistrovaný užívateľ
  - ***REGISTER*** – chyba pri registrácií
  - ***BAD_PASSWORD*** – zadané zlé heslo
  - ***LOGIN*** – chyba pri prihlasovaní
  - ***NOPLAYER*** – neprítomný žiadny hráč
  - ***NOOUTPUTSTREAM*** – došlo k chybe pri komunikácií s protihráčom
  - ***ACCEPT*** – chyba pri akceptácií, spôsobená odhlásením užívateľa
  - ***CHAT*** – chyba pri chatovaní
  - ***ADDCONTACT*** – chyba pri pridávaní kontaktu

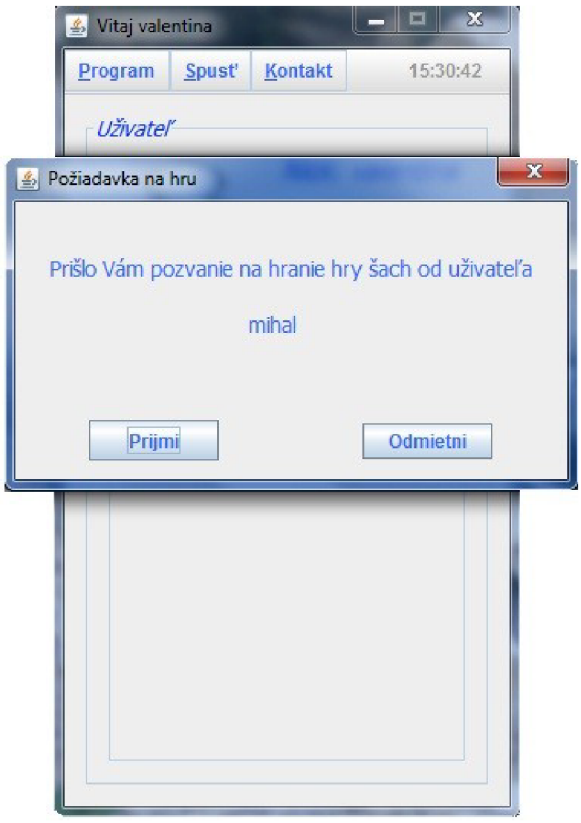
Online hranie

Pre hru online je nutné, aby aplikácia bola v režime online a bol online minimálne jeden užívateľ z vášho kontakt listu. Ako je možné vidieť na **obrázku 4.5,** online hráč sa zobrazí vo výberovom boxe oponentov. Vybraním užívateľa ako oponenta a následným vyzvaním, stlačením tlačidla *vyzvi*, je danému užívateľovi odoslaná výzva ako je možné vidieť na **obrázku 4.6**.

Po prijatí výzvy sa vyzvanému spustí daná hra, ktorá vyčká na pripojenie vyzývateľa, ktorému bolo odoslané potvrdenie o prijatí výzvy. Po obdržaní tejto správy sa vyzývateľovi zobrazí okno z prijatím výzvy a prechádza sa k samotnému hraniu šachovej partie. V prípade zamietnutia výzvy je vyzývateľ s odpoveďou oboznámený.



Obrázok 4.5



Obrázok 4.6

4.4 Doplnkové triedy

Pri realizácií implementácie aplikácie som si vytvoril triedy, ktorých implementačný zámer bol poskytnúť určitý spôsob uľahčenia. Ide o triedy zapúzdrujúce viac funkcií, prípadne poskytujúce spracovanie operácií vo vhodnom formáte, či zabezpečujúce automatické spracovanie.

V nasledujúcej časti sú v krátkosti popísané spomínané triedy.

ImagePanel.java

Jedná sa o mnou navrhnutú java swing Bean komponentu, poskytujúcu užívateľovi ľahkú manipuláciu s pridaním avatara do aplikácie.

Trieda dedí od *javax.swing.JPanel*, kde prepisuje rodičovskú metódu `paintComponent` na vykreslenie užívateľom zvoleného obrázku. Ako každý Bean komponent implementuje rozhranie java.io.Serializable. Reakcia na zmenu obrázku je prekreslením panelu na novo zvolený obrázok.

ChessFieldPanel.java

Je taktiež mnou navrhnutá java swing Bean komponenta, reprezentujúca políčko na šachovnici.

Prvou implementovanou vlastnosťou je schopnosť zmeny farby pozadia na základe zadaných súradníc, pričom políčka párových súradníc sú svetlejšieho odtieňa šedej farby a naopak nepárne sú odtieňa tmavšieho.

Druhou implementovanou vlastnosťou je schopnosť vykreslenia na popredí políčka obrázok, čo v mojom prípade umožňuje vykreslenie šachových figúrok.

Trieda dedí od *javax.swing.JPanel*. Vykreslenie šachovej figúrky zabezpečuje pridaný *javax.swing.JLabel* Okrem implicitného konštruktora triedy, ktorý nainicializuje súradnice na hodnoty rovné mínus jednej, poskytuje aj konštruktor s možnosťou zadania vlastných hodnôt súradníc.

```
public ChessFieldPanel(int x, int y)
```

Prvým parametrom je x-ová súradnica na šachovnici a druhý parameter je y-ová súradnica na šachovnici.

Pri vkladaní obrázka na políčko je nutné určiť aj typ vkladaného obrázka, v mojom prípade sa jedná o typ figúrky, ktorá na danom políčku stojí, čo značne uľahčuje prácu so šachovnicou.

Okrem *java.io.Serializable* implementuje trieda aj rozhranie *java.awt.event.MouseListener* z dôvodu implementácie *mouseEntered* a *mouseExited* efekt zvýraznenia šachového poľa pod kurzorom.

RandomGenerator.java

Trieda je zapúzdrením generátora poskytnutého štandardnou knižnicou *java.util.Random* generujúceho pseudonáhodné čísla rovnomerného rozdelenie a poskytuje metódy spracovávajúce vygenerované pseudonáhodné číslo do potrebného formátu.

Konštruktor je parametrizovaný, kde jediným parametrom je počiatočný seed dátového typu *long* pre generátor.

Metódy triedy sú využívané pri šachových variantoch z náhodným činiteľom a pri náhodnom rozostavení figúrok na šachovnici. Poskytované sú nasledujúce metódy.

- `getThrowDice()` – simulujúca hod kockou
- `getRandomIntInInterval(int from, int to)` – generujúca celé číslo z intervalu <from , to>
- `getRandomDoubleInInterval(int from, int to)` – generujúca číslo typu double z intervalu <from, to>

ImageLoader.java

Trieda zabezpečuje načítanie obrázkov akéhokoľvek druhu do aplikácie pomocou *java.awt.Toolkit*, ktorý ma na starosť získanie obrázka, pričom následne *java.awt.MediaTracker* zaobstaráva synchronizáciu načítania celého obsahu obrázka z behom aplikácie.

Načítanie obrázkov je možné buď jednotlivo, alebo zadaním zoznamu názvov obrázkov, ktoré majú byť stiahnuté a trieda automaticky vykoná všetky úkony. Taktiež je možné zadať veľkosť na ktorú chceme obrázok transformovať. Pri určení veľkosti sú transformačné hodnoty šírky a výšky inicializované na nulu, z dôsledku čoho nie je na obrázku vykonaná žiadna veľkostná transformácia.

Metóda obstaráva načítanie obrázka jednotlivo:

```
public void loadUserImage() throws InterruptedException
{
 image = toolkit.getImage(imageName);
 image = image.getScaledInstance(width, height, Image.SCALE_DEFAULT);
 tracker.addImage(image, 0, width, height);
 tracker.waitForID(0);
}
```

5 Grafické užívateľské rozhranie

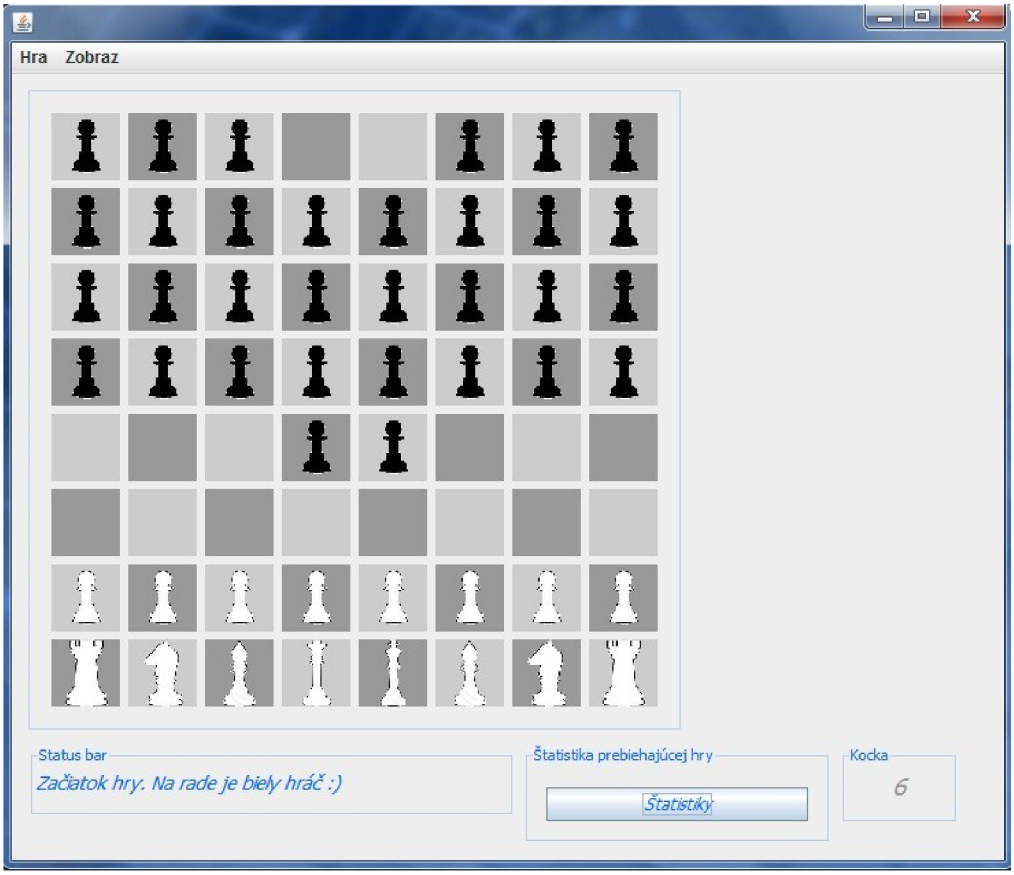
Kapitola pojednáva o stavbe a funkcionalite grafického užívateľského rozhrania. Na začiatku je popísané hlavné okno pri hre šachových partií, jeho základná funkcionalita a ovládacie prvky. Následné je pozornosť zameraná na popis štatistického okna, ktoré zabezpečuje informovanie o stave šachových sád na šachovnici a prípadnú manipuláciu. Záver sa venuje oknu používanému na základné rozmiestnenie figúrok na šachovnici.

Pri tvorbe užívateľského rozhrania som sa snažil vytvoriť rozhranie s jednoduchým a intuitívnym ovládaním, ktoré užívateľovi zabezpečí bezproblémové používanie všetkých poskytovaných funkcií. Na tvorbu grafického užívateľského rozhrania som použil štandardnú knižnicu *javax.swing* a využíval som grafický editor poskytovaný vývojovým prostredím Netbeans.

5.1 Hlavné šachové okno

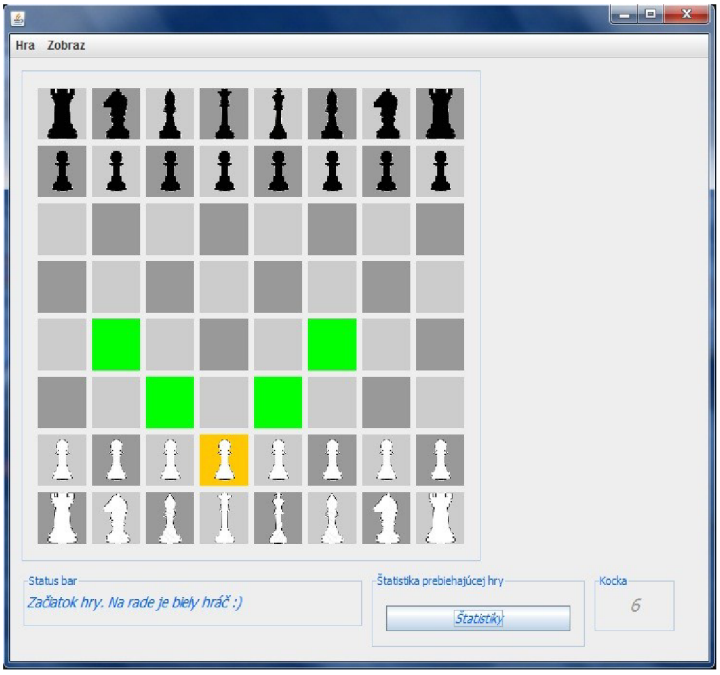
Reprezentuje okno zobrazujúce užívateľovi šachovnicu s hernými figúrkami, ponukové menu v hornej časti okna a informačné panely, čí tlačítka v dolnej časti. Ako je vidieť na **obrázku 5.1** skladá sa z nasledujúcich komponent:

- *chessPanel* reprezentuje šachovnicu zloženú z 64 *javax.swing.JPanel* komponent, ktoré obsahujú *javax.swing.JLabel* pre vykreslenie figúrky na panely. Jednotlivé šachové polia reagujú na kliknutie označením políčka a zaznamenaním si súradníc daného políčka. Pri kliknutí na políčko rovnakej pozície sa označenie ruší.
- *Menu* reprezentuje ponukové menu, ktoré je tvorené *javax.swing.JMenuBar* komponentou obsahujúcou položky *javax.swing.JMenu* s názvom *gameMenu* a *viewMenu* , ktoré sú tvorené *javax.swing.JMenuItem* položkami. Pri *gameMenu* ide o položky s názvom *newGameMenuItem* vykonávajúcu štart novej hry a *endGameMenuItem*, ktorá hru ukončí. Pre viewMenu ide o položky s názvom *helpMenuItem* zobrazujúcu nápovedu a *rulesMenuItem* zobrazujúcu pravidlá práve hranej hry
- *statusBarPanel* zabezpečuje zobrazenie stavu hry pomocou komponenty j*avax.swing.JLabel*
- *statisticPanel* obsahuje *javax.swing.JButton*, ktorý zobrazuje štatistické okno popísané v kapitole 5.2
- *dicePanel* zobrazuje aktuálnu hodnotu hodenú na kocke, pri šachových variantoch ovplyvnených hodmi kockou. Pri ostatných šachových variantoch je tento panel neaktívny. Hodnota je zobrazovaná pomocou komponenty *javax.swing.JLabel*



Obrázok 5.1

Ako je možné vidieť na **obrázku 5.2** po kliknutí na políčko s figúrkou sa farebne označia na šachovnici možné ťahy zvolenej figúrky.

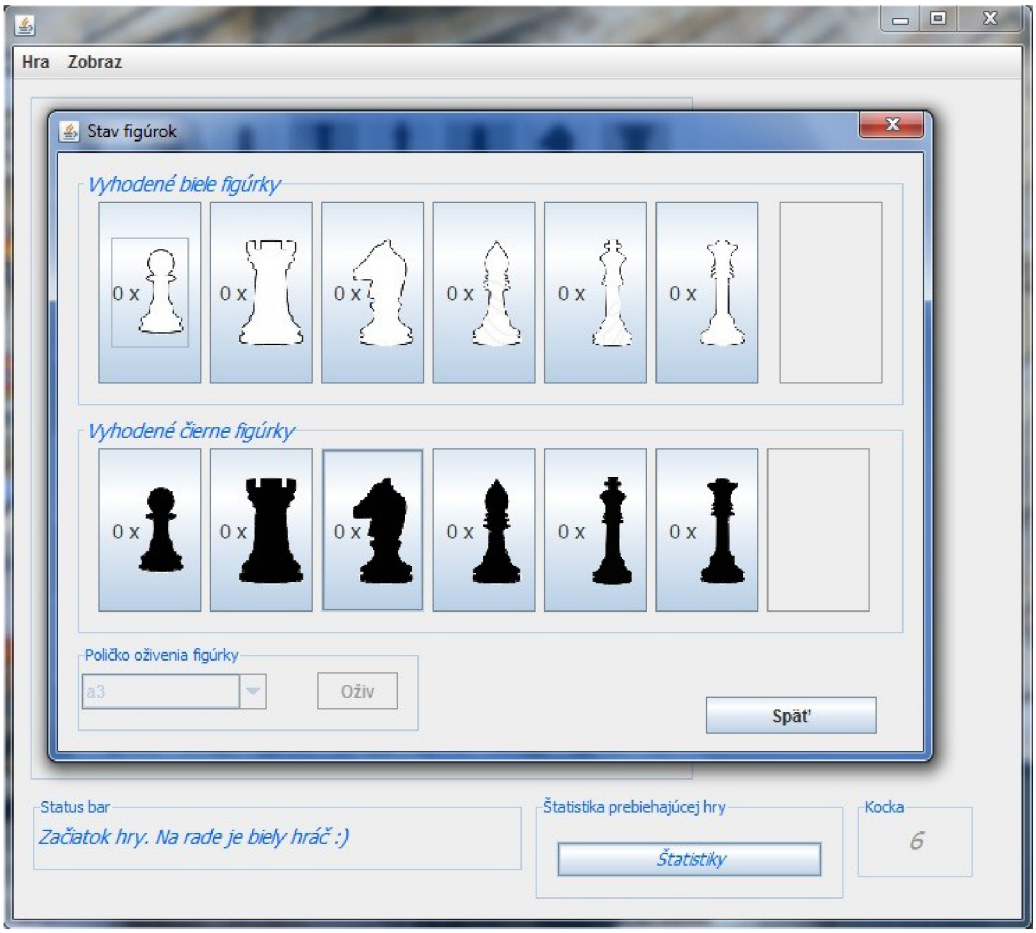


Obrázok 5.2

5.2 Štatistické okno

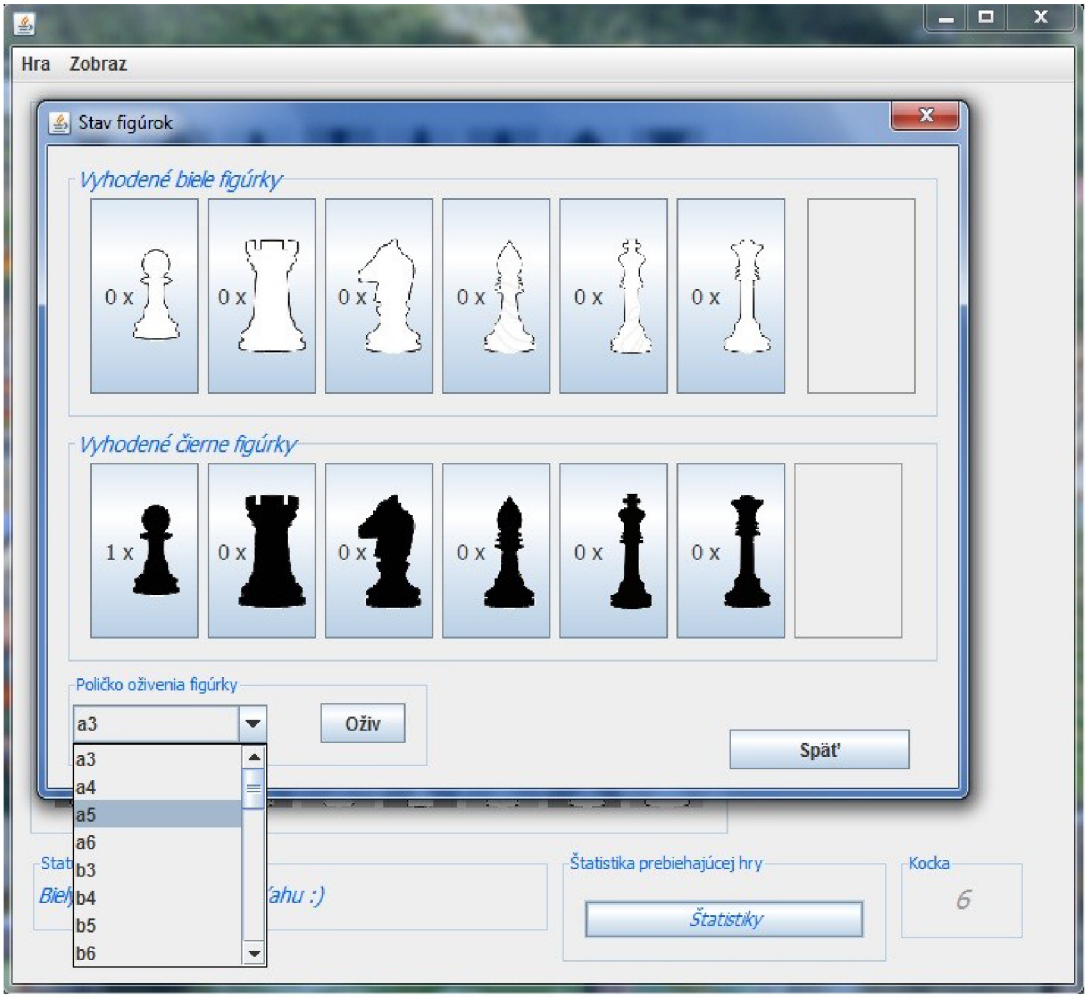
Získava a zobrazuje informácie o stave šachových sád jednotlivých hráčov a v prípade šachovej varianty umožňujúcej návrat vyhodených figúrok do hry, zabezpečuje výber a korektné umiestnenie figúrky späť na šachovnicu. Informácie v štatistickom okne sa aktualizujú vždy po získaní focusu. Vzhľad okna je možné vidieť na **obrázku 5.3** a obsahuje nasledujúce komponenty:

- *whiteFigurePanel* a *blackFigurePanel* obsahujú sedem javax.swing.JButton komponent, na ktorých je zobrazený počet vyhodených figúrok. V prípade šachovej varianty dovoľujúcej vrátenie vyhodenej figúrky do hry slúžia ako výberový mechanizmus pre typ figúrky určenej na vrátenie späť do hry
- *reanimatedFigurePanel* je panel obsahujúci komponenty javax.swing.JComboBox, ktorá poskytuje výber možných pozícií na vrátenia figúrky späť na šachovnicu, javax.swing.JButton, ktorý po stlačení vydá podnet na vykonanie vrátenia figúrky späť na šachovnicu a javax.swing.JLabel zobrazujúci prípadné chyby pri vykonaní vrátenia figúrky. Pri ostatných šachových variantoch je panel neaktívny. Aktívny je len v prípade hrania cyklických šachov
- *backButton* je tlačítko na uzatvorenie štatistického okna



Obrázok 5.3

V prípade výberu korektnej figúrky, ako je možné vidieť na **obrázku 5.4**, sa aktivuje *reanimatedFigurePanel*, ktorý nainicializuje pozičný výberový box korektnými pozíciami pre danú figúrku podľa pravidiel danej šachovej varianty. Po výbere pozície a potvrdením pomocou tlačidla nastáva vrátenie vybranej figúrky späť na šachovnicu.



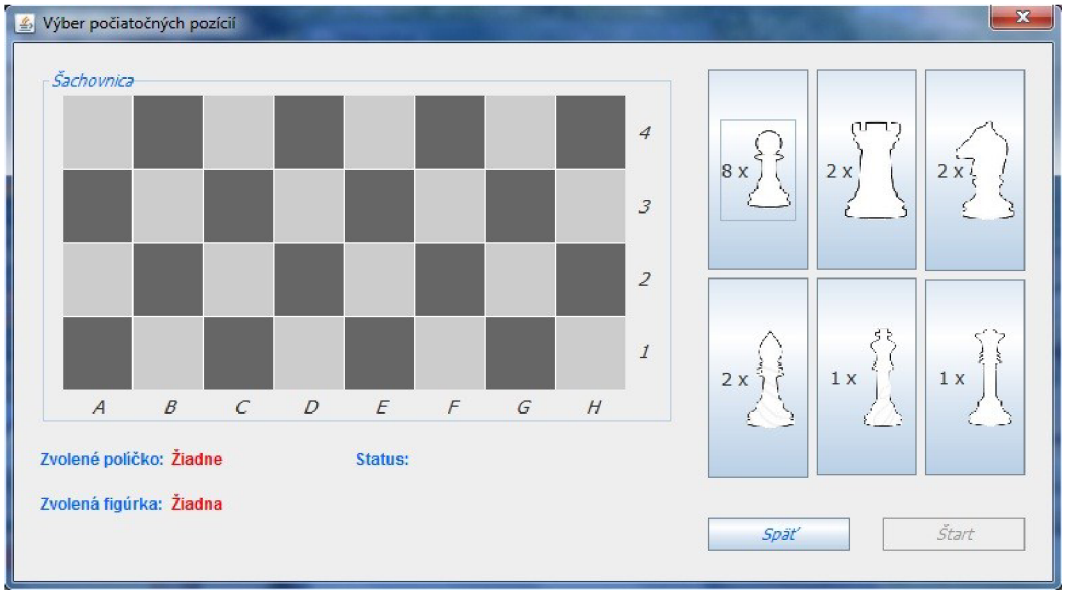
Obrázok 5.4

5.3 Okno pre základné rozostavenie figúrok

Využíva sa pri šachových variantoch pozičného šachu a voľne pozičného šachu. Vykonáva kontrolu dodržiavania pravidiel určených šachovou variantov pri rozostavovaní figúrok na šachovnici. V prípade porušenia pravidiel je užívateľ patrične upozornený. Vzhľad okna je možné vidieť na **obrázku 5.5** a je zložené z nasledujúcich komponent:

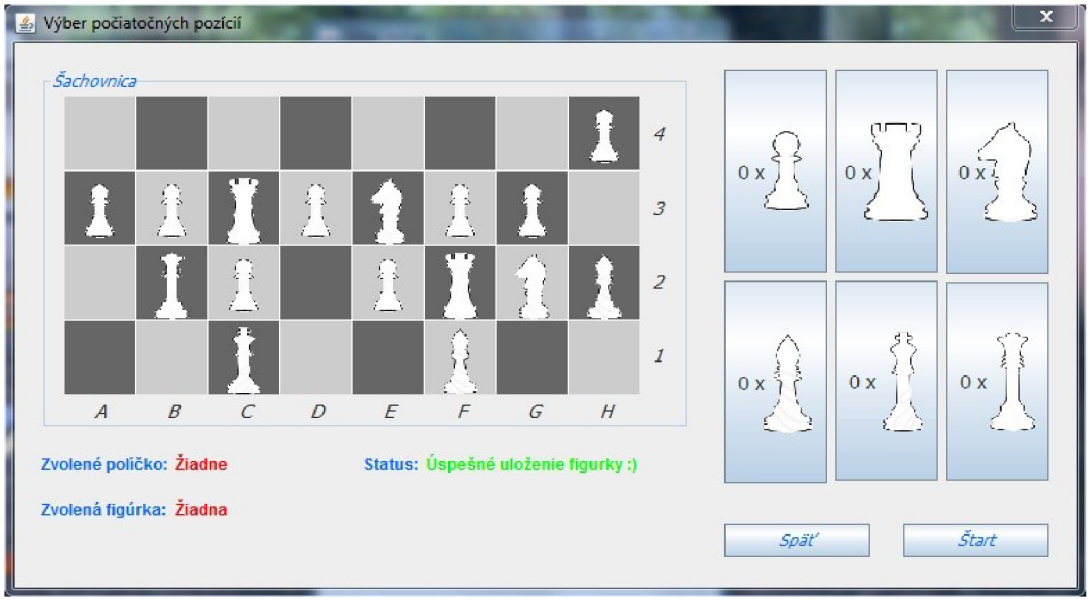
- *chessboardPanel* reprezentujúci polovicu šachovnice, na ktorú je umožnené užívateľovi rozostavovať figúrky. Pozostáva z tridsaťdva *chessField.ChessFieldPanel* komponentov usporiadaných pomocou *java.awt.GridLayout* a popisky súradníc šachovnice sú zabezpečené *javax.swing.JLabel* komponentami
- šesť *javax.swing.JButton* komponentov zobrazujúcich jednotlivé typy figúrok a zaobstarávajúce výber figúrky pre uloženie na šachovnicu. Pri každej figúrke je zobrazený aj počet možných uložení figúrky určitého typu

- informačné *javax.swing.JLabel* komponenty informujúce užívateľa o zvolených figúrkach či poliach šachovnice a v prípade pridania figúrky na šachovnicu o úspechu či neúspechu
- *javax.swing.JButton startButton* a *cancelButton* tlačítka vykonávajúce zrušenie procesu rozostavovania figúrok v prípade *cancelButton* a potvrdenie rozostavenia v prípade *startButton* tlačítka.



Obrázok 5.5

Po korektnom rozostavení figúrok po šachovnici je sprístupnené tlačítko „Štart“, názorne zobrazené na **obrázku 5.6**, a pri následnom stlačení dochádza k finálnej kontrole rozostavenia a v prípade úspechu k zápise na hlavnú šachovnicu. V prípade neúspechu je užívateľ patrične informovaný o dôvode neúspešného štartu.



Obrázok 5.6

6 Záver

Mojím cieľom bolo vytvorenie aplikácie, umožňujúcej hranie človeka proti počítaču a možnosti voľby vybraných variantov šachu.

V úvode práce som klasifikoval existujúce šachové varianty a následne porovnával prípadné rozdiely riešenia umelej inteligencie oproti šachom klasickým. V ďalších častiach som sa venoval realizácii implementácie aplikácie.

Vytvoril som aplikáciu umožňujúcu možnosť hrania, nie len človeka proti počítaču, ale aj dvoch ľudí v režime offline a v režime online pomocou architektúry client – server, komunikujúcej mnou navrhnutým protokolom. Grafické užívateľské rozhranie poskytuje jednoduché a intuitívne ovládanie, ktoré bolo testované užívateľmi s dobrými ohlasmi.

Prínosom práce je možnosť hrania rôznych šachových variantov v rôznych režimoch. Z pohľadu programátora bolo prínosom získanie nových skúseností pri tvorbe užívateľského rozhrania , umelej inteligencie a práce so sieťovou komunikáciou a databázou.

Jedným z možných vylepšení aplikácie v rámci umelej inteligencie by mohlo byť zlepšenie niektorých ohodnocovacích funkcií prípadným pridaním ďalších pravidiel špecifických pre jednotlivé šachové varianty, ktoré v aktuálnej podobe poskytujú obstojné vlastnosti. V prípade výberu šachových variantov je možné zlepšenie v implementácií ďalších existujúcich variantov.

Literatura

[1] PRITCHARD, D.B. The Classified Encyclopedia of Chess Variants. [Great Britain] : John Beasley, 2007. 384 s.

[2] Wikipedia [online]. 2001, 2010-05-18 [cit. 2010-05-18]. Chess variant. Dostupné z WWW: <http://en.wikipedia.org/wiki/Chess_variants>.

[3] Brainking [online]. c2010 [cit. 2010-05-18]. Dostupné z WWW: <http://brainking.com/>.

[4] Zboril, F.: Základy umelé inteligence, studijní opora. FIT VUTBR, 2006.

[5] Vít Valsa: Šachy, bakalárská práce, Brno, FIT VUT v Brne, 2009.

[6] Java™ Platform, Standard Edition 6 [online]. c2009 [cit. 2010-05-18]. Sun Microsystems. Dostupné z WWW: <http://java.sun.com/javase/6/docs/api/>.

[7] NetBeans [online]. c2010 [cit. 2010-05-18]. Dostupné z WWW: <http://netbeans.org/>.

Zoznam príloh

Príloha 1. DVD disk obsahujúci zdrojové dáta