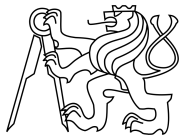
ČESKÉ VYSOKÉ UČENÍ TECHNICKÉ V PRAZE
FAKULTA INFORMAČNÍCH TECHNOLOGIÍ



ZADÁNÍ BAKALÁŘSKÉ PRÁCE

Název:	Aplikace pro řízení šachových turnajů
Student:	Jiří Pahorecký
Vedoucí:	Ing. Jiří Novák, Ph.D.
Studijní program:	Informatika
Studijní obor:	Webové a softwarové inženýrství
Katedra:	Katedra softwarového inženýrství
Platnost zadání:	Do konce letního semestru 2018/19

Pokyny pro vypracování

Cílem práce je vytvořit multiplatformní a open-source aplikaci, která bude sloužit rozhodčím a ředitelům šachových turnajů.

1) Nastudujte oficiální pravidla pro šachové turnaje ze zdrojů Mezinárodní šachové federace.
2) Proveďte podrobnou rešerši existujících aplikací pro řízení šachových turnajů, zhodnoťte jejich výhody a nevýhody.
3) Proveďte analýzu požadavků na systém a navrhněte vhodnou architekturu aplikace. Aplikace bude umět řídit různé typy šachových turnajů - systém každý s každým, švýcarský systém na libovolný počet kol, vyřazovací systém a turnaj družstev.
4) Systém implementujte jako desktopovou aplikaci v jazyce Java.
5) Aplikaci zdokumentujte a otestujte její funkčnost.
6) Zhodnoťte výhody implementovaného řešení vůči existujícím metodám.

Seznam odborné literatury

Dodá vedoucí práce.

Ing. Michal Valenta, Ph.D.
vedoucí katedry

doc. RNDr. Ing. Marcel Jiřina, Ph.D.
děkan

V Praze dne 20. listopadu 2017

FAKULTA
INFORMAČNÍCH
TECHNOLOGIÍ
ČVUT V PRAZE

Bakalářská práce

Aplikace pro řízení šachových turnajů

Jiří Pahorecký

Katedra softwarového inženýrství
Vedoucí práce: Ing. Jiří Novák, Ph.D.

25. června 2019

Poděkování

Chtěl bych poděkovat vedoucímu své práce Ing. Jiřímu Novákovi, Ph.D. za odborné vedení a cenné rady, které mi pomohly práci vypracovat.

Prohlášení

Prohlašuji, že jsem předloženou práci vypracoval(a) samostatně a že jsem uvedl(a) veškeré použité informační zdroje v souladu s Metodickým pokynem o etické přípravě vysokoškolských závěrečných prací.

Beru na vědomí, že se na moji práci vztahují práva a povinnosti vyplývající ze zákona č. 121/2000 Sb., autorského zákona, ve znění pozdějších předpisů. V souladu s ust. § 46 odst. 6 tohoto zákona tímto uděluji nevýhradní oprávnění (licenci) k užití této mojí práce, a to včetně všech počítačových programů, jež jsou její součástí či přílohou, a veškeré jejich dokumentace (dále souhrnně jen „Dílo"), a to všem osobám, které si přejí Dílo užít. Tyto osoby jsou oprávněny Dílo užít jakýmkoli způsobem, který nesnižuje hodnotu Díla, a za jakýmkoli účelem (včetně užití k výdělečným účelům). Toto oprávnění je časově, teritoriálně i množstevně neomezené. Každá osoba, která využije výše uvedenou licenci, se však zavazuje udělit ke každému dílu, které vznikne (byť jen zčásti) na základě Díla, úpravou Díla, spojením Díla s jiným dílem, zařazením Díla do díla souborného či zpracováním Díla (včetně překladu), licenci alespoň ve výše uvedeném rozsahu a zároveň zpřístupnit zdrojový kód takového díla alespoň srovnatelným způsobem a ve srovnatelném rozsahu, jako je zpřístupněn zdrojový kód Díla.

V Praze dne 25. června 2019 .....

Odkaz na tuto práci

Pahorecký, Jiří. *Aplikace pro řízení šachových turnajů*. Bakalářská práce. Praha: České vysoké učení technické v Praze, Fakulta informačních technologií, 2019.

Abstrakt

Práce se zabývá problematikou aplikací pro řízení šachových turnajů. Většina aplikací je placená. Ty, které jsou zdarma, bývají nepřehledné či se dokonce v nich vyskytuje reklama. Cílem práce je vytvoření multiplatformní a open-source aplikace, která bude sloužit ředitelům a rozhodčím amatérských turnajů. Nejprve byla popsána nejdůležitější pravidla pro šachové turnaje a provedena rešerše existujících aplikací. Poté byla provedena analýza požadavků a navrhntua architektura. Architektura byla zvolena tak, aby aplikace byla snadno spravovatelná a aby bylo jednoduché přidat další párovací enginy a pomocná hodnocení. Aplikace byla implementována tak, aby byla jednoduchá na ovládání, ale přesto umožnila plnohodnotné řízení amatérských šachových turnajů.

Klíčová slova multiplatformní aplikace, open-source, řízení turnajů, šachy, Java, JavaFX

Abstract

The thesis is about applications for managing chess tournaments. The ones that are free are often confusing or even contain advertisements. The aim of this thesis is to create a multi-platform and open-source application that will serve directors and referees of chess amateur tournaments. First the most important rules for chess tournaments were described, and the existing applications were compared. After that the requirements analysis was done and the architecture of the application was chosen. The architecture was chosen so that the application is easy manageable and to make it easy to add additional pairing engines and rankings. The application was implemented to be easy to use, yet still be able to manage amateur chess tournament.

Keywords multi-platform application, open-source, tournament management, chess, Java, JavaFX

Obsah

Seznam obrázků

Seznam tabulek

Úvod

Pořádání větších šachových turnajů se bez použití aplikací neobejde. Aplikace usnadňují celkové řízení turnajů. Umožňují například spravovat účastníky turnajů či automatické párování hráčů. Pro určité turnajové systémy by bylo potřeba mnoha úsilí pro párování kol. Pro oficiální turnaje Mezinárodní šachové federace je možno používat pouze aplikace jí oficiálně schválené. Schvalovací proces je relativně složitý. Většina z těchto oficiálně schválených aplikací však není bezplatná. Také některé jsou relativně složité či vizuálně působí starším stylem grafického uživatelského rozhraní. Co se týče bezplatných aplikací, tak bývají relativně nepřehledné či se dokonce v nich vyskytuje reklama. Z těchto důvodů jsem se rozhodl pro vytvoření vlastní aplikace. Bude bezplatnou alternativou pro řízení neprofesionálních šachových turnajů, intuitivní a jednoduchá na použití a s použitím moderních technologií pro grafické uživatelské rozhraní. Bude ovšem splňovat požadavky Mezinárodní šachové federace kladené na párování kol švýcarského systému. Proto bude v budoucnu možné po doimplementování aktuálně požadovaných funkcionalit zažádat o oficiální schválení.

Cílem práce je vytvořit multiplatformní a open-source aplikaci, která bude sloužit rozhodčím a ředitelům šachových turnajů. Prvním dílčím cílem je nastudovat a popsat oficiální pravidla pro šachové turnaje ze zdrojů Mezinárodní šachové federace. Navazujícím dílčím cílem je provedení podrobné rešerše existujících aplikací pro řízení šachových turnajů včetně zhodnocení jejich výhod a nevýhod. Dalším dílčím cílem je provedení analýzy požadavků a navržení vhodné architektury aplikace. Navazujícím dílčím cílem je pak naimplementování systému jako desktopové aplikace v jazyce Java a zhodnocení výhod implementovaného řešení vůči existujícím metodám. Posledním dílčím cílem je její zdokumentování a otestování její funkčnosti.

V práci se nejdříve v části 1 budu zabývat popisem nejdůležitějších pravidel pro šachové turnaje. Bude se jednat především o pravidla, ze kterých přímo či nepřímo vyplývají požadavky na aplikace pro řízení šachových tur-

najů. Poté se v části 2 zaměřím na porovnání existujících aplikací, stanovení a analýzu požadavků a analýzu problémové domény. Aplikací je mnoho, takže budou upřednostněny aplikace schválené Mezinárodní šachovou federací. Při stanovování požadavků bude brán zřetel na to, aby aplikace byla jednoduchá na použití ale přesto dokázala splnit nejdůležitější činnosti pro řízení turnajů. V části 3 bude proveden návrh architektury a výběr vhodných technologií. Bude brán především zřetel na to, aby aplikace byla multiplatformní a snadno spravovatelná. V části 4 budu popisovat implementaci aplikace včetně její instalace. Nakonec se v části 5 budu věnovat testování naimplementované aplikace.

Kapitola 1

Oficiální pravidla šachových turnajů

V této části budou stručně popsána oficiální pravidla pro šachové turnaje dané Mezinárodní šachovou federací. Budou popsána jen ta, která by mohla mít význam pro aplikace pro řízení šachových turnajů. Nebudou popisovány pravidla šachové hry. Zdrojem pro tyto informace bude oficiální webový portál Mezinárodní šachové federace https://www.fide.com.

Mezinárodní šachová federace je šachová organizace, která byla Mezinárodním olympijským výborem roku 1999 uznána jako mezinárodní sportovní federace [1]. Její používanou zkratkou je FIDE, která pochází z francouzského „Federation Internationale des Echecs“ [1]. V této práci budu tuto zkratku pro přehlednost používat. FIDE sdružuje 188 národních členských federací neboli svazů [1]. Mezi nimi je i Šachový svaz České republiky [2].

Nyní budou popsána oficiální FIDE pravidla pro turnaje. Na webovém portálu FIDE lze nalézt dokumenty *Fide Laws of Chess* popisující pravidla šachové hry a obecných soutěžních pravidel, *General Regulations for Competitions* popisující pravidla pro oficiální FIDE turnaje, *Standards of Chess Equipment, venue for FIDE Tournaments, rate of play and tie-break regulations* popisující technické záležitosti pro organizování turnajů a *Recommendations for Organization of Top-level Tournaments* popisující doporučení a pravidla pro organizaci top-level turnajů. Dle [3] se každý šachový turnaj musí řídit aktuálními pravidly *Fide Laws of Chess* a oficiální FIDE turnaje navíc ještě *General Regulations for Competitions*. Vzhledem k tomu, že ne všechny šachové turnaje jsou FIDE turnaje, tak se nejprve zaměřím na obecná soutěžní pravidla pro šachové turnaje. Nebudou popsána pravidla, která se týkají šachové hry samotné. Nebudou tedy popsána například pravidla šachu, používání šachových hodin, pravidla pro zápis tahů do partiářů apod. Poté popíši pravidla FIDE turnajů, a nakonec se ještě zmíním o doporučeních a pravidlech pro top-level turnaje. Nebudu však popisovat speciální turnaje, jako je například Šachová olympiáda.

Pro tento odstavec budu čerpat informace z [4]. Hráči by se měli chovat tak, aby nekazili dobré jméno šachu. Během probíhající hry nesmí používat žádné poznámky, získávat nápovědy a analyzovat jakoukoli partii na jiné šachovnici. Dokonce nesmí mít u sebe žádná elektronická zařízení. Rozhodčí však může udělit výjimky pro některá zařízení. Hráč má však vždy možnost si tato zařízení uschovat u rozhodčího. Pokud by se zjistilo, že má hráč některé elektronické zařízení u sebe, tak partii automaticky prohrává. Pravidla dané soutěže však mohou toto pravidlo změkčit. Hráči musí být ohleduplní vůči soupeři. Je zakázáno jakékoli rušení spoluhráče. Jedná se například o bezdůvodné nabídky remíz či vyvolávání bezdůvodných námitek. Jestliže hráč potřebuje vyhledat rozhodčího, má právo zastavit šachové hodiny. Rozhodčí následně posoudí, zda byl tento požadavek důvodný. Pokud nebyl, hráč bude penalizován. Pokud se hráč soustavě neřídí *Fide Laws of Chess*, automaticky prohrává. Jakmile hráči dokončí svojí partii, jsou automaticky považováni za diváky. Za výhru hráč obdrží jeden bod, za prohru žádný bod a za remízu polovinu bodu. Pravidla dané soutěže však mohou toto pozměnit. Ovšem nejsou povolena bodová hodnocení, která nedávají smysl. Jedná se například o 3/4 bodu pro vítěze a 1/4 bodu pro poraženého.

Opět budu informace z následujícího odstavce čerpat z [4]. Úkolem rozhodčího je zajistit dodržování pravidel šachu. Dalším úkolem je zajištění fair play a znemožnit podvádění. Musí také zajistit, aby herní prostředí bylo v dobrých podmínkách a aby hráči nebyli okolím rušeni. Má možnost při zvýšeném ruchu hráčům dát bonusový čas. Musí dohlížet na probíhající partie. A to obzvláště tehdy, když hráčům nezbývá příliš mnoho času na dokončení partie. Do probíhajících her nesmí zasahovat. Jsou však výjimky, kdy musí zasáhnout. Pokud nestíhá dozorovat probíhající partie, tak si může najít pomoc na dozor. V případě, kdy hráč probíhající partie si chce na něco stěžovat, má možnost se obrátit na rozhodčího. Pokud se hráč dopustí porušení pravidel, má rozhodčí možnost různých penalizací. Jedná se o varování, zvýšení času soupeři, snížení času provinělému, snížení bodů za danou partii provinělému, zvýšení bodů soupeři za danou partii, prohlášení partie za prohranou z pohledu provinělého, dát peněžní pokutu, vyloučit z několika kol či z celého turnaje. Zvýšení bodů protihráči však lze provést pouze do výše maximálního počtu bodů za danou partii. Peněžní pokuta musí být známa předem. Jestliže rozhodčí prohlásí provinělému partii za prohranou, musí také rozhodnout kolik bodů dostane jeho protihráč. Obecně všechny úkony rozhodčího musí být prováděny v nejvyšším zájmu dané soutěže.

Nyní popíši oficiální pravidla FIDE turnajů podle [3]. Opět budou popsána jen ta nejdůležitější a potenciálně se týkající řízení turnajů. Nebudu tedy popisovat například pravidla pro šachové vybavení. Jak již bylo zmíněno FIDE turnaje se musí řídit nejenom *Fide Laws of Chess* ale také *General Regulations for Competitions*. Hráči jsou organizátory pozváni prostřednictvím národních šachových federací. Pozvánka musí být co nejpodrobnější. Musí obsahovat například datum konání, ubytování, tempo hry, párovací systém, pomocná

hodnocení apod. Jakmile byl hráč pozván, jeho pozvánku již nelze odvolat. V případě odložení či zrušení turnaje, organizátoři mají povinnost poskytnout hráči odškodnění. Podobně to platí pro hráče. Jakmile potvrdí svoji účast, mají povinnost se turnaje zúčastnit. Výjimkou jsou však neočekávané události. Příkladem může být nemoc. Také je proti pravidlům opuštění rozehrané partie bez oznámení rozhodčímu. Tomuto hráči pak hrozí postih dle uvážení rozhodčího pro špatné sportovní chování. Hráči mají povinnost být slušně oblečeni. Televizní kamery jsou během hry povoleny, pokud jsou bezhlučné a nejsou nijak obtěžující a zároveň pokud jsou povoleny organizátory a hlavním rozhodčím. Fotografování je povoleno pouze autorizovaným osobám. Fotografování s bleskem je během hry povoleno jen během prvních 10 minut 1. kola a během prvních 5 minut ostatních kol. Fotografování bez blesku během dalších minut je povoleno pouze se souhlasem hlavního rozhodčího. Zvláštní pravidla platí pro kapitány v turnajích družstev. Pravidla daného turnaje určují, do kdy musí kapitáni předat soupis hráčů pro daná kola. Hráči nemohou bez povolení rozhodčího opustit hrací místnost. To však neplatí pro kapitány. Jakmile kapitán chce mluvit s některým ze svých hráčů, musí to nejdříve oznámit rozhodčímu. Komunikace pak probíhá za účasti rozhodčího, a to v jazyce, kterému rozhodčí rozumí. Kapitáni mají možnost radit svým hráčů, zda mají přijmout nabídku remízy či jí mohou sami přijmout. Pravidla daného turnaje však toto mohou zakázat. Nicméně kapitáni nesmí nijak zasahovat do probíhajících partií. Kapitán má také možnost předat svojí funkci jiné osobě. Musí to však písemně sdělit hlavnímu rozhodčímu s dostatečným předstihem.

Nyní popíši speciální pravidla pro top-turnaje podle [5]. Vzhledem k tomu, že pro ně platí většina již zmíněných pravidel, budou popsány pouze pravidla, které jsou pro tyto turnaje speciální. Hráči jsou pozváni s dostatečným předstihem, a to nejlépe půl roku před konáním daného turnaje. Diváci musí být usazeni a musí být v dostatečné vzdálenosti od hracího prostoru. Do toho prostoru nesmějí vstupovat. Mělo by být zajištěno, aby hráči nebyli rušeni diváky či venkovním prostorem. Musí být ustanovena speciální místnost pro pozápasovou analýzu partií a pro pozápasovou tiskovou konferenci. Během probíhajících partií musí být hráčům k dispozici občerstvení. Fotografování s bleskem je povoleno pouze pro první 3 minuty každého kola. Kamerové vysílání je povoleno jen pokud není pro hráče obtěžující. Toto vysílání je dokonce doporučeno. Organizátoři mají povinnost sehnat co nejvíce mediální a tiskové publicity. Hlavní rozhodčí turnaje by měl být zkušený a respektovaný. Hráči musí být velmi dobře oblečeni a upraveni. V opačném případě se to bere jako neúcta ke hře a také k sponzorům. Hrací doba musí být po 13:00. Každé kolo musí začínat ve stejnou hodinu. Je to z důvodu, aby byl zachován denní rytmus. V případě nečekané nutné změny hrací doby, musí to rozhodčí osobně prokonzultovat s hráči. Před posledním kolem by měl být volný den. Je to z toho důvodu, aby se všechny odložené partie mohly odehrát právě v tento den.

Nyní popíši pomocná hodnocení pomocí informací z [6]. Po odehrání jakéhokoli kola může nastat situace, kdy někteří hráči dosáhli stejného počtu bodů. Pomocná hodnocení slouží pro určení průběžného či konečného pořadí hráčů. Jejich výběr musí být proveden včas a musejí být oznámeni před zahájením turnaje. Pokud není po aplikaci daných pomocných hodnocení jednoznačně určené pořadí, pak rozhodne los. Nejlepší systémem je Play-Off, který však nemusí být vždy vhodný. Například není pro něj dostatek času. Systém Play-Off se skládá z extra partií a může mít mnoho podob. Pokud je pro turnaj použit, musí být před začátkem turnaje definován.

Nyní popíši pomocná hodnocení, která jsou nejpoužívanější. Zdrojem informací pro jejich popis bude opět [6]. Budou popsány v abecedním řazení. Slovem protihráč či soupeř budu mít na mysli hráče, se kterým daný hráč odehrál partii.

Buchholz systém
Buchholz systém je definován jako součet bodů všech soupeřů. Existují jeho různé modifikace. Střední Buchholz je Buchholz bez bodů protihráčů s nejnižším a nejvyšším počtem bodů. Střední Buchholz 2 je Střední Buchholz odečtený ještě o body protihráčů s druhým nejnižším a druhým nejvyšším počtem bodů. Další variantou je Buchholz Cut, což je Buchholz bez bodů soupeře s nejmenším počtem bodů. Buchholz Cut 2 je Buchholz Cut bez soupeře s druhým nejmenším počtem bodů.

Koya systém pro systém každý s každým
Je definován jako počet získaných bodů proti soupeřům, kteří dosáhli 50 % bodů nebo více.

Počet her hraných černými figurami Toto hodnocení je zřejmé ze svého názvu. Je určeno počtem her, ve kterých daný hráč hrál černými figurami. Neodehrané hry se berou jako, kdyby hráč hrál bílými.

Průměrný rating protihráčů
Průměrný rating protihráčů je definován jako součet všech ratingů soupeřů vydělený počtem odehraných her. Existuje také Cut verze, ve kterém se při jeho výpočtu neuvažuje jeden nebo více nejmenších ratingů soupeřů.

Sonneborn-Berger systém
Zde se rozlišují verze pro jednotlivce a pro družstva. Pro jednotlivce je definován jako součet součtu bodů soupeřů, které porazil a poloviny součtu bodů se kterými remizoval. Pro turnaje družstev je definován jako součet součinů každého soupeřova družstva a počtem bodů proti němu získanému.

Speciální pomocná hodnocení pro turnaje družstev
Turnaje družstev mohou mít jako hlavní hodnocení součet zápasových bodů družstva. Tedy například za výhru družstva jeden bod, za remízu polovinu bodu a za prohru žádný. Nebo mohou mít jako hlavní hodnocení součet všech bodů z odehraných her všech jeho členů. Jsou tedy dvě možnosti, jak zvolit hlavní hodnocení. Jako pomocné hodnocení lze pak zvolit to druhé, které nebylo zvoleno jako hlavní.

Vzájemné zápasy Pokud se všichni hráči se stejným počtem bodů střetli, tak je pořadí rozhodnuto dle celkového počtu bodů z těchto střetnutí.

Nyní budou popsány tři nejhranější turnajové systémy. První dva jsou velmi intuitivní a jsou všeobecně známy i z jiných her a sportů. Švýcarský systém je složitější a má dokonce různé varianty. Bude proto popsán podrobněji. Všechny tři systémy lze použít pro turnaje jednotlivců tak i pro turnaje družstev. Ke každému systému budou uvedena doporučená pomocná hodnocení, a to v pořadí v jakém jsou zapsány.

Systém každý s každým
Tento systém je výstižný svým názvem. Každý hráč se utká v některém kole s každým hráčem. Můžeme rozlišit jednokolovou a dvoukolovou verzi tohoto systému. U jednokolového se dva hráči střetnou pouze jednou, u dvoukolového dvakrát. U dvoukolového je tedy výhoda, že dva hráči budou proti sobě hrát jak bílými, tak i černými figurami. Systém každý s každým by měl být hrán pomocí tzv. Bergerových tabulek [3]. Tyto tabulky udávají párování jednotlivých kol. Pro dvoukolovou verzi je doporučeno obrátit pořadí posledních dvou kol první etapy [7]. Je to z důvodu, aby hráči nehráli třikrát se stejnou barvou figur [7].

Nyní uvedu příklad Bergerovy tabulky pro 3 či 4 hráče v jednokolové verzi pomocí [7]. Předpokládejme, že hráči dostali před zahájením turnaje číslo nasazení. Každý dostane jiné číslo a je tedy tímto číslem definován. Můžeme tedy napsat, že pro první kolo platí 1-4, 2-3. Znamená to, že hráč s číslem nasazení 1 hraje s bílými figurami s hráčem s číslem 4. Hráč s číslem 2 hraje s bílými figurami s číslem 3. Pro druhé kolo platí 4-3, 1-2. Pro třetí a zároveň poslední kolo platí 2-4 a 3-1. Na [7] můžeme vidět Bergerovy tabulky až pro 16 hráčů.

Pro turnaje jednotlivců je doporučeno zvolit jako pomocná hodnocení vzájemné zápasy, počet výher, Sonneborn-Berger a Koya systém [6]. Pro turnaje družstev jsou to počet zápasových bodů či počet všech bodů všech hráčů družstva (viz Speciální pomocná hodnocení pro turnaje družstev), vzájemné zápasy družstev a Sonneborn-Berger systém [6].

Vyřazovací systém
Tento systém je opět výstižný svým názvem. Hráči jsou předem definovaným způsobem párováni a pouze vítěz postupuje do dalšího kola. Poražený hráč je vyřazen z turnaje. Příkladem využití tohoto systému byl turnaj Fide World Chess Cup odehraný v roce 2017 [8].

Švýcarský systém
Jak již bylo řečeno, je tento systém relativně komplikovaný a má několik verzí. Oficiální verze se nazývá Holandský systém. Kromě tohoto oficiálního systému existují i další. Tyto systémy se však považují za zastaralé [9]. Jedná se například o Dubov systém, Burnstein systém či Lim systém. Švýcarský systém je výhodný pro turnaje, které nemohou být z důvodu velkého počtu hráčů hrány systémem každý s každým. Je tedy předem stanoven počet kol a párování je prováděno dle dané verze tohoto systému. Dále budou popsána obecná pravidla pro všechny verze švýcarského systému a jeho akcelerační variace.

Nyní popíši základní pravidla podle [10]. Počet kol je stanoven s dostatečným předstihem. Dva hráči nikdy nesmí spolu odehrát více než jednu partii. Pokud by byl celkový počet hráčů lichý, obdrží nespárovaný hráč volné kolo neboli bye. Za volná kola obdrží hráči počet bodů, jako kdyby hráli a vyhráli. Pravidla daného turnaje to mohou pozměnit. Volné kolo může hráč obdržet pouze jednou za celý turnaj. Obecně jsou párováni hráči se stejným počtem bodů. Žádný hráč nemůže dostat po třetí za sebou stejnou barvu figur a rozdíl mezi počtem kol s danou barvou a počtem kol s druhou nesmí být roven 2 či -2. Výjimkou je však poslední kolo turnaje. Obecně hráči dostávají pro dané kolo barvu figur se kterými hráli nejméně partií. Pokud jsou počty her oběma barvami stejné, dostanou barvu opačnou od předchozího kola.

Nyní popíši podrobnější pravidla podle [11]. Před zahájením turnaje je dle předem stanovených kritérií každému hráči přiřazeno párovací číslo. Většinou se pro to používá Elo koeficient neboli rating Elo. Tento koeficient odráží sílu hráče. Hráč s největším Elo koeficientem obdrží párovací číslo 1, hráč s druhým největším číslo 2 atd. V případě rovnosti Elo koeficientů, jsou hráči porovnáni oficiálními šachovými tituly a poté abecedně. Jestliže některý hráč nemá Elo koeficient, musí mu rozhodčí přiřadit přibližné Elo, a to nejpřesněji jak je možné. V případě nalezení chyb, lze tyto párovací čísla změnit do doby, než bude čtvrté kolo spárováno. Poté již nejsou změny možné. Pokud se hráči zúčastní až druhého či třetího kola, dostanou párovací číslo až dorazí na turnaj. Za každé neodehrané kolo dostanou počet bodů jako v případě prohry. Toto pravidlo však může být pozměněno pravidly daného turnaje. Pokud turnaj obsahuje hráče, kteří první či druhé kolo nehráli, tak se původní párovací čísla berou jako dočasná. Definitivní párovací čísla jsou až po příjezdu všech účastníků. Pokud je nějaká partie daného kola odložena, je pro párovací účely dalších kol brána jako remíza. Pokud hráč odstoupí z turnaje či byl vyřazen

rozhodčím, není již ve zbývajících kolech párován. Také se může stát, že hráči se nezúčastní z určitého důvodu nějakého kola. V tomto kole dostanou stejný počet bodů, jako kdyby prohráli. Opět toto pravidlo může být pozměněno pravidly daného turnaje. Pro první kolo se seznam hráčů rozdělí do dvou skupin dle párovacích čísel. Pro párování prvního kola platí, že hrají spolu první hráči obou skupin, druhý s druhým atd. To však způsobuje, že většinou v prvním kole zvítězí silnější hráč neboli hráč s větším párovacím číslem. Pouze několik málo procent her skončí jinak. Pro eliminaci tohoto jevu slouží akcelerační systémy.

Nyní popíši akcelerační verzi podle [12]. Akcelerační systém je variace švýcarského systému, která se snaží eliminovat nevítaný jev prvních kol. Musí pro něj platit, že výsledné pořadí hráčů bude na konci statisticky stejné jako u normální verze švýcarského systému. U tohoto systému platí, že se pro párovací účely nepoužívá pouze body hráčů. Může se jednat například o virtuální body. Pro párování se pak používá u každého hráče součet normálních a virtuálních bodů. Dosud jediným schváleným akceleračním systémem je Baku akcelerace. Při ní se hráči před prvním kolem rozdělí do dvou skupin. První bude obsahovat první polovinu hráčů dle párovacích čísel zaokrouhlenou nahoru na nejbližší prvočíslo. Před párováním prvních třech kol je každému hráči v první skupině přidělen jeden virtuální bod. Tyto virtuální body jsou před párováním čtvrtého a pátého kola sníženy na 0,5. Po odehraném pátém kole již žádný hráč neobdrží žádný virtuální bod.

Pomocí [6] nyní popíši doporučená pomocná hodnocení. Pro turnaje jednotlivců je doporučeno zvolit jako pomocná hodnocení vzájemné zápasy, počet výher, počet her hraných černými figurami, průměrný rating protihráčů verze Cut, Buchholz verze Cut 1, Buchholz a Sonneborn-Berger systém. Pro turnaje družstev jsou to počet zápasových bodů či počet všech bodů všech hráčů družstva (viz Speciální pomocná hodnocení pro turnaje družstev), vzájemné zápasy družstev, Buchholz verze Cut 1, Buchholz a Sonneborn-Berger systém.

Pro řízení velkých šachových turnajů hraných švýcarským systém je potřeba aplikace. FIDE doporučuje používání aplikací, které sama oficiálně schválila. Tyto informace jsou uvedeny v [13]. Autor aplikace pro řízení šachových turnajů může zažádat o schválení zasláním formuláře FE-1. Schválení se dostává pro jakoukoli FIDE verzi švýcarského systému, tedy například Dubov verze.

Dále podle [14] popíši jaké požadavky musí aplikace splňovat. Všechna párování musí být v souladu s pravidly daného herního systému. Kromě Dubov systému musí být párování prováděno pomocí párovacích čísel. Po spárovaném pátém kole se již tato čísla nesmí měnit. Aplikace musí také implementovat oficiální akcelerační systémy. V současné chvíli tedy pouze Baku akceleraci. Aplikace musí být schopna správně importovat a exportovat TRF souborový formát. Export tohoto formátu je doporučen s použitím UTF-8 kódováním. Aplikace musí nabízet možnost dávat neobvyklé výsledky. Například se jedná o výsledky 0,5–0, 0–0,5, 0-0. Na druhou stranu nesmí

nabízet nesmyslné výsledky typu 1–0,5 či 1–1. Pro případy kontumace partie jsou povoleny pouze výsledky 1F–0F, 0F-1F, 0F-0F. Písmeno F je zde převzato z anglického slova „forfeit“. Musí být také schopna správně zpracovávat odložené či přerušené partie. Musí nabízet možnost volby kolik bodů dostane hráč, který nebyl spárován pro dané kolo. Aplikace by měla umožnit kdykoli zobrazit oficiální FIDE ratingový list.

Nyní podle [13] popíši, co ještě musí daná aplikace splňovat. Musí nabízet anglickou verzi uživatelského rozhraní. Dále musí mít uvnitř zabudován kontrolor párování, který je schopen přijmout soubor TRF. Poté pro každé kolo daného turnaje v tomto souboru vytvoří párování pomocí zabudovaného enginu. Nakonec porovná, zda se toto párování shoduje s párováním v souboru. Také je nutnost mít naimplementovanou malou aplikaci, která je schopna generovat a simulovat turnaje a vytvořit výstupy v TRF formátu. Je zde nutnost, aby tato malá aplikace přesně dodržela pravidla, která engine schvalované aplikace implementuje. Všechny tyto požadavky je nutné splnit. Přesto nejsou vyčerpávající a jejich splnění negarantuje úspěšnost oficiálního schválení. Co se týče párovacího enginu, tak aplikace může mít naimplementován svůj vlastní či využít externí. Příkladem externího enginu, který je oficiálně schválenými aplikacemi nejvíce používán, je JaVaFo.

Kapitola 2

Analýza

V této kapitole provedu nejdříve analýzu existujících aplikací pro řízení šachových turnajů. Poté stanovím a popíši požadavky na svojí aplikaci a následně popíši problémovou doménu. Hlavním vstupem informací budou informace uvedené v kapitole 1. Abych dosáhl co největší přehlednosti, budu používat UML diagramy.

2.1 Porovnání existujících aplikací

Aplikací pro řízení šachových turnajů je mnoho. Také lze tyto aplikace třídit podle mnoha kritérií. Například podle licence či zda jsou oficiálně schválené FIDE. Pro roztřídění použiji právě informaci, zda jsou schválené či nikoli. Schválené aplikace musely projít relativně složitým schvalovacím procesem, takže je u nich zaručena funkčnost [13]. Vzhledem k tomu, že aplikací je mnoho, budu největší důraz věnovat právě schváleným aplikacím. Přehled schválených aplikací je na [15]. Z dosud neschválených aplikací vyberu pouze jednoho zástupce. Informace budu čerpat z oficiálních webových stránek produktů a ze samotné aplikace, či její demoverze. Většina demoverzí obsahuje vše, co plná verze, jen je nějak limitována. Může tomu tak být omezením počtu kol, účastníků či zakázáním používáním pro reálné turnaje. Pro párování švýcarského systému aplikace využívají buď vlastní interní párovací engine nebo externí od jiného dodavatele. Co se týče externích enginů, tak se nejčastěji jedná o JaVaFo engine. Tento engine je zdarma [16]. Při hodnocení přehlednosti a zpracování grafického uživatelského rozhraní (GUI) budu používat svůj názor, jelikož je tato stránka subjektivní. Vzhledem k tomu, že budu implementovat desktopovou aplikaci, budou porovnávány pouze desktopové aplikace.

2.1.1 Aplikace oficiálně schválené FIDE

Swiss-Manager
Zdrojem informací pro tuto aplikaci bude [17]. Aplikace není bezplatná. Nabízí však demo verzi, která je omezena na 4 kola a na maximálně 60 hráčů. Jako turnajový systém umožňuje zvolit švýcarský systém, systém každý s každým a vyřazovací systém. Tyto systémy nabízí ve verzích pro jednotlivce i pro družstva. Pro švýcarský systém lze zvolit i Baku akcelerační verzi.

Umožňuje zvládnout švýcarský systém do 1500 účastníků a do 23 kol turnaje. V případě verze pro družstva zvládne až 300 družstev. Co se týče systému každý s každým, tak zvládne také až 1500 účastníku, ale co se týče verze pro družstva, tak pouze 50 družstev. Přibližně 200 000 turnajů bylo úspěšně řízeno touto aplikací a přibližně 100 000 z nich jsou dohledatelné na webové stránce [18].

Při vytváření turnaje nabízí aplikace umožňuje nastavit řadu atributů. Kromě obecných, jako je například název turnaje, organizátor, počet kol, nabízí také zvolení pomocných hodnocení, podle jakého ratingu řadit hráče a spoustu dalších. Umožňuje také zvolit specifický typ turnaje. Příkladem může být Mistrovství Evropy či Šachová olympiáda. Nabízí mnoho pomocných hodnocení pro všechny turnajové systémy, které podporuje. Umožňuje zvolit jakékoli pomocné hodnocení popsané v části 1. Pro každý systém jich nabízí přibližně 20. Pro daný turnaj jich však lze zvolit maximálně pět. Lze také umožnit volbu nestandardních výsledků partií. Zadávání účastníků lze ručně nebo nahráním xml souboru s účastníky. Při nasazování kol, tedy při párování účastníků turnaje, lze některé hráče vyjmout z nasazení pro dané kolo či je vyloučit z turnaje úplně. Obojí je vratné. Lze také přidat hráče během turnaje. Během turnaje se lze k předchozím kolům vracet a v případě potřeby poupravit výsledky. Také lze kdykoli zobrazit či vytisknout seznam párovaní hráčů pro dané kolo a průběžná hodnocení. Aplikace umožňuje import a export různých souborů. S těmito soubory pak lze v aplikaci pracovat. Co se týče textových souborů, tak umožňuje import a export údajů o hráčích, termíny, údaje o družstvech, nasazení družstev, nasazení hráčů. Lze také importovat a exportovat PGN soubory se šachovými partiemi. Aplikace umožňuje zvolení několika párovacích enginů. Ty lze zvolit při nasazování daného kola. Pro FIDE turnaje je možné zvolit pouze JaVaFo engine. Aplikace umožňuje doinstalovat příručku v různých jazycích. Po jejím nainstalování, je možné jí používat přímo v aplikaci. Aplikace ukládá do logovacího souboru prováděné akce. V aplikaci je tento logovací soubor možno zobrazit. V aplikaci je také možnost pro aktualizaci programu. Tato funkcionalita nabídne stažení nejnovější verze aplikace. Nabízí také odkazy na domovskou stránku aplikace a webový portál databáze šachových turnajů *Chess-Results.com*. Na tento portál je možné nahrát turnaj přímo z aplikace. Co se týče GUI, tak vzhledem na mě působí starším stylem. Nicméně dle mého názoru, je po seznámení s aplikací, její ovládání relativně intuitivní. Navíc v porovnání s ostatními

šachovými turnajovými aplikacemi, kterými se v této rešerši zabývám, má nejvíc možností pro nastavování různých atributů turnaje. Příkladem může být již zmíněná spousta pomocných hodnocení. Z důvodu velkého množství, nebyly popsány všechny možnosti, ale jen ty z mého pohledu nejdůležitější pro řízení turnajů.

V porovnání s ostatním aplikacemi nabízí nejvíce funkcionalit. Z tohoto důvodu budu občas při popisu ostatních aplikací srovnávat právě s touto aplikací.

Vega

Zdrojem informací pro tuto aplikaci bude [19]. U této aplikace se licence liší podle verze pro operační systém. Pro GNU/Linux je zdarma, zatímco pro Windows je zdarma pouze pro turnaje do 30 hráčů. Autor navrhl a vytvořil aplikaci tak, aby byla intuitivní a jednoduchá k použití. Používá C++ multiplatformní framework Ultimate++. Právě proto je jí možné spustit nejen na Windows ale i na GNU/Linux. Aplikace zvládne turnaje do 1200 hráčů a maximálně 23 kol pro švýcarský systém. Pro systém každý s každým zvládne až 24 hráčů.

Aplikace obsahuje dvě verze. Jedna je pro turnaje jednotlivců a druhá pro turnaje družstev. První je nazvána Vega a druhá VegaTeam. Obě verze jsou automaticky nainstalovány. Nejdříve popíši verzi pro turnaje jednotlivců. U verze pro družstva budu popisovat pouze odlišnosti od verze pro jednotlivce.

U verze pro jednotlivce lze při zakládání turnajů zadat jejich atributy. Je jich sice mnohem méně než u aplikace Swiss-Manager, ale obsahují ty nejdůležitější. Jedná se o název turnaje, místo konání, národní federaci, datum, hlavního rozhodčího atd. Aplikace umožňuje zvolit jednokolový a dvoukolový systém každý s každým a všechny druhy švýcarského systému. Oficiální verze švýcarského systému používá párovací engine JaVaFo. Ostatní verze používají interní párovací engine. Také je možné zvolit akcelerační verzi švýcarského systému. Nabízí volbu 18 pomocných hodnocení. Jako Swiss-Manager obsahuje všechna, která byla v předchozí kapitole popsána. Na rozdíl od Swiss-Manageru lze zvolit jejich libovolný počet. Párování jednotlivých kol lze provádět automaticky či manuálně. Umožňuje zadat nestandardní výsledky partií. Podobně jako Swiss-Manager umožňuje přidat či odebrat hráče do již probíhajícího turnaje, vracet se k již odehraným kolům a měnit jejich výsledky. Také umožňuje tisk a export důležitých turnajových dat, jako je například párování daného kola. Zde je velká výhoda této aplikace. Obsahuje totiž textový editor. Ten umožňuje pozměnit jednotlivé informace a upravit jejich formátování. Tyto turnajové informace navíc dokáže exportovat do pdf formátu či Microsoft Office formátu. Aplikace umožňuje ověřit, zda je používána aktuální verze. Pokud však není, tak neumožňuje automatické stažení nejnovější verze. Také aplikace přímo neobsahuje uživatelskou příručku. Tu je nutné vyhledat na webovém portálu aplikace. Co se týče GUI,

tak mi připadá velmi přehledné a s moderním vzhledem. Obsahuje mnohem méně funkcionalit než Swiss-Manager. Což je sice nevýhoda, ale právě proto mi přijde přehlednější.

Pro verzi pro turnaje družstev platí většina věcí jako pro verzi pro turnaje jednotlivců. U verze pro turnaje družstev je omezen výběr pomocných pravidel. Je jich celkem 8. Jako u Swiss-Manageru lze zvolit hlavní hodnocení. Body pro tým jsou představovány zápasovými body nebo jsou představovány součtem bodů všech jeho hráčů. K danému turnaji lze nastavit kolik partií bude každé družstvo v daném kole hrát. Družstva mohou mít registrované i náhradníky.

SwissSys

Zdrojem informací pro tuto aplikaci bude [20]. Aplikace je opět placená. Jako demoverzi nabízí plně funkční aplikaci s omezením dvou kol pro každý turnaj. Aplikace posloužila pro řízení více než 1000 turnajů.

Aplikace nabízí zjednodušení pro uživatele, kteří s ní začínají. Lze totiž zvolit zjednodušené menu. To však obsahuje méně možností. Po zvolení této možnosti je uživatel upozorněn, že až se s aplikací seznámí, měl by si zapnout plnou verzi, aby z aplikace získal co nejvíce. Umožňuje výběr turnaje jednotlivců i družstev. U obojího lze zvolit švýcarský systém, systém každý s každým a vyřazovací systém. Pro švýcarský systém lze zvolit akcelerační verzi Baku. Aplikace obsahuje párovací enginy JaVaFo, bbpPairings a vlastní engine. Při výběru jiného engine než bbpPairings, aplikace upozorňuje, že byla schválena FIDE pouze s tímto enginem. Podobně jako Swiss-Manager či Vega nabízí mnoho pomocných hodnocení. Jedná se v podstatě o identická pomocná hodnocení až na pár výjimek. Výhodou této aplikace je, že považuje turnaj jako množinu turnajových sekcí. Tuto funkcionalitu lze využít pro turnaje, které se skládají z různých turnajových skupin. Například skupina A hraje švýcarským systémem a skupina B systémem každý s každým. V aplikaci je pak možné vytvořit jediný turnaj, který se skládá ze dvou sekcí. Obecně je počet sekcí neomezený. Lze také přesouvat hráče z jedné sekce do druhé. Jako u předchozích aplikací je možné zařadit do turnaje pozdní účastníky, vyloučit hráče na určité kolo či z celého turnaje. Je možno vytisknout jakákoli turnajová data včetně párování kol. Kromě tisku je možné je také exportovat do xls, html či txt souborového formátu. Výhodou této aplikace je možnost vytvoření hlavičky pgn formátu párování libovolného kola. Nevýhodou však je, že neobsahuje možnost zjistit, zda je právě používána aktuální verze. Obsahuje pouze možnost navštívit oficiální portál produktu. GUI na mě působí velmi starým stylem. Swiss-Manager na mě působí moderněji. Přehlednost je přibližně stejná jako u Swiss-Manageru.

JavaPairing

Zdrojem informací pro tuto aplikaci bude [21]. Tato aplikace je zdarma a open-

source pod licencí GPLv3. Je naimplementována pouze v Javě a díky tomu je multiplatformní. Jediná podmínka pro její spuštění je mít nainstalovanou JRE verzi 1.6 nebo vyšší. Pro grafické uživatelské rozhraní byl použít Java AWT a Java Swing.

Aplikace umožňuje řídit turnaje pro jednotlivce i pro družstva. Pro oba typy turnajů nabízí stejné turnajové systémy a pomocná hodnocení. Jedná se o různé typy švýcarského systému, včetně oficiálního, systém každý s každým, manuální systém, a ještě jeden speciální. Při manuálním systému provádí párování uživatel aplikace. Švýcarský systém lze volit i v jeho akcelerační verzi. Lze zvolit celkem až 6 pomocných hodnocení. Jedná se o Buchholz, Buchholz Cut 1, střední Buchholz, Sonneborn-Berger, vzájemné zápasy, počet výher, počet her hraných černými figurami a 4 speciální. Párování kol provádí interní engine. Pouze při zvolení manuálního systému lze vytvářet párování ručně. Aplikace umožňuje zobrazit turnajová data, jejich tisk, vytvoření html souboru, či zkopírování jako prostý text. Nevýhodou aplikace je, že oproti ostatním aplikacím obsahuje málo funkcionalit. Co se týče GUI, tak mi přijde vzhledem moderní. Je porovnatelné s GUI aplikace Vega. Vzhledem k tomu, že aplikace obsahuje málo funkcí, tak mi přijde relativně přehledná. Aplikace je open-source, takže se lze podívat na její zdrojové kódy. Obsahuje pouze dva java soubory. Soubor Main.java slouží pro spuštění programu a inicializaci GUI. Soubor EnterFrame.java obsahuje hlavní jádro programu. Po zhlédnutí těchto zdrojových kódů, lze prohlásit, že implementace aplikace je velmi nepřehledná a nesplňuje základní pravidla objektového či procedurálního návrhu. Aplikace také míchá prezentační vrstvu s logikou aplikace. Celá aplikace obsahuje pouze dvě třídy Main a EnterFrame. Třída EnterFrame obsahuje spousty vnitřních a statických vnitřních tříd. Jedná se především o rozšíření jednotlivých prvků GUI. Celá tato třída představuje jádro aplikace. Stará se o GUI a vlastní párování. Pro párování kol je zodpovědná vnitřní třída doNextRound, která je spouštěna jako nové vlákno. Z názvu této třídy je vidět, že nedodržuje ani konvenci pro pojmenování tříd v jazyce Java. Je naimplementována velmi nepřehledně. Kód se často opakuje, je velmi rozvětvený apod. I přesto, že aplikace je velmi malá s málo funkčnostmi, tak její udržování musí být náročné.

2.1.2 Aplikace dosud neschválená FIDE

stChess

Zdrojem informací pro tuto aplikaci bude [22]. Aplikace je zdarma. Není však na rozdíl od JavaPairing open-source. Umožňuje zvolit pro turnaje jednotlivců systém každý s každým, vyřazovací systém a švýcarský systém. Pro turnaje družstev umožňuje pouze systém každý s každým. Pro tento systém zvládne řídit turnaje až pro 40 hráčů či družstev, pro vyřazovací systém zvládne až 64 hráčů a pro švýcarský systém zvládne až 140 hráčů. Přestože tato aplikace není dosud schválená FIDE, je párování švýcarského systému založeno na

Tabulka 2.1: Základní přehled popsaných aplikací

Aplikace	Nepodporuje	GUI	open-source	plně zdarma
Swiss-Manager	–	zastaralé	ne	ne
Vega	vyřazovací systém	solidní	ne	ne
SwissSys	–	velmi zastaralé	ne	ne
JavaPairing	–	solidní	ano	ano
stChess	–	velmi zastaralé	ne	ano

FIDE pravidlech pro párovaní kol pro švýcarský systém. Jako u ostatních aplikací nabízí možnost přidat či odebrat hráče během rozehraného turnaje. Pro spuštění aplikace je nutné mít na počítači nainstalován .Net Framework V2.0.

Aplikace neumožňuje zvolit si pomocná hodnocení. Lze volit pouze mezi danými pomocnými hodnoceními a mezi ručním nastavením pořadí hráčů. Daná pomocná hodnocení jsou podle pořadí Buchholz, referenční Buchholz, stejné pořadí, Sonnenborn-Berger a jedno speciální. Aplikace umožňuje zobrazit turnajová data, jejich tisk a vytvoření html souboru. Dále obsahuje funkce pro změny v grafickém uživatelském prostředí, které však nemají na řízení turnajů vliv. Nevýhodou aplikace je neustálé zobrazovaní reklamy ve formě hypertextových odkazů. Podobně jako JavaPairing obsahuje málo funkcionalit. Dokonce jich má méně. GUI vypadá ze všech popsaných aplikací nejhůře. Také neumožňuje report FIDE ani žádné národní šachové federaci.

2.1.3 Základní přehled popsaných aplikací

V tabulce 2.1 můžeme pro přehled vidět porovnání popsaných aplikací. Jsou pro přehlednost do ní zaneseny pouze základní údaje. Jedná se o to zda aplikace nepodporuje některý ze základních turnajových systémů. Základním systémem mám na mysli švýcarský systém, systém každý s každým a vyřazovací system. Dalším údajem je můj názor na GUI aplikací. Dále pak zda je aplikace open-source a zda je kompletně zdarma.

2.2 Stanovení požadavků

V této části stanovím funkční i nefunkční požadavky. Vstupem pro jejich výběr bude kapitola 1 a porovnání aplikací z předchozí podkapitoly.

Z kapitoly 1 vyplynulo, co by aplikace měla umět a jaká jsou pravidla pro turnaje. Jde především o zjištění jaké a jakým způsobem se v šachových turnajích používají turnajové systémy, pomocná hodnocení a jaké výsledky partií by měla aplikace umožňovat. Z popisu šachových turnajů také vyplynulo jaké údaje bude třeba v aplikaci evidovat. Jde například o rating hráčů, kapitány jednotlivých družstev, kdo měl v dané partii bílé a kdo černé figury apod.

V kapitole 2.1 byly porovnány existující aplikace. Její výstup také využiji pro stanovení požadavků. Pomocí ní stanovím požadavky tak, aby aplikace uměla nejdůležitější funkcionality jako ostatní aplikace a také, aby byla vhodnější pro cílovou skupinu uživatelů, tedy ředitele amatérských turnajů.

Jelikož bude aplikace určena především pro amatérské turnaje, budou požadavky vybrány tak, aby výsledná aplikace byla intuitivní a jednoduchá na ovládání. Bude však brán důraz na to, aby dokázala plnohodnotně řídit šachové turnaje. Aplikace se tedy nebude počtem funkcionalit blížit například aplikaci Swiss-Manageru. Bude se spíše blížit aplikaci JavaPairing. Oproti ní však bude mít výhodu snadnější spravovatelnosti. Budou zahrnuty i požadavky, které byly zadány v zadání této bakalářské práce.

2.2.1 Funkční požadavky

Funkční požadavky se týkají funkcionalit dané aplikace. Jinými slovy se jedná o to, co aplikace bude umět.

F1 – Správa turnajů
Aplikace bude umožňovat vytvořit nový turnaj, změnit jeho atributy, vymazat či otevřít uložený turnaj. Turnaj bude tvořen hráči nebo družstvy a jednotlivými koly. Zahájení dalšího kola bude možné pouze, pokud není žádné probíhající kolo. Bude také možné se vrátit k jakémukoli kolu a upravit jeho výsledky.

F2 – Správa jednotlivých kol
Po zahájení kola bude možné provést párovaní hráčů, kteří byli nasazeni pro toto kolo. Kdykoli bude možné během probíhajícího kola jej zrušit, restartovat a zkontrolovat, zda byly výsledky partií správně zadány. Ke každé partii bude možné každému hráči dát buď 1 nebo 1/2 nebo 0 bodů.

F3 – Zobrazení turnajových dat a jejich tisk
Bude umožňovat zobrazit pro každý turnaj průběžná pořadí a výsledky jednotlivých kol. Tyto údaje bude pak možno vytisknout.

F4 – Export a import
Umožní export účastníků turnaje. Pomocí tohoto exportu bude pak možné je naimportovat do jiného turnaje. Bude také možné nahrát jakýkoli soubor ke každému kolu. Tento soubor bude pak možné stáhnout při prohlížení daného kola.

F5 – Možnost zvolení standardních pomocných hodnocení
Aplikace umožní volbu libovolných pomocných hodnocení popsaných v kapitole 1 a to v jakémkoli počtu.

F6 – Správa účastníků turnaje
Hráče bude možné přidat do turnaje, který ještě nebyl zahájen. Pro švýcarský systém a jeho Baku verzi, je však bude možné přidat i během turnaje a to před párováním 4. kola. Před zahájením i během turnaje bude možné hráče suspendovat. Tento hráč pak nebude nasazen do dalšího kola. Export a import účastníků již byl zmíněn v požadavku F4.

F7 – Podpora češtiny a angličtiny
Uživatel bude mít možnost zvolit si jazyk uživatelského rozhraní. Bude implementována čeština a angličtina.

F8 – Podpora standardních turnajových systémů
Bude implementován jednokolový a dvoukolový systém každý s každým, vyřazovací systém a holandská varianta švýcarského systému, a to včetně Baku akcelerace. Párování švýcarského systému bude implementováno dle platných pravidel FIDE. Všechny systémy budou implementovány jak pro turnaje jednotlivců, tak pro turnaje družstev.

2.2.2 Nefunkční požadavky

Nefunkční požadavky se týkají požadavků, které se netýkají funkcí dané aplikace. Jedná se o požadavky na grafické uživatelské rozhraní, na jakých zařízeních bude aplikaci možné spustit apod.

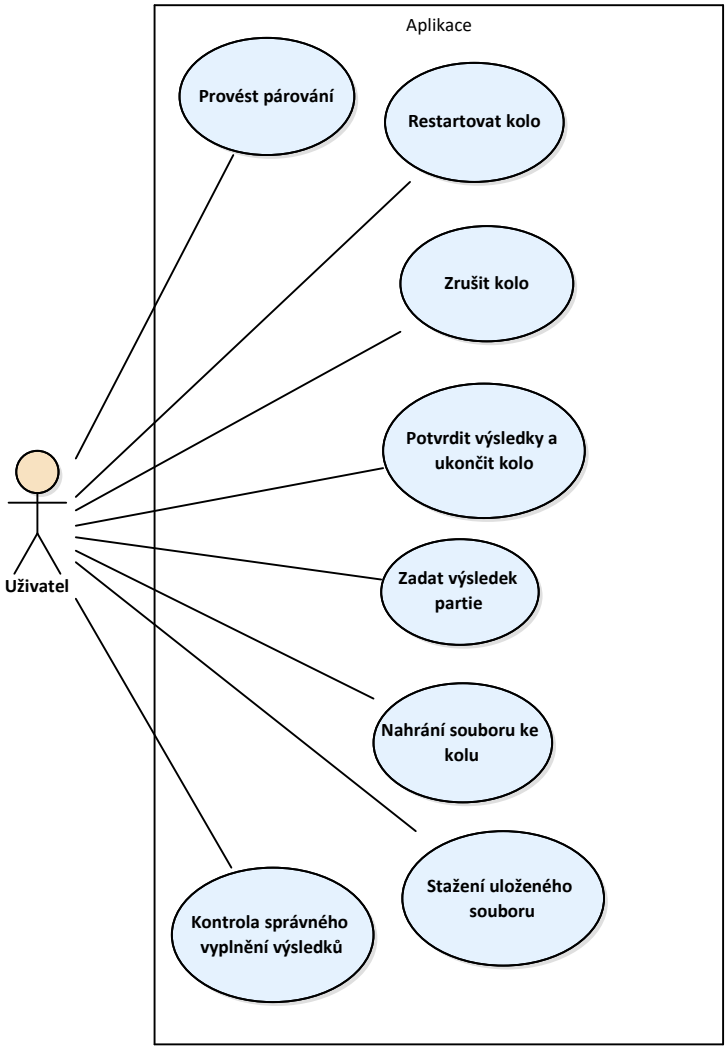
N1 – Aplikace bude desktopová
Aplikaci bude možné spustit na desktopu.

N2 – Aplikace bude multiplatformní
Aplikaci bude možno spustit na libovolném operačním systému.

N3 – Intuitivní grafické uživatelské rozhraní s moderním vzhledem
Grafické uživatelské rozhraní bude navrženo tak, aby bylo jednoduché na ovládání. Budou použity moderní technologie pro tvorbu tohoto rozhraní.

2.3 Případy užití

Stanovené požadavky jsou relativně přímočaré. Z toho důvodu nebudu jednotlivé požadavky rozebírat pomocí případů užití. Pro přehlednost však můžeme na obrázku 2.1 vidět všechny případy užití pro jednotlivá kola turnaje jednotlivců. Pro kola turnaje družstev jsou případy užití identické až na zádávání výsledků. Místo zadání výsledku zápasu družstev, zde bude případ užití zobrazení vlastních partií hráčů. A až po zobrazení tohoto seznamu partií, bude uživatel moci zadat výsledek jednotlivých partií.



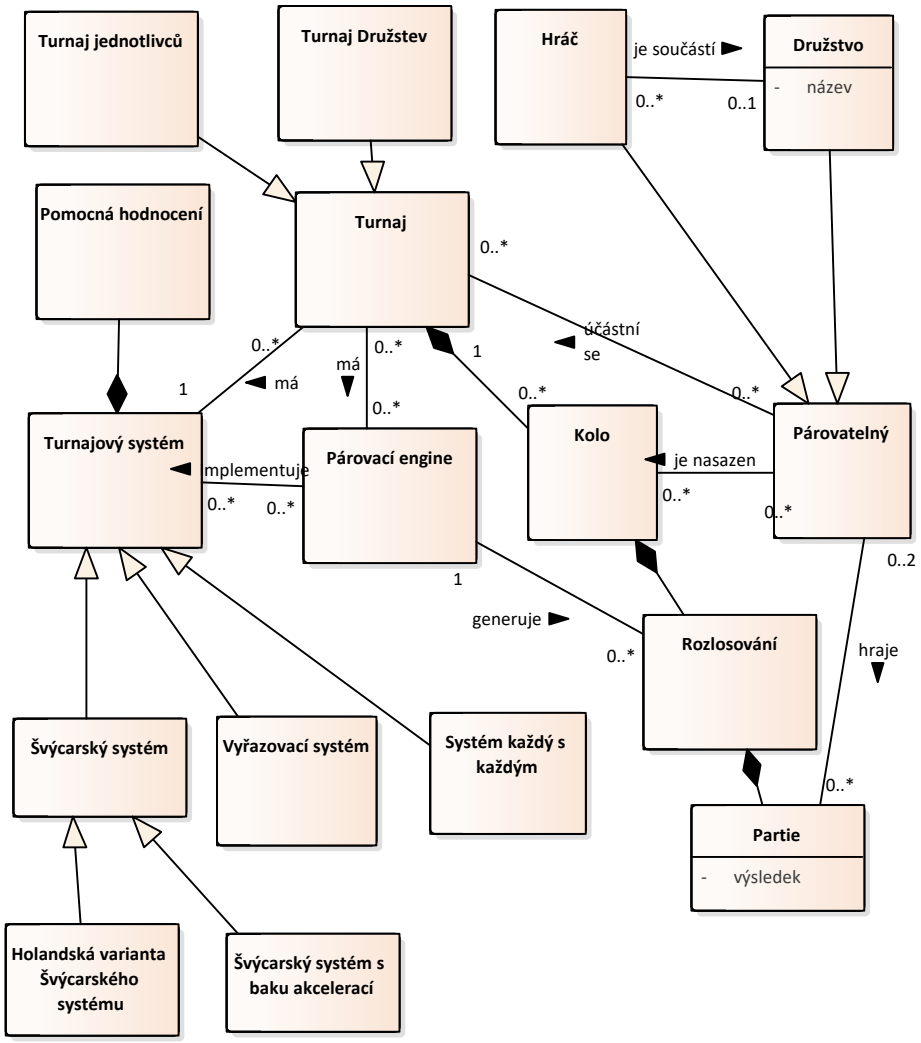
Obrázek 2.1: Případy užití pro jednotlivá kola turnaje jednotlivců

2.4 Doménový model

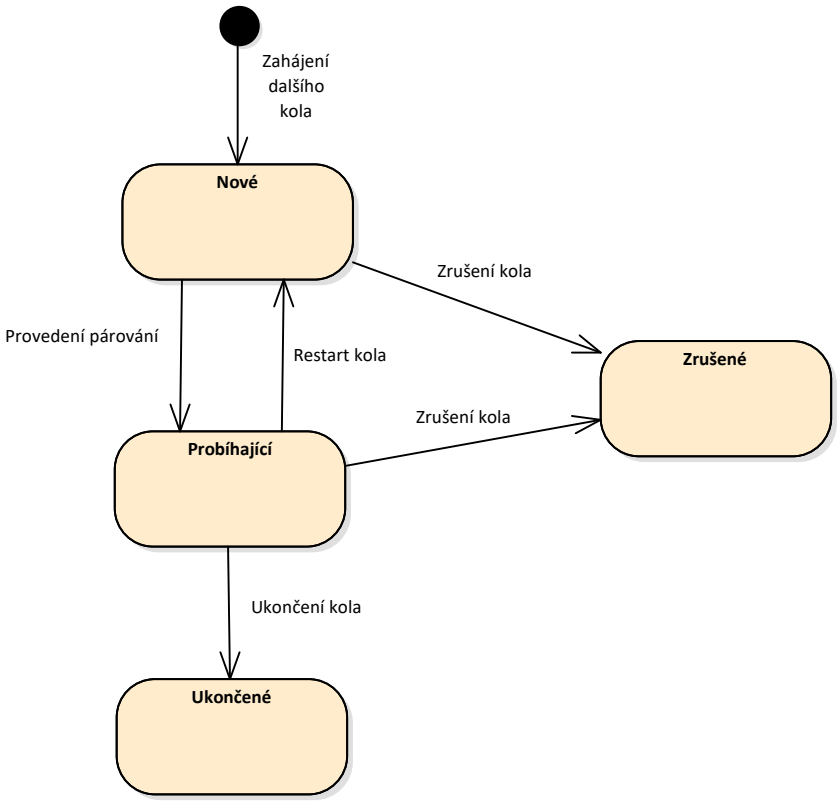
V této sekci popíši hlavní entity z problémové domény. Tyto entity jsou zachyceny na obrázku 2.2. Jedná se o UML diagram tříd. Jsou na něm z důvodu přehlednosti zachyceny pouze nejdůležitější entity. Také ze stejného důvodu nebyly brány v úvahu atributy. V kapitole 3 budou obě věci rozpracovány detailněji. Tento model bude jedním ze vstupů pro kapitolu 3. Z obrázku je vidět, že jsou dva typy turnajů. Jedná se o turnaj jednotlivců a družstev. Turnaje se líší pouze jejich účastníky, tedy buď osoby neboli hráči nebo družstva. Jinak mají stejné atributy a vztahy s ostatními entitami. Každý turnaj musí mít určen turnajový systém a párovací engine. Turnajových systémů je více a byli již vyjmenovány. Všechny systémy mají definované pomocné hodnocení, které lze pro ně použít. Turnaje se účastní hráči nebo družstva. Tito účastníci mají společné to, že jsou pak nasazováni do jednotlivých kol turnaje. Proto jsem pojmenoval tuto entitu jako párovatelný. Párování pak je provedeno pomocí párovacího enginu. Výsledkem párování jsou pak jednotlivé partie účastníků turnaje. Role hráčů se liší podle typu turnaje. V turnaji jednotlivců vystupují jako samostatná entita pro účely párování a pořadí. Zatímco v turnaji družstev jsou vždy součástní družstva. V tomto typu turnaje totiž jako jednotka pro párování kola vystupuje družstvo. Nicméně hráči družstva jsou v kolech párováni s hráči druhého družstva. V jednotlivých kolech tedy hraje družstvo proti družstvu. Nicméně vlastní šachové hry jsou samozřejmě hrány hráči jednoho družstva proti hráčům druhého. Co se týče osob a rolí, tak v modelu se vyskytuje pouze entita hráč. Role jako ředitel turnaje nebo kapitán družstva nebyly z důvodu přehlednosti zachyceny. Budou totiž pouze atributem daných entit. Například kapitán družstva bude pouze atributem entity družstvo.

Na obrázku 2.3 je znázorněn UML stavový diagram pro jednotlivé stavy kola. Diagram začíná zahájením neboli vytvořením nového kola. Po provedení párování je kolo zahájeno a lze k jednotlivým zápasům přiřazovat výsledky. Pokud se dané kolo restartuje, bude ve stejném stavu jako po jeho vytvoření. Pokud je kolo ve stavu připraveném pro párování nebo již probíhá, lze ho zrušit. Zrušené kolo již nelze obnovit. Probíhající kolo je po zadání všech výsledků možno ukončit. Tím se pak stává součástí turnaje a již nelze smazat.

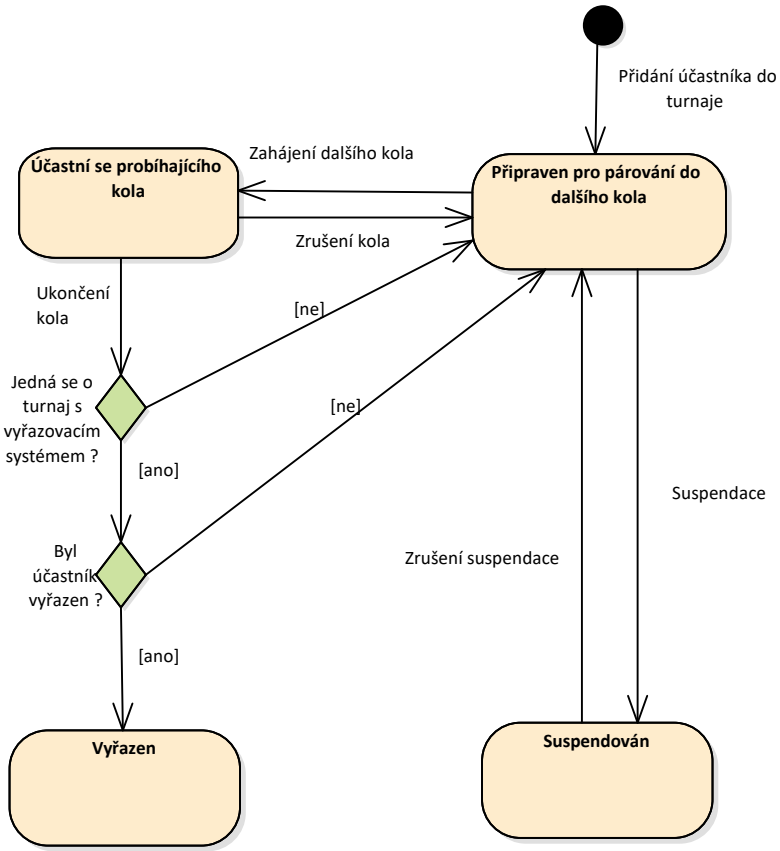
Na obrázku 2.4 je opět znázorněn UML stavový diagram, nyní však pro znázornění stavů účastníků turnaje. Pro zjednodušení není zachycen případ, kdy je turnaj již ukončen. Po přidání do turnaje, je daný účastník připraven pro nasazení do dalšího kola. Pokud však před nasazením do dalšího kola byl hráč suspendován, nezúčastní se tohoto kola ani kol následujících, dokud se nezruší jeho suspendace. Pokud je připraven pro nasazení do dalšího kola a dané kolo bylo zahájeno, zúčastní se ho. Po ukončení tohoto kola se kromě u turnajů s vyřazovacím systémem vrátí vždy do stavu připraven pro párování do dalšího kola. U turnajů s vyřazovacím systémem, záleží na výsledku účastníka v daném kole. Pokud prohrál, přechází natrvalo do stavu vyřazen.



Obrázek 2.2: Doménový model



Obrázek 2.3: Stavový diagram kola turnaje



Obrázek 2.4: Stavový diagram účastníka turnaje

Kapitola 3

Návrh

V této kapitole se budu věnovat návrhu architektury aplikace a provedu výběr technologií pro implementaci. Hlavním vstupem bude výstup předchozí kapitoly, tedy analýzy. Pro popisky v UML diagramech již nebude používána čeština ale angličtina. Je to z toho důvodu, že jednotlivé elementy budou většinou představovat nějaké elementy ve zdrojových souborech aplikace. Při programování bude totiž používána angličtina.

3.1 Verze Javy a GUI

Verze Javy byla vybrána Java SE 8, jelikož je pro účely aplikace dostačující. Vyšší verze nebyla zvolena, protože budoucí uživatelé aplikace mohou mít stále nainstalovanou verzi 8. Pro testování budou použity jednotkové testy pomocí JUnit verze 5.

Pro GUI bylo možno vybírat z více možností. Jedná se například o AWT, Swing, SWT, JavaFX. Zvolil jsem JavaFX, jelikož je to nejnovější GUI framework od firmy Oracle Corporation. Pro statickou definici GUI lze využít FXML formát souborů, čímž lze oddělit tuto definici s vlastním programováním funkčností. Tento formát souborů je rozšířením formátu XML. Lze ho upravovat manuálně či použít speciální aplikaci. Firma Oracle Corporation nabízí aplikaci JavaFX Scene Builder. Tuto aplikaci budu využívat, jelikož velmi usnadní práci.

3.2 Architektura aplikace

Aplikace bude dle nefunkčních požadavků desktopová a multiplatformní. Všechny její části budou běžet na jednom počítači. Multiplatformita bude zaručena použitím programovacího jazyku Java. Bude také Pro rozdělení softwarových modulů jsem zvolil třívrstvou architekturu, obsahující prezentační, business a datovou vrstvu. Přesněji řečeno se jedná o její relaxovanou verzi,

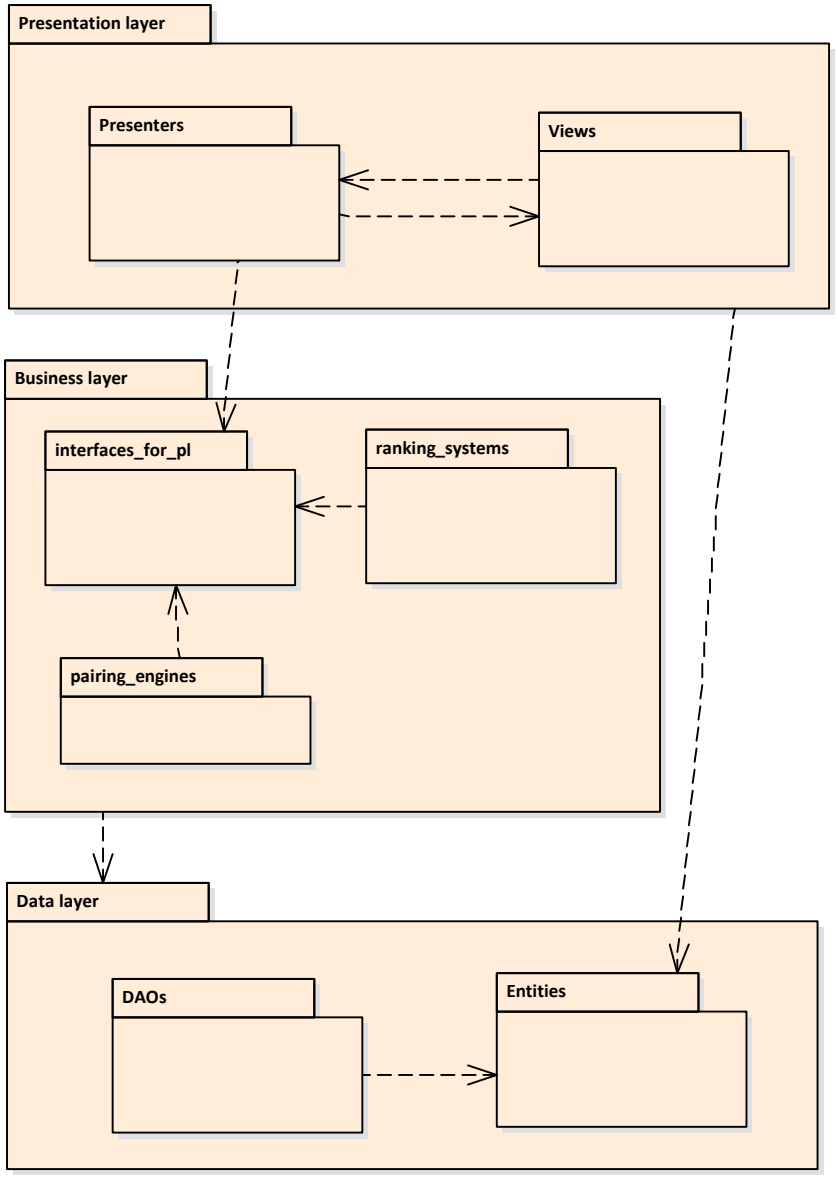
kdy prezentační vrstva může využívat entity z datové vrstvy. Výhodou je jednodušší implementace. Jinak by například musely být entity mapovány na jiné objekty v business vrstvě. Na obrázku 3.1 je UML diagram zobrazující všechny vrstvy aplikace. Jedná se o UML diagram balíčků. Třídy za řízení turnaje a jeho kol jsou v business vrstvě neboli v balíčku Business layer. Třívrstvou architekturu jsem zvolil, abych rozdělil zodpovědnost za GUI, business logiku a práci s perzistentním úložištěm. Jednotlivé vrstvy bude možno snadno vyměnit. Například se v budoucnu může uvažovat o změně desktopového GUI na webové rozhraní. V následujících podkapitolách popíši jednotlivé vrstvy. Budu nejvíce používat UML diagram tříd. Z důvodu přehlednosti budu zanášet do těchto diagramů pouze nejdůležitější atributy a metody.

3.2.1 Prezentační vrstva

Účelem této vrstvy bude zobrazování dat uživateli a reakce na jeho požadavky. Tedy například reakce na stisknutí tlačítka nebo zobrazení seznamu hráčů turnaje. Dle [23] je doporučeno použití vzoru MVC, kde FXML soubory představují jednotlivé pohledy a kontrolery pak reagují na akce v GUI. Dle mého názoru však v tomto případě FXML soubory představují v podstatě pouze statické šablony. Proto jsem zvolil MVP, který se mi zdá zde lepší volba. Prezentační vrstvu rozdělením na pohledy a prezentery. Obě části budou tvořeny třídami. Pohledy budou zachytávat požadavky z GUI a jejich vlastní zpracování delegují na prezenter. Prezenter zkontroluje daný požadavek a pomocí business vrstvy jej zpracuje. Pokud je poté potřeba zobrazit uživateli informace, pak přikáže pohledu, aby je zobrazil v GUI. Jedná se variantu MVP s pasivními pohledy, kde pohledy nekomunikují s objekty z business vrstvy. Obsahují minimální logiku a v podstatě jen zachytávají akce v GUI a zobrazují data. Pohledy obsahují objekty z JavaFX, kdežto prezentery ne. Tím je dosaženo ideálního prostředí pro budoucí testování, kdy lze testovat aplikaci bez interakce s GUI, jelikož flow logika zpracování požadavků je přítomna jen v prezenterech. To je hlavní důvod, proč si myslím, že je zde MVP lepší a tedy proč jsem zvolil MVP místo MVC s FXML pohledy. Dále aby se oddělily statické definice GUI od používaní objektů, využívají pohledy FXML. Každý pohled patří jednomu prezenteru a naopak. Pohledy mezi sebou nekomunikují. Vše jde přes prezentery, který mohou mezi sebou komunikovat.

3.2.2 Business vrstva

Tato vrstva je zodpovědná za vlastní logiku aplikace. Poskytuje rozhraní pro prezentační vrstvu. Probíhá zde hlavní validace požadavků. Prezenter z prezentační vrstvy totiž validuje jen základní věci. Jedná se například o to, zda byly všechny políčka vyplněny. Kontrola, zda byly vyplněny správně, probíhá především v této vrstvě. Business vrstva využívá data z datové vrstvy, které



Obrázek 3.1: Architektura aplikace

získáva pomocí DAO rozhraní z této vrstvy. Kromě získávání dat, využívá DAO objekty i pro perzistetní ukládání dat.

Na obrázku 3.2 můžeme vidět hiearchii rozhraní a tříd představujících turnaj a také rozhraní TournamentManager, které je zodpovědné za CRUD operace obou typů turnajů. Na obrázku 3.3 zase hierarchii rozhraní a tříd představujících kolo turnaje a jejich závislost na rozhraní PairingEngine. Jsou zde zaneseny i dvě entitní třídy z datové vrstvy, představujících zápas jednotlivců a družstev, pro znázornění vztahu s koly turnaje. Jedná se o třídy MatchEntity a TeamMatchEntity.

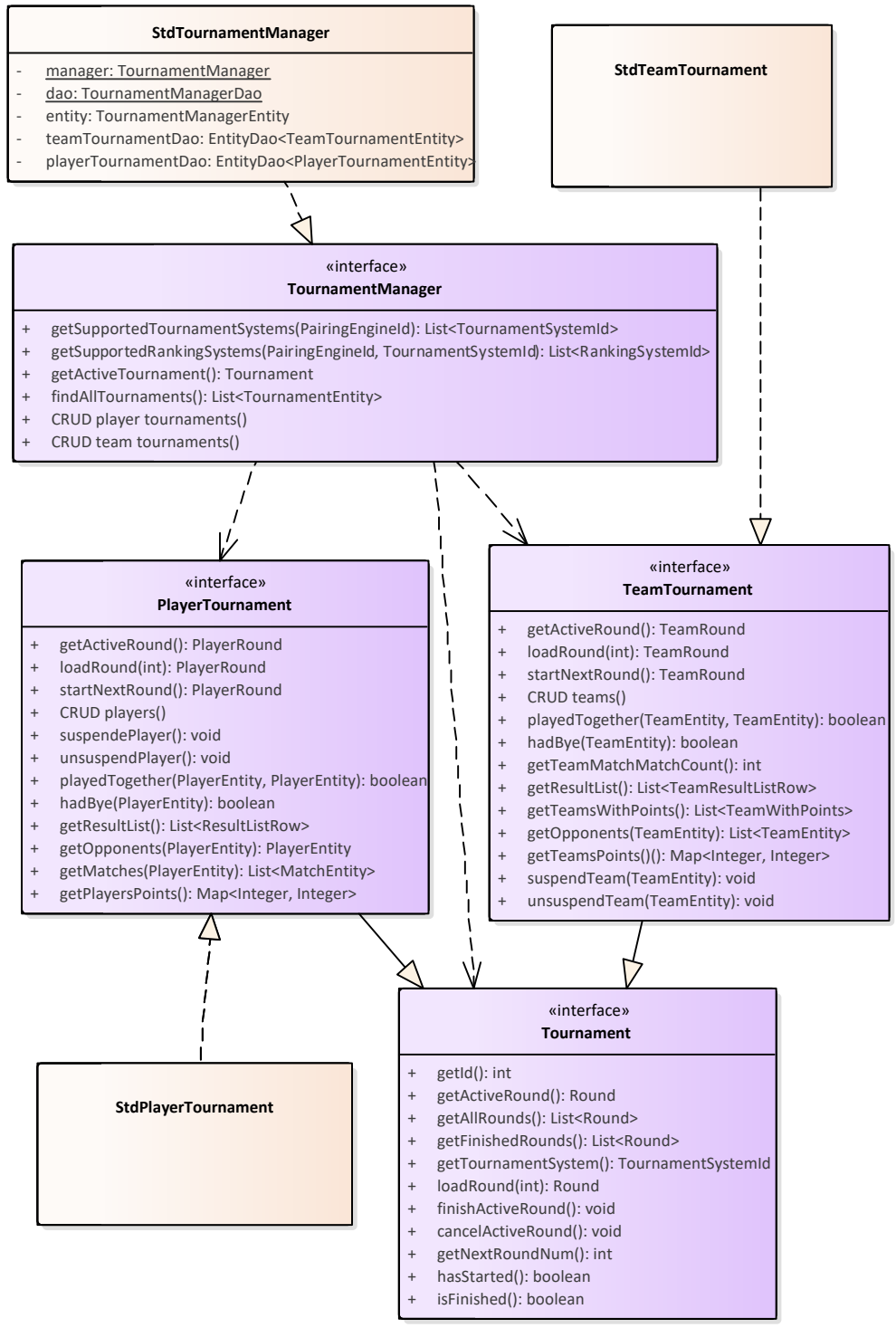
V této vrstvě budou implementovány také párovací enginy a pomocná hodnocení. Jednokolový a dvoukolový systém každý s každým a vyřazovací systém bude implementován vlastním enginem aplikace. Abych zajistil, že párování pro švýcarský systém a jeho Baku akceleraci bude dle pravidel FIDE, bude implementováno externím enginem JaVaFo. JaVaFo lze volat jako samostatný program nebo ho použít jako Java knihovnu. Použiji druhou možnost, jelikož je to dle mého názoru jednodušší.

Jelikož se v budoucnu bude nejspíše uvažovat o přidání párovacího enginu, byl tomu přizpůsoben návrh aplikace. Pro přidání bude stačit pouze vytvořit třídu implementující rozhraní PairingEngine a přidat do výčtového typu PairingEngineId jeho identifikátor. Na obrázku 3.3 můžeme vidět všechny metody, které musí třída představující párovací engine implementovat. Jednoduše se jedná o metodu, která vrací seznam podporovaných turnajových systémů. Dále pak o dvě metody, které provedou párování pro turnaje jednotlivců a družstev. Obě metody mohou vyhodit výjimku TournamentSystemNotSupportedException, která znamená, že daný párovací engine byl požádán o provedení párování pro turnajový systém, který nepodporuje.

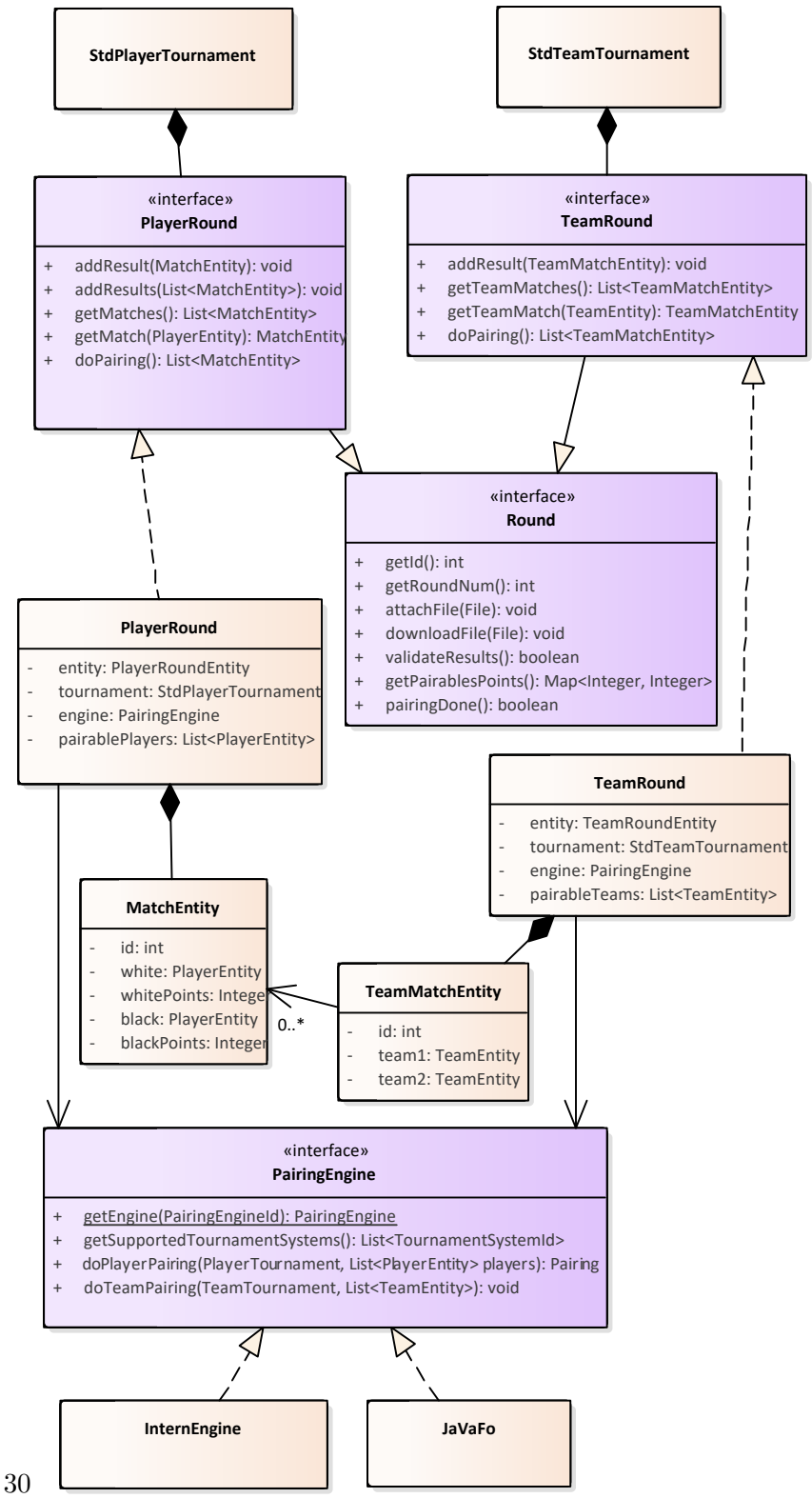
3.2.3 Datová vrstva

Datová vrstva má za úkol především perzistentně ukládat data a také je z tohoto perzistentního prostředí získávat. Kromě toho má za úkol implementaci logování. Využívá objektově relační mapování JPA. Jednotlivé entity jsou mapovány na dané tabulky v relační databázi. Díky této vrstvě lze snadno vyměnit relační databázi za jinou. Pro aplikaci jsem zvolil databázový systém H2, jelikož je jednoduchý a může se použít jako Java knihovna. Pro potřeby aplikace je plně dostačující. CRUD operace s entitami jsou zpřístupněny pro business vrstvu pomocí DAO rozhraní. Díky tomu je možné přejít z relačních databází na jiné perzistentní prostředí, například na nerelační databáze.

Nyní popíši navrženou strukturu databáze. Na obrázku 3.4 můžeme vidět relační datový model. Jelikož entity budou využívat dědičnost, musel jsem rozhodnout, jakým způsobem budou vypadat související tabulky. Potomci entity Tournament se liší velmi málo, proto jsem se v tomto případě zvolit jednu tabulku pro oba potomky. To samé pak platilo pro entitu Round a její potomky. Pro třídu Pairable jsem se rozhodl pro jiný způsob, jelikož její potomci se

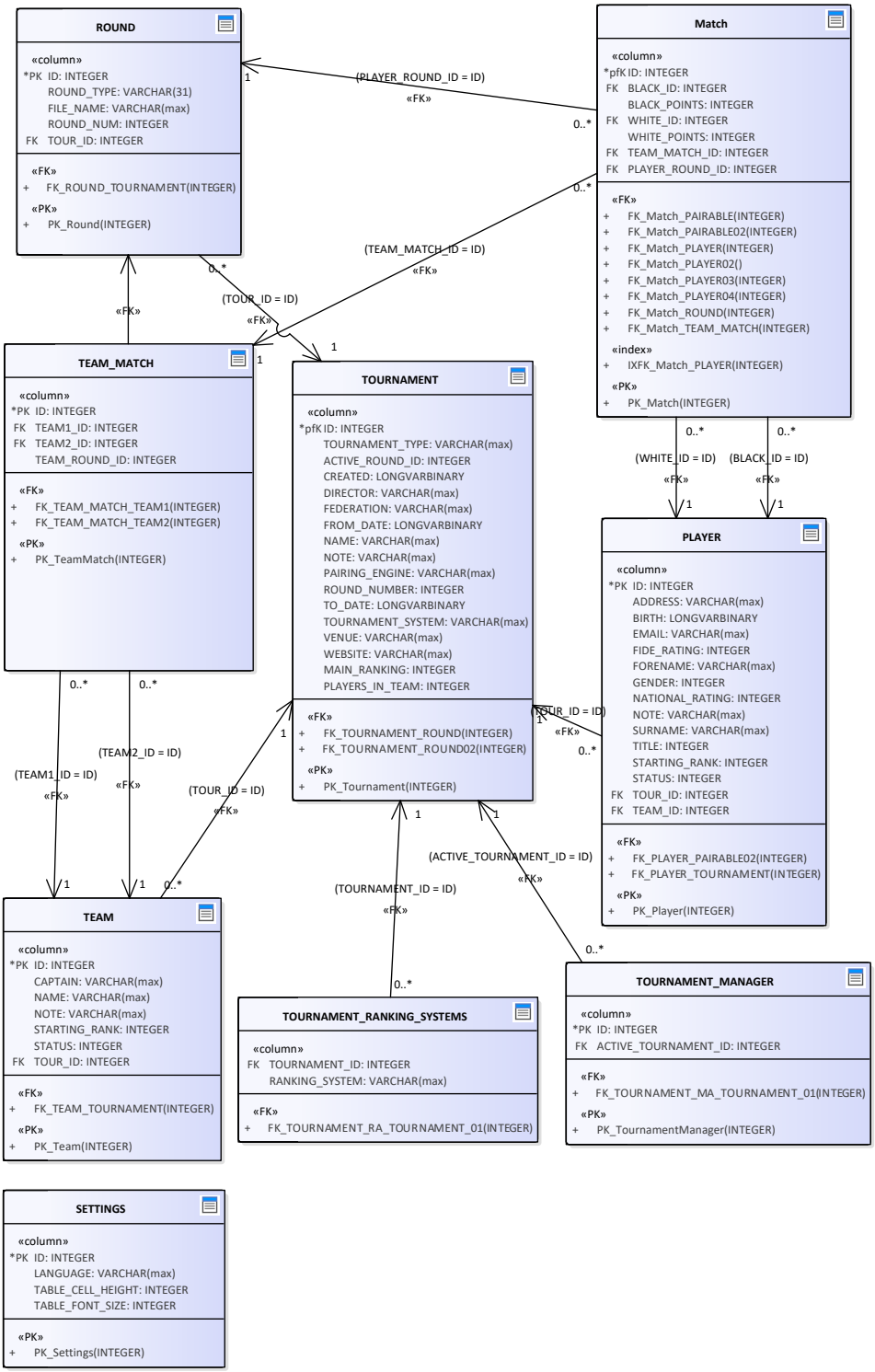


Obrázek 3.2: Hierarchie tříd představujících turnaj a jejich vztah s manažerem turnajů



Obrázek 3.3: Hierarchie tříd představujících kolo a jejich vztah s párovacími enginy

velmi liší. A vzhledem k tomu, že třída Pairable nebude potřeba ukládat do databáze jako samostatná jednotka, rozhodl jsem se pro vytvoření tabulek pouze jejích potomků, tedy entit Player a Team. Není pak nutné, aby byla entitou, a proto budou entity pouze její potomci.



Obrázek 3.4: Relační databázový model

Kapitola 4

Implementace

V této kapitole se zaměřím na vlastní implementaci aplikace. Budu popisovat postupy, které jsem při ní použil. Během implementace také ukáži ukázky kódu a obrázky GUI aplikace. Nakonec popíši strukturu výsledného programu a jak zprovoznit aplikaci na desktopu. Hlavním vstupem bude kapitola 3, kde byla především navrhnuta architektura aplikace a kapitola 2, kde byly stanoveny požadavky na aplikaci. Částečně bude také čerpáno z kapitoly 1, kde jsou popsány jednotlivé turnajové systémy, pomocná hodnocení apod.

4.1 Postupy při implementaci

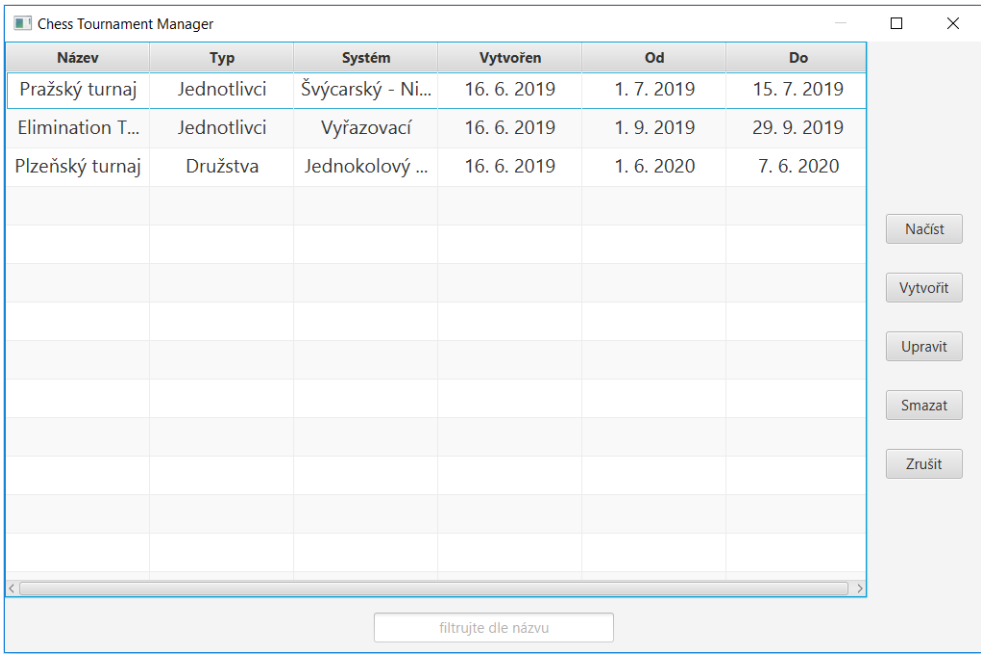
Při implementaci jsem dodržoval architekturu aplikace, která byla navrhnuta v kapitole 3. Jedná se hlavně o její hlavní principy. Příkladem může být zákaz v datové vrstvě používání objektů z jiné vrstvy, neboť datová vrstva není závislá na žádné vrstvě. Kromě toho budu také co nejvíce dodržovat principy OOP. Díky zvolené architektuře je přidávání nových párovacích enginů a pomocných hodnocení velmi jednoduché. Stačí pouze implementovat dané rozhraní, respektive dědit z dané abstraktní třídy a přidat svoje id do daného výčtového typu a následně jen implementovat požadované metody. Díky navržené architektuře jsem mohl implementovat jednotlivé vrstvy téměř paralelně. Budu samozřejmě dodržovat také konvence Javy. Co se týče konvencí, tak například třída bude začínat velkým písmenem, metoda malým apod.

4.2 Grafické uživatelské rozhraní

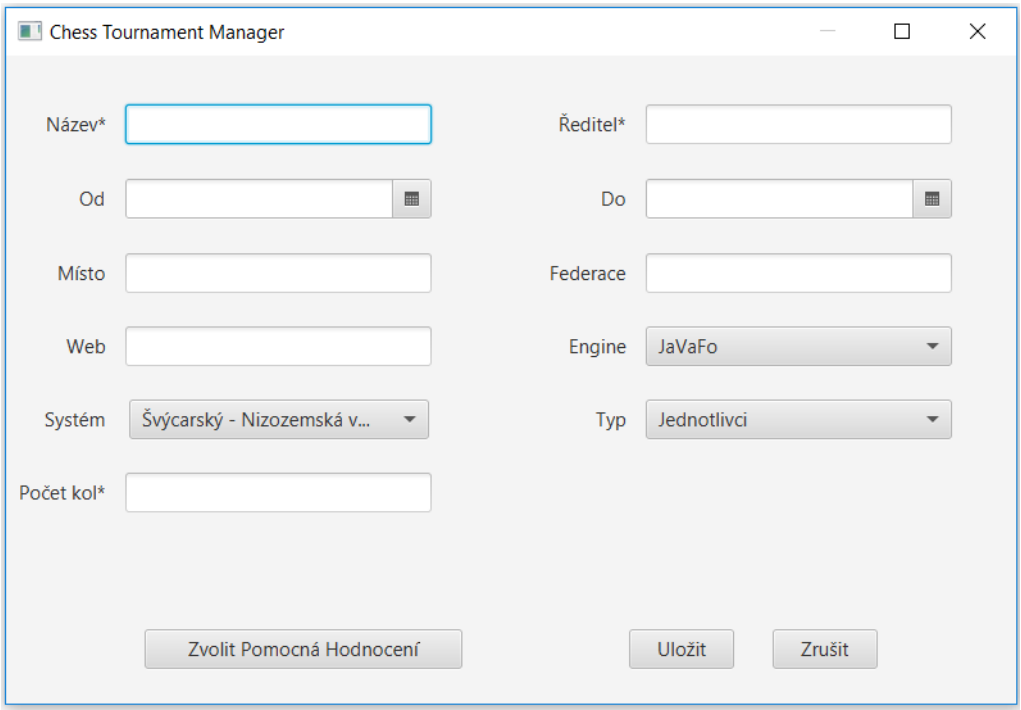
V této části popíši, jak jsem naimplementoval GUI. Pro jeho statickou definici jsem použil FXML, což bylo navrhnuto v předchozí kapitole. Jednotlivá okna můžeme rozdělit do tří sekcí. A to na sekci pro turnaj jednotlivců, sekci pro turnaj družstev a na sekci která patří k oběma turnajům. Teoreticky to šlo

udělat tak, že oba typy turnajů, by byly kompletně rozděleny a nesdílely by tak žádné okno. Nicméně to by způsobilo zbytečnou duplikaci kódu. Příkladem okna, které patří oběma typům, může například být formulář pro vytváření turnaje či okno pro zvolení kola. Nyní popíši nejdůležitější okna aplikace. Jejich velikost nebude úplně souhlasit se skutečností, aby se větší vešly do šířky dokumentu. Nebudou z důvodu přehlednosti popsána všechna okna. Budou především vynechány upozornění a potvrzovací dialogy. Příkladem potvrzovacích dialogů můžou být například dialogy, které nás upozorňují, že dané smazání je trvalé. Ukázky oken budou v češtině.

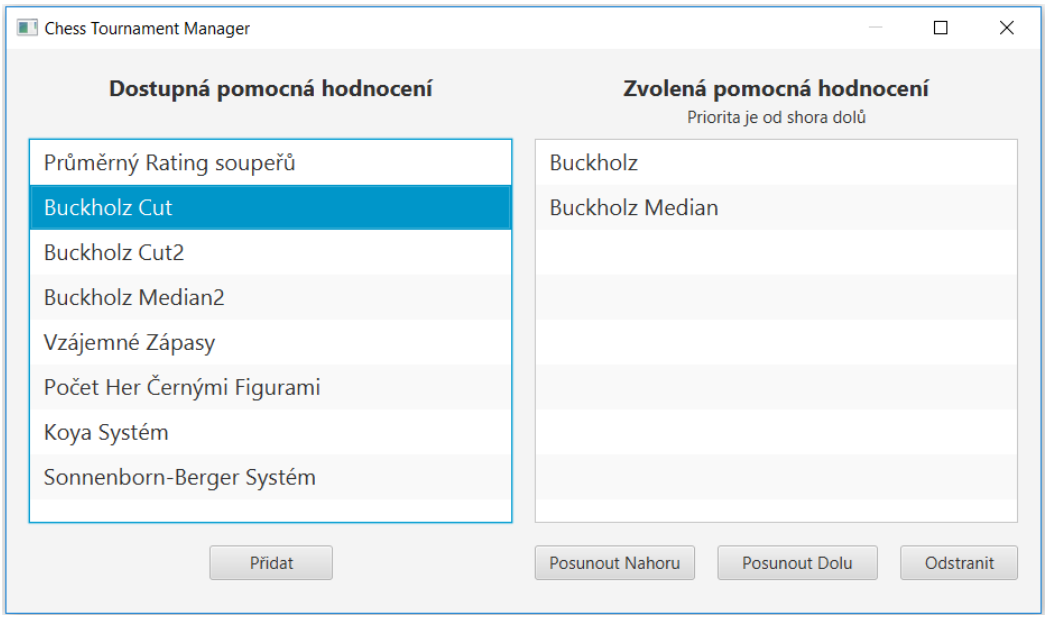
Pokud nebyl dosud žádný turnaj načten, zobrazí se okno pro výběr turnajů. Toto okno můžeme vidět na obrázku 4.1. Můžeme zde načíst, mazat a měnit atributy vytvořených turnajů. Pro mazání je potřeba odsouhlasit tři potvrzovací dialogy. Implementoval jsem to z důvodu bezpečnosti. Smazání turnaje je totiž nevratné a nechtěná ztráta celého turnaje je nejhorší co se může stát. Lze turnaje filtrovat podle jejich názvu. Máme zde také volbu pro vytvoření nového turnaje. Když tuto volbu zvolíme otevře se nám okno, které je zachyceno na obrázku 4.2. Jedná se o formulář, kterým definujeme atributy turnaje. Povinné údaje jsou označeny symbolem * na konci popisku. Pokud se pokusíme uložit turnaj bez jejich vyplnění, zobrazí se upozornění, že jsme nevyplnili povinné údaje a že tyto údaje jsou označeny již zmíněným symbolem. Ve formuláři můžeme vybírat mezi dostupnými párovacími enginy. V současné době jsou dostupný dva. Jedná se o JaVaFo a vlastní engine aplikace. První zmíněný je externí a aplikace ho používá jako Java knihovnu. Pro volání jeho funkcí jsem však musel naimplementovat generování TRF souboru, jelikož ho JaVaFo používá jako jeden ze způsobů vstupu. Interní engine aplikace jsem naimplementoval zcela sám. Pokud zvolíme ve formuláři JaVaFo, ve formuláři budeme moci jako turnajový systém volit dvě verze Švýcarského systému. První je jeho standardní verze nazývána Holandská varianta. Druhá je pak Baku akcelerační verze této Holandské varianty. Pokud zvolíme jakoukoli variantu Švýcarského systému, budeme muset ještě zvolit počet kol, který je libovolný. Co se týče typu turnaje, tak můžeme volit mezi turnajem jednotlivců a turnajem družstev. V případě zvolení turnaje družstev, zobrazí se nám ještě dvě povinné kolonky. Jedná se o zvolení hlavního bodového hodnocení a počtu hráčů každého družstva, kteří budou párováni. Družstvo pak může mít i více hráčů, bude však párován pouze tento počet. Pokud jich bude méně, budou místo jejich chybějících hráčů figurovat tzv. bye, což je zástupce pro neexistujícího hráče či družstvo. Ve formuláři pak ještě máme tlačítko *Zvolit Pomocná hodnocení*. Pokud tuto volbu zvolíme, zobrazí se nám okno, které je zachyceno na obrázku 4.3. Zde pak můžeme vybrat z dostupných pomocných hodnocení pro daný turnajový systém. Pro zjednodušení práce jsem umožnil pro seznam dostupných pomocných hodnocení zvolení více položek najednou. Po stisknutí tlačítka *Přidat* se pak přesunou do seznamu zvolených pomocných hodnocení. Jak můžeme vidět z popisku, je priorita od shora dolů. Aby se ulehčila práce, naimplementoval jsem posun jeho položek nahoru a dolů.



Obrázek 4.1: Okno se seznamem turnajů

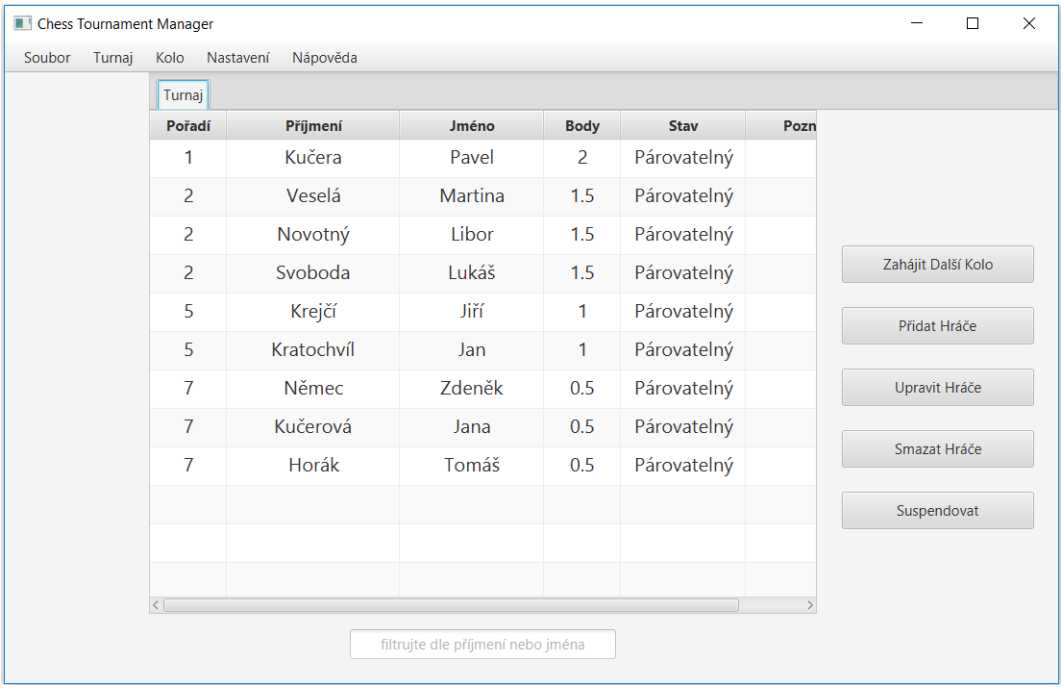


Obrázek 4.2: Okno s turnajovým formulářem



Obrázek 4.3: Okno pro výběr pomocných hodnocení

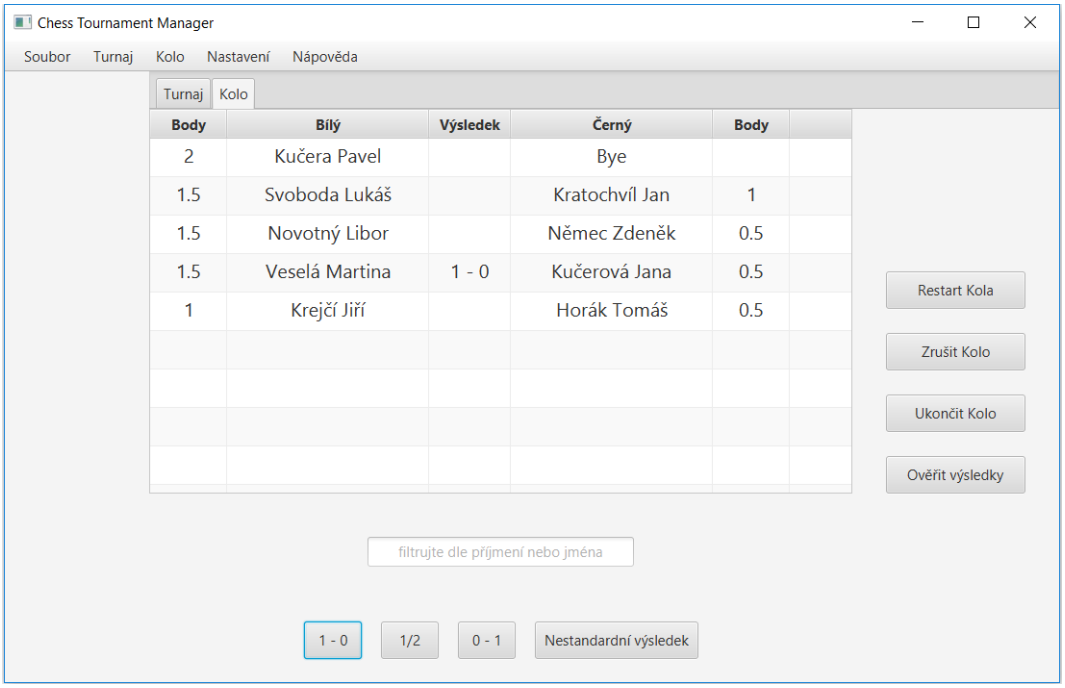
Nyní se vrátím k oknu se seznamem turnajů. Dosud byla okna shodná pro oba typy turnajů. Aplikace totiž po zvolení daného systému dynamicky měnila položky, které je nutné vyplnit. Pokud nyní však načteme libovolný turnaj, zobrazí se okno pro daný typ. Nejprve popíši, jak jsem naimplementoval turnaj jednotlivců a potom družstev. Základní okno pro turnaj jednotlivců je zobrazen na obrázku 4.4. Na tomto obrázku vidíme již rozehraný turnaj. Toto okno slouží především pro řízení hráčů. Můžeme je přidávat, upravovat, mazat, suspendovat a zrušit jejich suspendaci. Pro mazání hráčů musíme potvrdit jeden potvrzovací dialog. Tabulka hráčů umožňuje vícenásobný výber. Opět aby se zjednodušila práce. Při zvolení více hráčů a zvolení volby *Upravit Hráče*, se zobrazí formulář shodný s formulářem pro vytvoření nového hráče, avšak již s vyplněnými hodnotami. Jsou vyplněny pouze položky, které jsou shodné. Při úpravě jakékoli položky, jsou změny aplikovány na všechny vybrané hráče. Naimplementoval jsem tuto funkcionalitu opět z důvodu ušetření práce. Uživatel aplikace může například chtít k několika hráčům přidat stejnou poznámku. Bez této funkcionality by musel upravovat jednotlivé hráče po jednom, což by bylo velmi pracné. Díky této funkcionalitě vybere dané hráče a upraví je všechny na jednou. Hráče lze filtrovat podle jejich jména a příjmení. Pokud zvolíme pouze suspendované hráče nebo mix párovatelných a supendovaných, změní se dynamicky tlačítko *Suspendovat* na *Zrušit suspendaci*. Tabulka hráčů obsahuje pro přehled jen nejdůležitější údaje hráčů. Jedná se o současné pořadí, příjmení, jméno, počet bodů, stav a poznámku. Stavy jsou *Párovatelný*, *Suspendován* a *Eliminován*. Poslední zmíněný je pouze pro



Obrázek 4.4: Hlavní okno turnaje jednotlivců

vyřazovací systém. Pokud je hráč či družstvo vyřazeno, již nelze změnit jeho stav. Stav hráče případně družstva je důležitý pro volbu *Zahájit Další Kolo*. Tato volba totiž vytvoří a zahájí další kolo a umožní párovat hráče pouze se stavem *Párovatelný*. Pokud se v turnaji ještě nezahájilo kolo, bere se jako nezahájený. V opačném případě jako zahájený. To je důležité, jelikož pro zahájený turnaj již nemůžeme mazat hráče apod. Volbu *Zahájit Další Kolo* umožní aplikace pouze v případě, že žádné kolo není aktivní. V opačném případě zobrazí upozornění, že je nutné pro zahájení dalšího kola, ukončit nebo zrušit současné kolo.

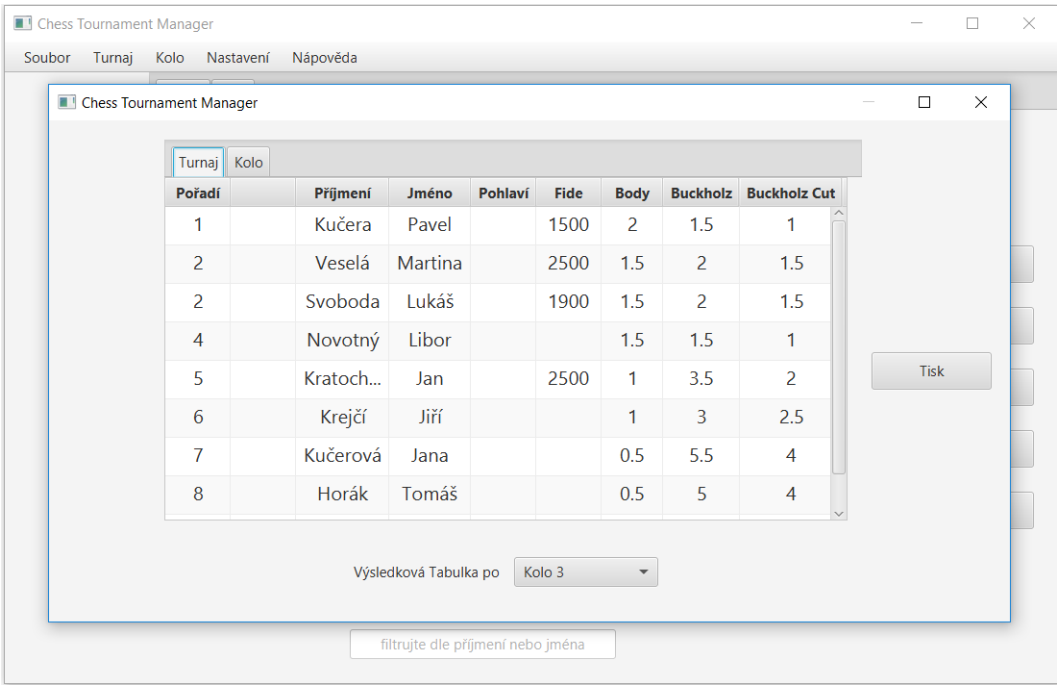
Pokud úspěšně zahájíme další kolo, přidá se záložka v hlavním okně a zobrazí se. Tuto záložku můžeme vidět na obrázku 4.5. Vidíme zde však již kolo, kde bylo provedeno párování hráčů. Než bylo provedeno párování, bylo zde místo tlačítka *Restart Kola* tlačítko *Provést Párování*. Po jeho stisknutí se provedlo párování, které vidíme na obrázku a tlačítko se dynamicky změnilo na tlačítko pro restartování kola. Pokud ho zvolíme, zrušíme dané párování a to včetně již vložených výsledků. Aplikace se nás proto zeptá, zda jsme si toho vědomi. To samé platí pro zrušení kola. V tomto případě je však kolo kompletně zrušeno. Co se týče zadávání výsledků, tak kromě standardních, můžeme zvolit i nestandardní. Pro ověření, zda jsme vyplnili všechny výsledky, můžeme zvolit volbu *Ověřit Výsledky*. Pokud jsme již se zadanými výsledky hotovi a chceme ukončit dané kolo, zvolíme volbu *Ukončit Kolo*. Při zvolení



Obrázek 4.5: Záložka aktivního kola turnaje jednotlivců

této volby aplikace automaticky zkontroluje, zda jsme vložili všechny výsledky partií. Pokud ne, tak nás upozorní a kolo neukončí.

Ostatní možnosti pro turnaj jednotlivců můžeme nalézt v liště s menu. V menu *Soubor* máme volbu pro ukončení aplikace a pro zobrazení seznamu turnajů. Druhá volba otevře modální okno a je stejné jako na obrázku 4.1. Pokud jí zvolíme a vymažeme současný turnaj nebo načteme jiný turnaj, okno současného turnaje se zavře. V menu *urnaj* můžeme zobrazit seznam všech kol. Jakékoli ukončené kolo můžeme načíst a upravit jeho výsledky. Načíst můžeme pouze, pokud turnaj nemá žádné kolo aktivní. Další volbou je zde zobrazení výsledků. Tuto volbu popíši později. Posledními dvěma volbami jsou import a export hráčů. Tato funkcionalita značně usnadní práci, pokud chceme přenést hráče z jednoho turnaje do druhého. Menu *Kolo* se týká pouze aktivního nebo ukončeného kola, které bylo načteno. Umožňuje připojit ke kolu jakýkoli soubor a stáhnout tento soubor. Stažení bylo naimplementováno tak, že po zvolení adresáře, se soubor uloží do tohoto adresáře s původním názvem. Soubory se ukládají do speciálního adresáře. Toto bude popsáno podrobněji až budu popisovat adresářovou strukturu aplikace. Menu *Nastavení* umožňuje měnit nastavení uživatelského rozhraní. Lze volit jazyk, výšku buněk tabulek a velikost fontu v tabulkách. Lze také nastavit výchozí nastavení. Po uložení zvoleného nastavení, aplikace upozorní, že změny budou použity při příštím startu aplikace. Poslední menu *Nápověda* obsahuje základní informace

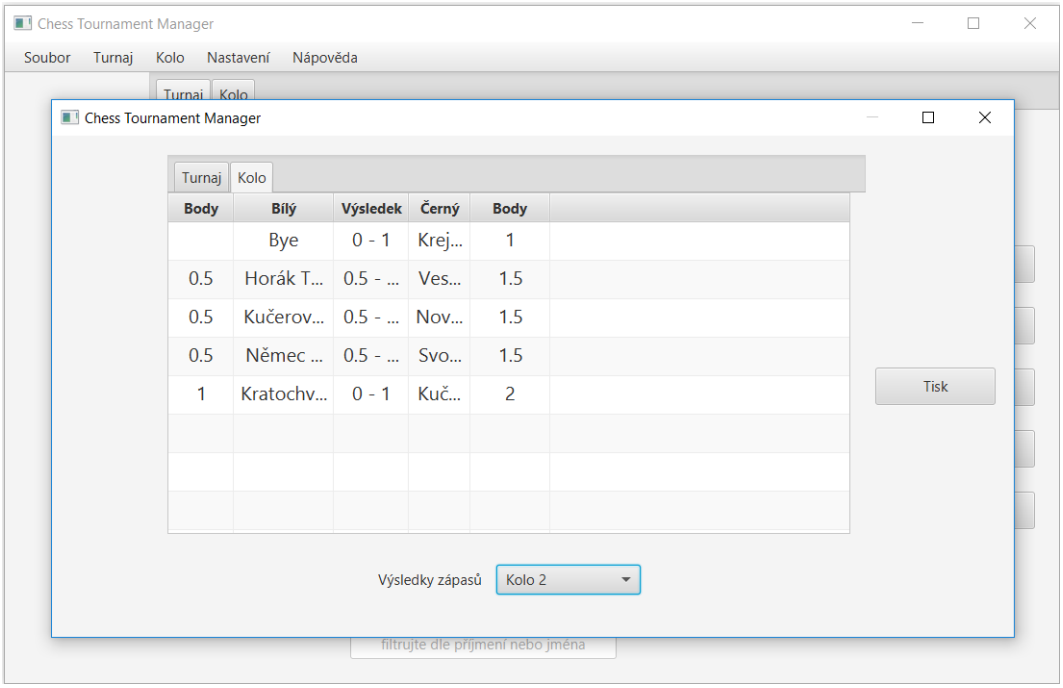


Obrázek 4.6: Okno s výsledkovou tabulkou

pro ovládání aplikace, jak zaktualizovat aplikaci a údaje o aplikaci.

Nyní popíši, jak jsem naimplementoval okno pro zobrazení výsledkové tabulky a výsledků partií v jednotlivých kolech. Okno pro výsledkovou tabulku můžeme vidět na obrázku 4.6. Tabulka je podobná tabulce hráčů v hlavním okně turnaje. Obsahuje však všechny důležité údaje. Také neobsahuje zbytečné údaje, jako je například stav hráče či poznámku. Jsou zde uvedeny všechna zvolená pomocná hodnocení a jejich vypočtené hodnoty. Můžeme volit výsledkovou tabulku po jakémkoli kole. Pokud zvolíme *Kolo 0*, tak se zobrazí startovní listina. Jakoukoli tabulku můžeme vytisknout. Pro zobrazení výsledků jednotlivých kol, musíme přepnout na záložku *Kolo*, která je zobrazena na obrázku 4.7. Volit můžeme výsledky jakéhokoli kola. Opět můžeme jakoukoli tabulku vytisknout. Dokonce můžeme zobrazit právě rozehrané kolo a vytisknout tak párování daného kola.

Nyní přejdu k oknům pro turnaje družstev. Většina věcí je společná. Proto také mají prezentery obou turnajů společného abstraktního předka, který obsahuje společné metody. Budu popisovat především rozdíly mezi oběma turnaji. Na obrázku 4.8 je zobrazeno hlavní okno turnaje družstev. Tabulka družstev je velmi podobná tabulce hráčů. Lze filtrovat družstva pomocí jejich jmen. Volby jsou v podstatě shodné. Je zde však jedna specifická volba a to *Hráči*. Po jejím zvolení se nám zobrazí okno, které je zachyceno na obrázku 4.9. V tomto okně přidáváme, mažeme, upravujeme, suspendujeme a rušíme

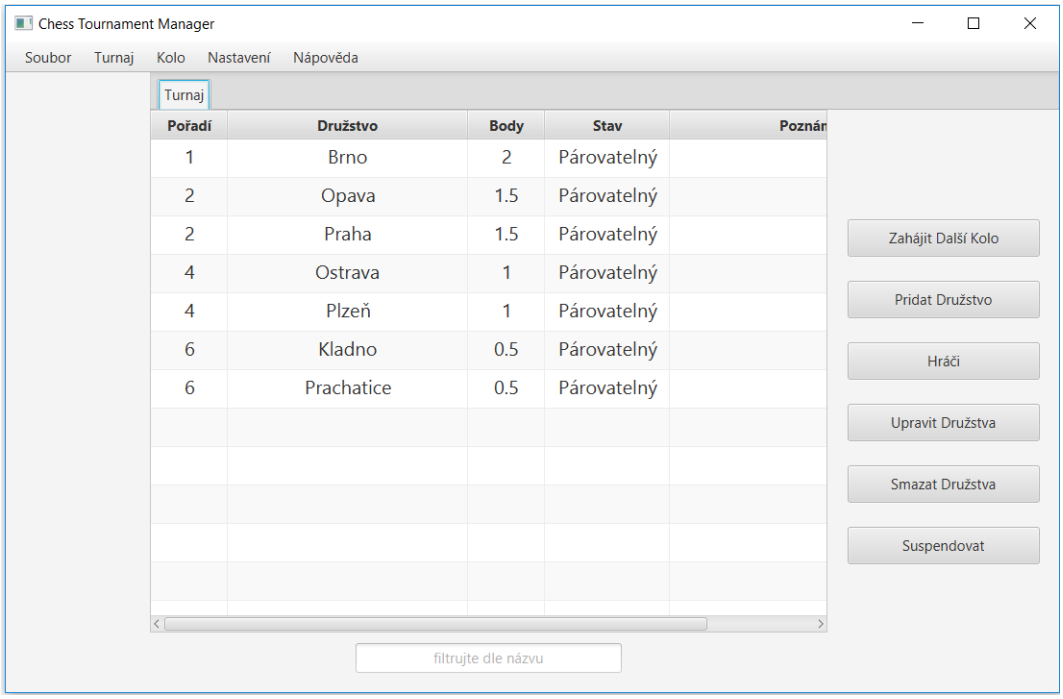


Obrázek 4.7: Okno s výsledky kol

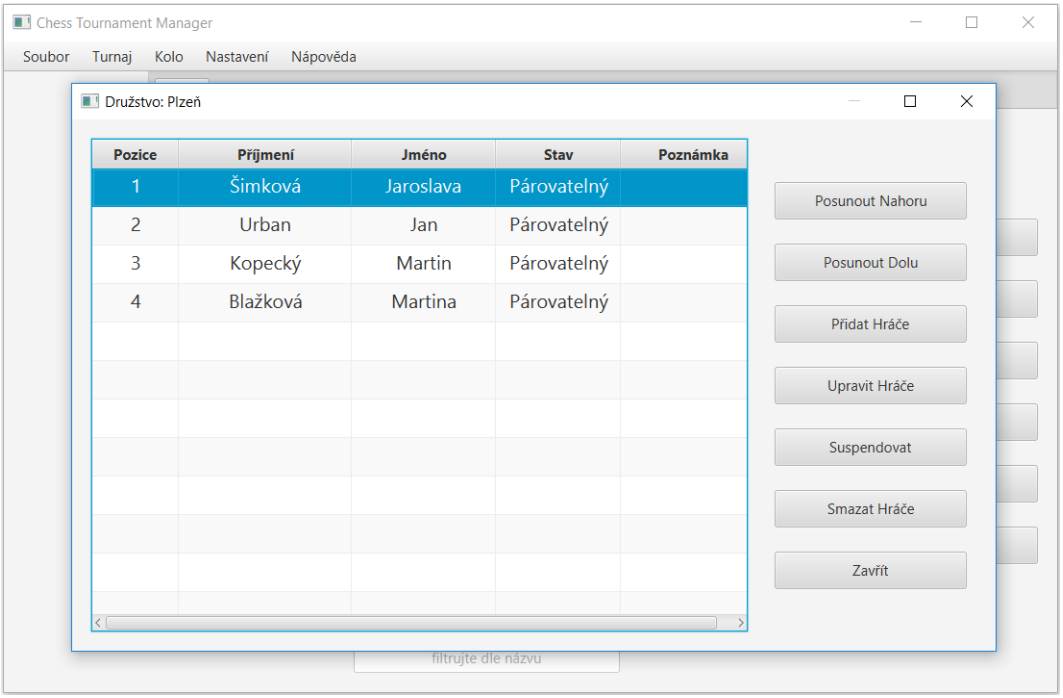
suspendaci hráčů daného družstva. Jelikož zde záleží na pořadí, implementoval jsem zde posun hráče nahoru a dolů. Tabulka je opět více výběrová. Pouze posun funguje po vybrání pouze jediného hráče. Pokud jich zvolíme více a zvolíme posun, upozorní nás aplikace, že je nutné zvolit pouze jednoho hráče.

Na obrázku 4.10 vidíme aktivní kolo, ve kterém proběhlo párování družstev. Volby jsou opět téměř stejné jako u turnaje jednotlivců. Je tu však naimplementována jedna volba navíc. Jedná se o zobrazení vlastních partií hráčů zvoleného zápasu družstev. Tyto partie se zobrazí v novém okně. Toto okno můžeme vidět na obrázku 4.11. V tomto oně zadáváme pouze výsledky jednotlivých partií. Na obrázku lze vidět dvě bye. To je způsobeno tím, že druhé družstvo mělo zadáno pouze dva hráče. Proto místo chybějících dvou hráčů je uvedeno bye. Co se týče lišty s menu, tak je shodné s turnajem jednotlivců.

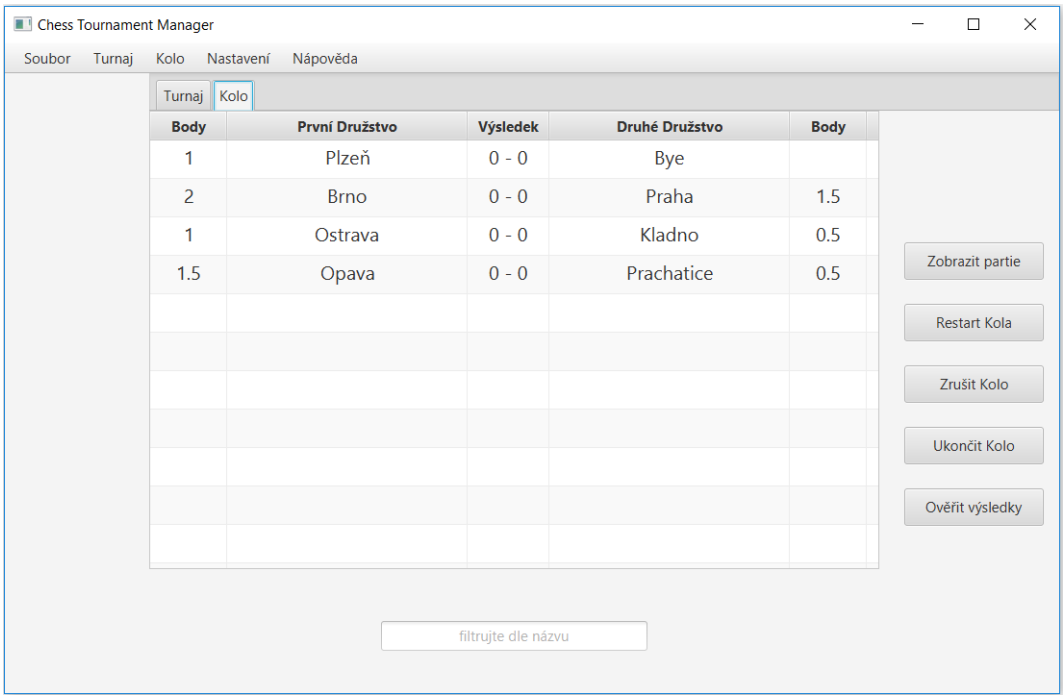
Na závěr popíši jaký je algoritmus pro zobrazení oken při startu aplikace. Pokud spustíme aplikaci a nějaký turnaj již byl načten zobrazí se jeho hlavní okno. Pokud však tento turnaj má aktivní kolo, bude zobrazeno okno tohoto kola. Jak již bylo zmíněno, pokud není žádný turnaj načtený, zobrazí se okno se seznamem turnajů.



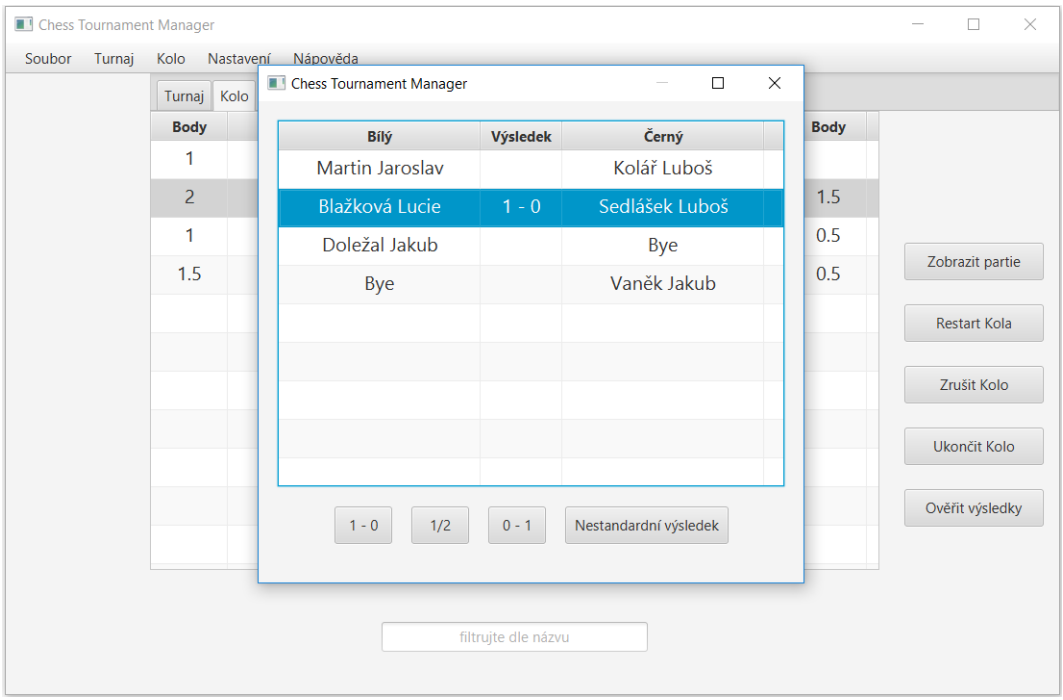
Obrázek 4.8: Hlavní okno turnaje družstev



Obrázek 4.9: Okno pro řízení hráčů daného družstva



Obrázek 4.10: Záložka aktivního kola turnaje družstev



Obrázek 4.11: Okno vlastních partií daného zápasu družstev

4.3 Implementace párování kol

V této části popíši implementaci párování kol. Popíši pouze turnaj jednotlivců, protože pro turnaj družstev, je implementace obdobná. Implementaci jsem provedl přesně podle navržené architektury aplikace. Požadavek na párování je zachycen instancí třídy PlayerRoundView. Ta bez jakéhokoli zpracování předá tento požadavek svému prezenterovi. Jedná se o instanci třídy PlayerRoundPresenter. Ta požádá o párování rozhraní PlayerRound, které již patří do business vrstvy. Tím se požadavek přesunul z prezentační vrstvy do business vrstvy. Prezenter pak očekává vrácení seznamu s párováním. Je ovšem také připraven na vyhození výjimky ApplicationLogicException, které by v tomto případě znamenalo neočekávanou interní chybu aplikace. Pokud však prezenter úspěšně obdrží párování, připraví získaná data pro svůj pohled a přikáže mu ho zobrazit. Tím je role prezenteru ukončena a pohled je nyní zodpovědný za dané zobrazení. Poté již uživatel aplikace vidí provedené párování v GUI.

Nyní popíši co se odehrává v business vrstvě. Jak již bylo zmíněno, prezentační vrstva volá metodu deklarovanou v rozhraní PlayerRound. Pod tímto rozhraní se mohou nacházet různé implementace. V současné době je pouze jediná implementace tohoto rozhraní, jelikož plně dostačuje. V budoucnu se však může uvažovat o změně této implementace. Dané rozhraní je implementováno třídou StdPlayerRound. Její instance obdrží požadavek na párování pomocí volání metody doPairing. Provede nezbytné kontroly tohoto požadavku a požádá párovací engine o provedení párování. Pokud párování proběhne bez problému, zaktualizuje svůj stav a požádá instanci třídy StdPlayerTournament o uložení svého stavu do perzistentního prostředí. Jelikož je její součástí, musí instance třídy StdPlayerTournament také aktualizovat svůj stav. Následně požádá rozhraní TournamentManager o perzistentní uložení. Rozhraní TournamentManager je implementováno třídou StdTournamentManager a představuje jedináčka. Je to z toho důvodu, že je potřeba pouze jediná instance této třídy. Tato třída požádá rozhraní EntityDao o perzistetní uložení. Toto rozhraní patří do datové vrstvy. Implementace tohoto rozhraní StdEntityDao pak pomocí třídy EntityManager uloží entitu PlayerTournamentEntity do databáze.

Nyní se vrátím k párovacím enginu. Ten obdrží požadavek na párování a provede ho podle zvoleného turnajového systému. Algoritmus se liší podle zvoleného turnajového typu, systému a enginu. Popíši nyní algoritmus pro jednotlivé systémy, které aplikace podporuje. Opět zanedbám turnajový typ a popíši jen turnaj jednotlivců, jelikož pro turnaj družstev je to obdobné.

4.3.1 Holandská varianta švýcarského systému

Tento systém je implementován externím enginem JaVaFo. Vlastní algoritmus je tedy skryt. Mým úkolem bylo pouze připravit vstupy a zpracovat jeho výstupy. Tento engine podporuje více vstupů. Byl vybrán TRF formát.

4.3.2 Baku akcelerace holandské varianty švýcarského systém

Tento systém je opět implementován pomocí enginu JaVaFo. Platí vše, co bylo zmíněno v předchozí části. Rozdíl byl pouze v zavolání jiného kódu v dané metodě. Pomocí tohoto kódu totiž rozliší o jakou variantu systému se jedná. Pro klasickou holandskou variantu je definovaným kódem 1000, zatímco pro její Baku akceleraci je definovaným kódem 1001.

4.3.3 Jednokolový systém každý s každým

Tento systém je implementován vlastním enginem aplikace. Jako algoritmus byly zvoleny Bergerovy tabulky, které byly popsány v kapitole 1. Pokud je lichý počet hráčů, je přidán pro párovací účely hráč, který představuje bye. Na následující ukázce kódu můžeme vidět metodu gerRoundRobinMatches, zodpovědnou za párování. Tato metoda očekává seznam hráčů se sudým počtem, čehož se docílí již zmíněným postupem. Dále pak očekává skutečný počet hráčů, číslo kola a parametr, který je pro jednokolovou verzi každý s každým vždy false. V metodě se volá metoda roundRobinAddition, která je zodpovědná za vypočtení indexu hráče v daném seznamu hráčů.

Metoda pro generování párování v systému každý s každým

```
/**
* Generates a pairing
* Berger tables method is used.
* @param players all players in the tournament, must be even
* @param playersCount actual players count
* @param roundNum number of the round
* @param secondHalfDoubleRoundRobin if pairing is for second half,
    for single round robin it is always false
* @return list of matches (ie. the pairing)
*/
private List<MatchEntity> getRoundRobinMatches(List<PlayerEntity>
    players, int playersCount, int roundNum, boolean
    secondHalfDoubleRoundRobin) {
List<MatchEntity> matches = new ArrayList<>();
for (int i = 0; i < players.size() / 2; i++) {
PlayerEntity whitePlayer;
PlayerEntity blackPlayer;
if (i == 0) {
if (roundNum % 2 == 1) {
whitePlayer = players.get(roundRobinAddition(players.size(), 0,
    roundNum));
blackPlayer = playersCount % 2 == 1 ? null :
    players.get(players.size() - 1);
} else {
whitePlayer = playersCount % 2 == 1 ? null :
    players.get(players.size() - 1);
```

```
blackPlayer = players.get(roundRobinAddition(players.size(), 0,
    roundNum));
}
} else {
whitePlayer = players.get(roundRobinAddition(players.size(), i,
    roundNum));
blackPlayer = players.get(roundRobinAddition(players.size(),
    players.size() - i - 1, roundNum));
}
MatchEntity match = new MatchEntity(whitePlayer, blackPlayer);
if (secondHalfDoubleRoundRobin && i != 0) {
PlayerEntity temp = match.getWhite();
match.setWhite(match.getBlack());
match.setBlack(temp);
}
matches.add(match);
}

return matches;
}
```

4.3.4 Dvoukolový systém každý s každým

Tento systém je opět implementován vlastním enginem aplikace. Algoritmus je v podstatě stejný jako pro jednokolovou verzi. Dokonce je využita stejná metoda. Hlavní rozdíl je pouze v tom, že při jejím volání záleží na tom, zda je turnaj v první nebo druhé polovině. Pokud v první, tak volání je totožné jako pro jednokolovou verzi a parametr secondHalfDoubleRoundRobin je false. Naopak v druhé polovině turnaje je true. Co se týče samotného algoritmu, tak se liší pouze tím, že pokud je turnaj v druhé polovině, tak se prohodí barvy hráčů každé partie. Neboli hráč, který by měl v první polovině turnaje v dané partii bílé figury, bude mít černé a naopak.

4.3.5 Vyřazovací systém

Tento systém je také implementován vlastním enginem aplikace. Při jeho implementaci jsem neuvažoval, kdo s kým má být párován. Dané párování je pseudonáhodné. Na následující ukázce kódu můžeme vidět metodu doEliminationPairing, která je zodpovědná za vlastní párování. Metoda je relativně jednoduchá. Vytvoří mělkou kopii seznamu hráčů, který obdrží jako parametr. Poté zkontroluje, zda není v dané kopii seznamu lichý počet hráčů. Pokud ano, pak do ní jednoduše přidá další položku. Přesněji řečeno hodnotu null. Hodnota null totiž představuje bye. Dále pak v cyklu pseudonáhodně vybírá dva hráče a odstraní je z kopie seznamu a vytvoří jejich zápas. Tento zápas jednoduše přidá do seznamu zápasů. Po skončení cyklu jsou již všichni hráči

spárováni a metoda vrací seznam zápasů. Opět zde seznam zápasů představuje párování daného kola.

Metoda pro generování párování ve vyřazovacím systému

```
   /**
* Generates a pairing
* @param tour player tournament
* @param playerList list of players to pairing
* @return list of matches (ie. the pairing)
*/
private List<MatchEntity> doEliminationPairing(PlayerTournament tour,
    List<PlayerEntity> playerList) {
List<PlayerEntity> players = new ArrayList<>(playerList);
if (players.size() % 2 == 1) {
players.add(null);
}

List<MatchEntity> matches = new ArrayList<>();
Random r = new Random();
final int iterations = players.size() / 2;
for (int i = 0; i < iterations; i++) {
int index = r.nextInt(players.size());
PlayerEntity white = players.get(index);
players.remove(index);

index = r.nextInt(players.size());
PlayerEntity black = players.get(index);
players.remove(index);

MatchEntity match = new MatchEntity(white, black);
matches.add(match);
}

return matches;
}
```

4.4 Implementace pomocných hodnocení

Pomocné hodnocení jsou potomky abstraktní třídy RankingSystem. Tato třída deklaruje dvě abstraktní metody, které každé pomocné hodnocení musí implementovat. Jedná se výpočet hodnoty pomocného hodnocení pro hráče či družstva daného turnaje. Pomocí těchto dvou metod pak definuje porovnání dvou hráčů nebo družstev daného turnaje. Implementace jednotlivých pomocných hodnocení bylo provedeno pomocí informací získaných z kapitoly 1. Pro jejich implementaci jsem nepoužil žádný engine, vše jsem implemento-

val sám. Na následující ukázce kódu, můžeme vidět příklad naimplementovaného pomocného hodnocení pro turnaj jednotlivců. Jedná se o Buchholzův systém. Nejprve získá od turnaje body všech hráčů a seznam soupeřů hráče, pro kterého se hodnota tohoto pomocného hodnocení vypočítává. Poté jednoduše sečte body všech jeho soupeřů a tento součet vrátí.

Metoda pro výpočet hodnoty pomocného hodnocení Buchholz

```
   @Override
public int getValue(PlayerTournament tour, PlayerEntity player)
    throws ApplicationLogicException {
Map<Integer, Integer> playersPoints = tour.getPlayersPoints();
List<PlayerEntity> opponents = tour.getOpponents(player);
int opponentPoints = 0;
for (PlayerEntity i : opponents) {
opponentPoints += playersPoints.get(i.getId());
}
return opponentPoints;
}
```

4.5 Logování

Logování bylo implementováno v datové vrstvě pomocí třídy Logger. Třída obsahuje pouze jednu statickou metodu writeToLog. Tato metoda přijímá dva parametry. První je výjimka a druhý je zpráva, která má být uložena do logovacího souboru. Výjimka slouží pro uložení záznamu zásobníku k dané zprávě. Hodnota null je pro oba parametry povolena. Byly také naimplementovány výjimky, po jejichž vytvoření dojde automaticky k zápisu do logovacího souboru. Děje se tak právě prostřednictvím třídy Logger. Jedná se o výjimky BaseException a UnexpectedException. BaseException je abstraktní třída a její potomci představují vlastní výjimky. Jedná se o ApplicationLogicException, FileException, InvalidDataException a NoEntityFoundException. Jelikož výjimka BaseException dědí od třídy Exception, musí se ona i všechny její potomci v aplikaci zachytávat a ošetřit. Naproti tomu výjimka UnexpectedException je potomkem třídy RuntimeException a tedy se zachytávat nutně nemusí.

4.6 Instalace aplikace

Instalace je velmi jednoduchá. Aplikace je představována pouze jedním kořenovým adresářem, který obsahuje spouštěcí soubor ChessTournamentManager.jar a pomocné adresáře db, logs, uploads a temp. Aplikace se může distribuovat například zabalena do libovolného archivu. Na cílovým počítači je

pak potřeba pouze extrahovat kořenový adresář aplikace na libovolné místo. Je však nutné, aby všechny části aplikace měly práva číst a zapisovat.

Nyní popíši jednotlivé podadresáře. Podadresář db je využíván pro uložení databázového souboru database.mv.db. Pokud tento soubor neexistuje, je po spuštění aplikace automaticky vytvořen. Tento soubor představuje celou databázi a tedy všechna data aplikace. Může se jednoduše zálohovat pouhým překopírováním. Pokud uživatel například smaže omylem turnaj, může pak překopírovat zálohu do adresáře db. Musí se však jmenovat stejně. Pro logovací účely může aplikace vygenerovat do toho adresáře logovací soubor database.mv.db. Dalším podadresářem je logs, který obsahuje soubor log.txt. Do něj aplikace ukládá kromě interních chyb také různá upozornění, které pak mohou pomoci například při zjišťování, proč se daná operace nezdařila. Pokud neexistuje, bude aplikací v případě potřeby automaticky vytvořen. Dalším podadresářem je uploads. Zde aplikace ukládá všechny soubory, které byly připojeny k určitému kolu. Pokud tedy uživatel smaže některý soubor, již ho nebude moci z aplikace stáhnout. Také zde platí, že lze tyto soubory zálohovat a také, že při vracení zálohy se musí soubory jmenovat stejně jako předtím. Posledním podadresářem je temp. Slouží pro ukládání dočasných souborů. Pokud aplikace není spuštěna mohou se tyto soubory kdykoli smazat.

4.7 Dokumentace aplikace a zveřejnění otevřeného kódu

Pro zdokumentování zdrojových kódů aplikace jsem využil Javadoc. Po zdokumentování nejdůležitějších metod a atributů, jsem pak vygeneroval dokumentaci. Tu můžeme nalézt v přiloženém DVD v adresáři doc. Jelikož má být aplikace open-source, vytvořil jsem na portálu github.com repozitář pro zdrojové kódy. Repozitář jsem však zatím ponechal soukromý, abych předešel potížím. Například by někdo mohl můj kód použít a já bych pak musel vysvětlovat, proč používám stejný kód jako daná osoba. Repozitář je plně připraven a po obhajově této práce ho nastavím jako veřejný. Jeho url je https://github.com/pahorjir/Chess_Tournament_Manager. Tato url je bez přihlášení do mého účtu na portálu github.com z výše zmíněných důvodů zatím nedostupná.

4.8 Porovnání implementovaného řešení s existujícími metodami

Aplikaci jsem naimplementoval tak, aby byla intuitivní a snadno použitelná. Neobsahuje tedy tolik funkcionalit jako například aplikace Swiss-Manager nebo SwissSys. Naproti tomu má však oproti nim použitou moderní technologii pro grafické uživatelské rozhraní. Počtem funkcionalit je srovnatelná

s open-source aplikací JavaPairing. Na rozdíl od ní má však lépe navrženou architekuru a je dle mého názoru obecně lépe naimplementovaná. Díky zvolené architektuře lze snadno vyměnit uživatelské rozhraní, databázi, přidat párovací enginy, pomocná hodnocení atd., což nemusí být pro JavaPairing jednoduché. Oproti aplikaci stChess má mnohem více funkcionalit a neobsahuje reklamu a oproti aplikaci Vega je kompletně zdarma a open-source.

Kapitola 5

Testování

V této kapitole budu popisovat, jak jsem otestoval naimplementovanou aplikaci. Jednalo se o manuální testování GUI a používání jednotkových testů. Oba způsoby jsem však používal již při vývoji.

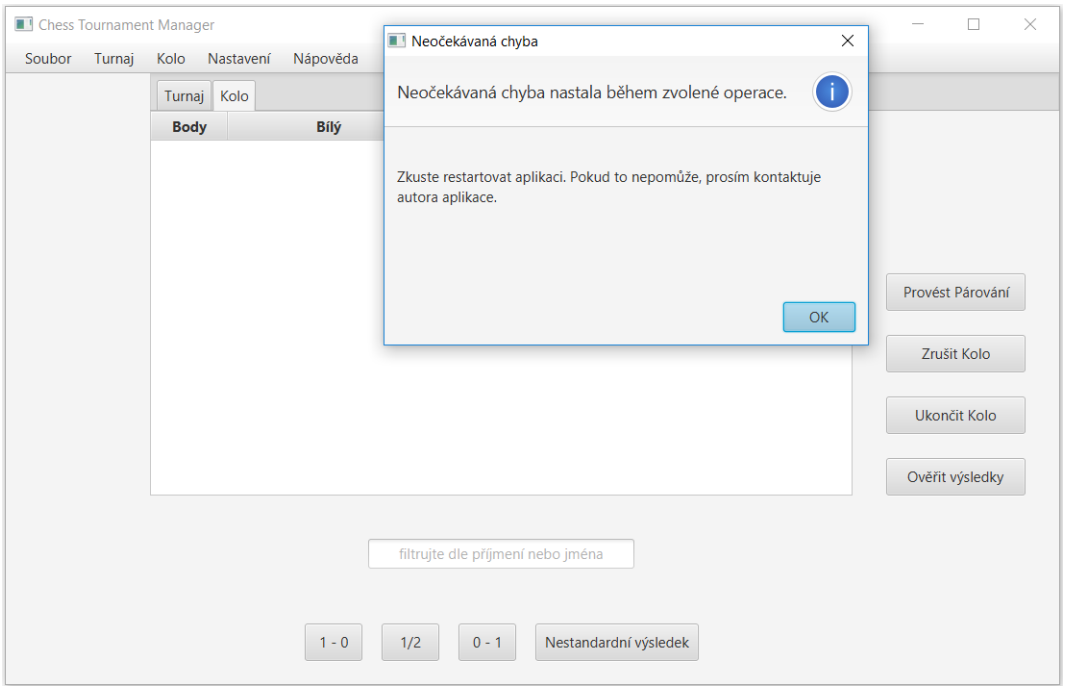
5.1 Manuální testování GUI

Manuální testování jsem používal především na ověření, zda jednotlivé GUI elementy správně fungují. Například jestli stisknutí tlačítka vykoná danou akci. Také jsem to používal pro testování, zda se s aplikací pracuje pohodlně. Manuálním testováním většinou nešlo o testování správného fungování logiky aplikace.

Chybu jsem během implementace většinou rozpoznal tak, že jsem viděl, že aplikace neprovedla danou akci nebo jí provedla špatně. Pomáhali také chybové hlášky a upozornění ve vývojovém prostředí. Později pomáhaly také naimplementované chybové dialogy a logování. Na obrázku 5.1 můžeme vidět chybový dialog, který oznámil chybu při vykonávání dané akce. Konkrétně se zde jednalo o chybu při párování hráčů v systému každý s každým v turnaji jednotlivců. Kdykoli po upozornění jsem se mohl podívat do logovacího souboru, co danou chybu způsobilo.

5.2 Jednotkové testy

Pomocí jednotkových testů jsem testoval, zda logika aplikace funguje správně. Jednotkové testy byly také velmi užitečné, když jsem provedl změny v kódu. Po jejich spuštění jsem zjistil, zda jsem nerozbil nějakou funkčnost aplikace. Díky zvolené architektuře lze kromě business a datové vrstvy také testovat prezentery z prezentační vrstvy. To je způsobeno tím, že prezentery nepoužívají grafické elementy. Ty jsou používány pouze pohledy. Nejprve ukáži ukázku jednotkového testu pro třídu StandardPlayerTournament. Pomocí nich jsem



Obrázek 5.1: Chybový dialog

jí odladil. Poté ukáži ukázku jednotkového testu pro třídu PlayerTournamentPrezenter. Oba jednotkové testy budou testovat stejnou věc a to přidávání, supendování a zrušení suspendace hráčů. Poté co jsem odladil třídu StandardPlayerTournament, jsem mohl považovat za business logiku této problematiky v pořádku. Pak již zbývalo pouze odladit prezenter PlayerTournamentPrezenter zodpovědný za volání správných metod třídy StandardPlayerTournament. Testování, zda se nakonec přidaný hráč nebo jeho změna stavu projevila v grafickém uživatelském rozhraní jsem otestoval pomocí manuálního testování.

Metoda pro testování business logiky práce s hráči

```
   @Test
void testPlayers() throws Exception {
assertEquals(0, tour.getAllPlayers().size());
tour.addPairablePlayer(new PlayerEntity());
assertEquals(1, tour.getAllPlayers().size());

PlayerEntity player = tour.getAllPlayers().get(0);
final int tempId = player.getId();
assertThrows(ApplicationLogicException.class, () ->
    tour.unSuspendPlayer(tempId));
tour.suspendPlayer(tempId);
assertEquals(Status.SUSPENDED,
```

```
    tour.getAllPlayers().get(0).getStatus());
assertThrows(ApplicationLogicException.class, () ->
    tour.suspendPlayer(tempId));
tour.unSuspendPlayer(tempId);
assertEquals(Status.PAIRABLE,
    tour.getAllPlayers().get(0).getStatus());
}
}
```

Metoda pro testování flow logiky práce s hráči

```
@Test
void testPlayers() throws Exception {
assertEquals(0, tour.getAllPlayers().size());
tour.addPairablePlayer(new PlayerEntity());
tour.addPairablePlayer(new PlayerEntity());
tour.addPairablePlayer(new PlayerEntity());
tour.addPairablePlayer(new PlayerEntity());

assertEquals(4, tour.getAllPlayers().size());
presenter.onSuspendPlayers(new ArrayList<>(tour.getAllPlayers()));
assertEquals(4, tour.getAllPlayers().size());
assertEquals(0, tour.getPairablePlayers().size());
assertEquals(4, tour.getSuspendedPlayers().size());
presenter.onUnsuspendPlayers(new ArrayList<>(tour.getAllPlayers()));
assertEquals(4, tour.getAllPlayers().size());
assertEquals(4, tour.getPairablePlayers().size());
assertEquals(0, tour.getSuspendedPlayers().size());
}
```

Závěr

Cílem práce bylo vytvořit multiplatformní a open-source aplikaci, která bude sloužit rozhodčím a ředitelům šachových turnajů. Tento cíl byl splněn. Nejprve jsem nastudoval oficiální pravidla pro šachové turnaje ze zdrojů Mezinárodní šachové federace. Ta nejdůležitější jsem pak popsal v kapitole 1. Byla popsána pouze ta, které by mohla mít význam pro řízení šachových turnajů. Poté jsem v kapitole 2 provedl podrobnou rešerši existujících aplikací pro řízení šachových turnajů. Upřednostnil jsem aplikace, které jsou oficiálně schválené FIDE. Schválené totiž musely projít relativně složitým schvalovacím procesem, takže je u nich zaručena funkčnost. U všech popsaných aplikací jsem popsal jejich výhody a nevýhody. Následně jsem stanovil požadavky na aplikaci a provedl jejich analýzu. Provedl jsem také analýzu problémové domény. V kapitole 3 jsem pak navrhl architekturu aplikace. Zvolil jsem relaxovanou třívrstvou architekturu. Pro prezentační vrstvu jsem použil vzor MVP. Párování kol a výpočet pomocných hodnocení jsem umístil do business vrstvy, jelikož jsou součástí business logiky aplikace. Pro švýcarský systém a jeho Baku akceleraci jsem navrhl použití enginu JaVaFo. Aplikace byla naimplementována jako desktopová v jazyce Java. Z turnajových systému byl naimplementován jednokolový a dvoukolový systém každý s každým, švýcarský systém na libovolný počet kol včetně jeho Baku akcelerace a vyřazovací systém. Všechny tyto systémy byly implementovány jak pro turnaj jednotlivců tak pro turnaj družstev. Pro turnaj družstev bylo navíc umožněno zvolení hlavního bodového hodnocení. Aplikace je díky implementaci v jazyce Java multiplatformní. Po její implementaci jsem zhodnotil výhody implementovaného řešení vůči existujícím metodám. Aplikace byla zdokumentována pomocí Javadoc a pro zveřejnění jejích zdrojových kódů, byl vytvořen repozitář na portálu github.com. Repozitář bude po ohajobě práce nastaven jako veřejný. Dokumentaci lze nalézt v přiloženém DVD v adresáři *doc* a repozitář bude po obhajobě veřejně přístupný z https://github.com/pahorjir/Chess_Tournament_Manager. Nakonec byla aplikace otestována pomocí manuálního testování a jednotkových

testů. Oba postupy však byly používány již při vývoji. Závěrem mohu prohlásit, že hlavní cíl a všechny dílčí cíle práce byly splněny.

Literatura

[1] FIDE - World Chess Federation [online]. [cit. 2019-06-02]. Dostupné z: http://www.fide.com/fide.html

[2] Czech Republic / Šachový Svaz České Republiky [online]. [cit. 2019-06-02]. Dostupné z: https://ratings.fide.com/fide_directory.phtml?country=CZE&list=825

[3] General Regulations for Competitions [online]. [cit. 2019-06-02]. Dostupné z: https://www.fide.com/FIDE/handbook/Competition_Rules.pdf

[4] Fide Laws of Chess taking effect from 1 January 2018 [online]. [cit. 2019-06-02]. Dostupné z: https://www.fide.com/fide/handbook.html?id=208&view=article

[5] Recommendations for Organization of Top-level Tournaments [online]. [cit. 2019-06-02]. Dostupné z: https://www.fide.com/component/handbook/?id=15&view=category

[6] Standards of Chess Equipment,venue for FIDE Tournaments, rate of playand tie-break regulations [online]. [cit. 2019-06-02]. Dostupné z: https://www.fide.com/FIDE/handbook/Standards_of_Chess_Equipment_and_tournament_venue.pdf

[7] General Regulations for Competitions. Annex 1: Details of Berger Table [online]. [cit. 2019-06-02]. Dostupné z: https://www.fide.com/component/handbook/?id=184&view=article

[8] Regulations for the FIDE World Chess Cup 2017 [online]. [cit. 2019-06-02]. Dostupné z: http://www.fide.com/FIDE/handbook/WorldCup2017Regulations.pdf

[9] C.04.4 Other FIDE-approved Pairing Systems [online]. [cit. 2019-06-02]. Dostupné z: https://www.fide.com/component/handbook/?id=203&view=article

[10] C.04.1 Basic rules for Swiss Systems [online]. [cit. 2019-06-02]. Dostupné z: https://www.fide.com/component/handbook/?id=83&view=article

[11] C.04.2 General handling rules for Swiss Tournaments [online]. [cit. 2019-06-02]. Dostupné z: https://www.fide.com/component/handbook/?id=84&view=article

[12] C.04.5 FIDE-approved Accelerated Systems [online]. [cit. 2019-06-02]. Dostupné z: https://www.fide.com/component/handbook/?id=204&view=article

[13] C.04.A Appendix: Endorsement of a software program [online]. [cit. 2019-06-02]. Dostupné z: https://www.fide.com/component/handbook/?id=206&view=article

[14] Verification Checklist [online]. [online]. [cit. 2019-06-02]. Dostupné z: https://www.fide.com/FIDE/handbook/C04Annex4_VCL17.pdf

[15] Endorsed Tournament Managers [online]. [cit. 2019-06-02]. Dostupné z: https://www.fide.com/FIDE/handbook/C04Annex3_FEP16.pdf

[16] JaVaFo, a pairing enginefor the FIDE (Dutch) System [online]. [cit. 2019-06-02]. Dostupné z: http://www.rrweb.org/javafo/JaVaFo.htm

[17] Herzog, D. I. H.: Swiss-Manager [online]. [cit. 2019-06-02]. Dostupné z: http://swiss-manager.at

[18] Tournament Database of Chess Results [online]. [cit. 2019-06-02]. Dostupné z: http://chess-results.com/TurnierSuche.aspx?lan=1

[19] Chess Tournament Administration [online]. [cit. 2019-06-02]. Dostupné z: http://www.vegachess.com/ns/

[20] Welcome to SwissSys Software [online]. [cit. 2019-06-02]. Dostupné z: http://www.swisssys.com

[21] JavaPairing [online]. [cit. 2019-06-02]. Dostupné z: https://sourceforge.net/projects/javapairing/?source=typ_redirect

[22] Menne, S.: stChess [online]. [cit. 2019-06-02]. Dostupné z: http://www.stchess.de

[23] Hommel, S.: *Implementing JavaFX Best Practices [online]*. Oracle Corporation, [cit. 2019-06-02]. Dostupné z: https://docs.oracle.com/javafx/2/best_practices/jfxpub-best_practices.htm

Příloha **A**

Seznam použitých zkratek

CRUD Create, read, update and delete

DAO Data access object

FIDE Fédération Internationale des Échecs

GPLv3 GNU General Public License version 3

GUI Graphical user interface

HTML Hypertext markup language

JPA Java persistence API

MVC Model–view–controller

MVP Model–view–presenter

OOP Object-oriented programming

TRF Tournament report file

UML Unified modeling language

XML Extensible markup language

Příloha **B**

Obsah přiloženého DVD

- `readme.txt` ..... stručný popis obsahu DVD
- `exe` ..... adresář se spustitelnou formou implementace
- `src`
  - `impl` ..... zdrojové kódy implementace
  - `thesis` ..... zdrojová forma práce ve formátu LaTeX
- `doc` ..... dokumentace zdrojových kódů
- `text` ..... text práce
  - `thesis.pdf` ..... text práce ve formátu PDF