Západočeská univerzita v Plzni

Úvod do počítačových sítí

KIV/UPS

Dokumentace semestrální práce - Šachy

Jakub Vaněk
A16B0160P va-
nekjak@students.zcu.cz

13. ledna 2019

Obsah

1 Zadání

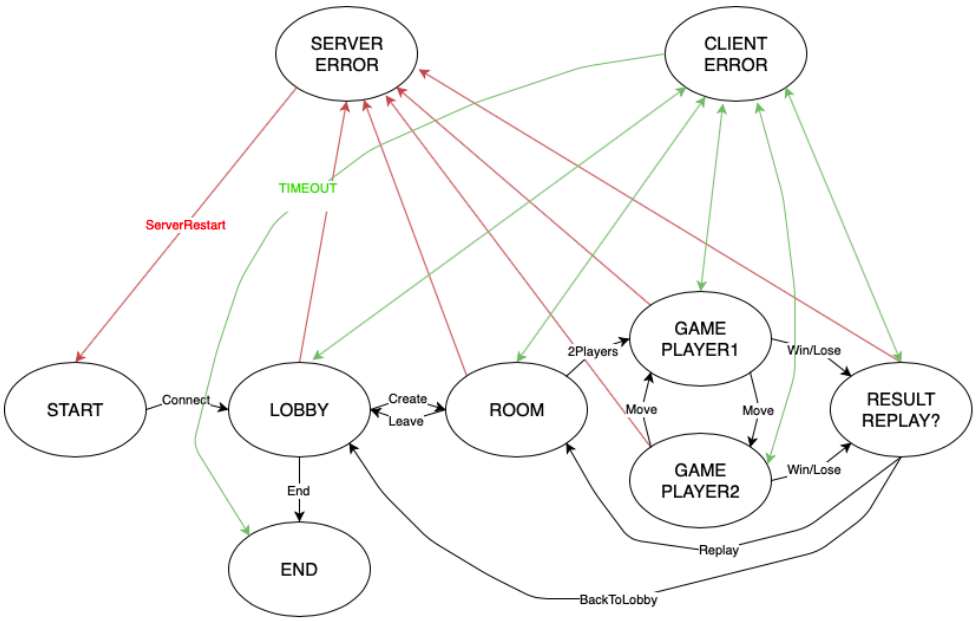
Vytvořte program realizující vybranou hru. Vytvořte server, který bude obsluhovat více hráčů i her současně a bude schopen uložit stav hry. Klient bude po opětovném přihlášení pokračovat tam, kde přestal. Přenosový protokol je TCP. Jazyk použitý pro server: C++. Jazyk použitý pro klienta: Java. Vybranou hrou je dáma.

2 Programátorská dokumentace

V této sekci bude uveden diagram, dle kterého byla aplikace vytvořena, bude popsána funkčnost serveru a klienta a popis jejich implementace.

2.1 Stavový diagram

Toto je stavový diagram naprogramované aplikace. Elipsy označují stavy a šipky označují možné přechody mezi nimi. V podkapitole serveru a klienta bude uveden seznam všech zpráv, které přechody mezi stavy obstarávají.



Obrázek 1: Stavový diagram

2.2 Protokol

Typ spojení užié v aplikaci je TCP. TCP je protokol, který naváže spojení mezi dvěma body a toto spojení drží, dokud není spojení uzavřeno. Vytváří tak komunikační linku, po které spolu obě strany komunikují.

Protokol použitý v této aplikaci je textový a nešifrovaný. Zprávy jsou zapsány velkými písmeny a mohou mít jednu i více částí. Jednotlivé části zpráv jsou od sebe odděleny středníkem. Zprávy, které odesílá server klientovi jsou navíc zakončeny novým řádkem. První část značí typ zprávy a další části jsou parametry, které jsou k výkonu instrukcí nutné.

V sekcích serveru a klienta budou výčet veškerých zpráv, které si klient a server mezi sebou posílají.

2.3 Server

Server je naprogramován v jazyce C++. Jeho funkce je ovládání aplikace. Jeho datové struktury drží a pracují s klienty, kteří se k němu připojí. Na základě zpráv, které přijímá od klienta s těmito strukturami pracuje. V dalších částech budou popsány hlavní datové struktury, se kterými server pracuje.

2.3.1 Server

Struktura server slouží k navázání spojení. Chyby při navázání spojení jsou řešeny ošetřením funkcí *bind* a *listen*. Pokud tyto funkce proběhnou vpořádku, server začne poslouchat na přiřazeném portu a adrese.

Příchozí zprávy jsou řešeny pomocí funkce select. Tato funkce obsahuje sadu file descriptorů neboli socketů a čeká až bude některý z nich "připravený"pro komunikaci. Pokud některý z nich je, vybere ho a na tomto socketu je pomocí funkce *ioctl* načten buffer obsahující zprávu.

2.3.2 MessageManager

Po přečtení zprávy na straně serveru je zpráva poslána do *MessageManageru*. Tato struktura slouží k dekódování zprávy a její validaci. Zprávy jsou rozděleny na základě delimetru a jejich části jsou zpracovány.

2.3.3 GameManager

Po přečtení zprávy MessageManager dává instrukce struktuře *GameManager*. Tato struktura v sobě drží sadu připojených hráčů, sadu odpojených hráčů a

sadu vytvořených her v podobě hashmap. Na základě instrukcí MessageManageru tato struktura s těmito sadami pracuje. Připojuje jednotlivé hráče, odpojuje je, přidává do sad, do her a celkově je ovládá.

2.3.4 Game

Struktura *Game* je využívána ve chvíli, kdy dva hráči hrají. Je to struktura, která hlídá pravidla hry. Jednotlivé figurky mají různá pravidla na jejichž základě se pohybují.

2.3.5 ConnectionThread

ConnectionThread není strukturou, ale vláknem, které jednou za vteřinu kontroluje, zda je hráč stále připojen. Pokud bylo připojení přerušeno, nebo se s ním cokoliv stalo, dává o tom zprávu GameManageru, který s klientem nakládá podle jeho aktuálního stavu.

2.3.6 Ostatní struktury

Server obsahuje spoustu dalších struktur, které dopomáhají celkové funkci serveru. Mezi důležité patří *ClientState*, což je výčtový typ všech stavů, kterých může klient nabýt. Dále mezi nimi najdeme struktury *Client* a *Match*, které si drží jednotlivé údaje o hráčích a hrách. Dalšími strukturami jsou *Piece*, *Field*, *Color* a *Type*.

2.3.7 Zprávy

Zde lze najíst kompletní výčet zpráv, které server rozpoznává a popis jeho reakce.

- **EXIT;** - validní ukončení klienta. Klienta a záznamy o něm vymaže ze serveru
- **PING;** - kontrola ztráty spojení mezi serverem a klientem
- **CONNECT;OK;** - klient žádá o připojení, server ho zanese do svých struktur
- **CONNECT;FAIL;** - klientovi se nezdařilo připojení, server uzavře socket
- **CREATE_MATCH;** - klient žádá o vytvoření hry, server vytvoří hru

- **FIND_MATCH;** - server odešle seznam her, kteŕe jsou k připojení k dispozici
- **MOVE;sRow;sCol;dRow;dCol** - server zkontroluje pravidla o pohybu figurky ze souřadnic {sRow,sCol} na souřadnice {dCol,dRow}
- **REPLAY_Y;** - hráč žádá o opětovné hraní, pokud zažádají oba klienti, je hra spuštěna znovu
- **REPLAY_N;** - hráč již nechce znovu hrát a je vrácen do Lobby, server vrátí do Lobby i druhého hráče
- **STOP_WAIT;** - hráč se navrátí do lobby, pokud nechce čekat na dotaz o opětovném spuštění hry druhého klienta
- **YIELD;** hráč se vzdal a ukončil tak hru

2.4 Klient

Klientská aplikace slouží k uživatelskému ovládání. Aplikace je naprogramována v Javě a k tvorbě uživatelského rozhraní je využita *JavaFX*. Na základě GUI odesílá serveru zprávy a podle jeho reakcí GUI upravuje a umožňuje tak uživateli přívětivé ovládání hry. V dalších částech budou popsány hlavní třídy obsluhující klientskou aplikaci.

2.4.1 ReaderThread

ReaderThread je vlákno, které zpracovává všechny zprávy, které dostane od serveru a říká GUI, jak se má zachovat. Čtení zpráv obstarává *BufferedReader*, který čte z InputStreamu na připojeném socketu. Zprávy čte na základě stavu, ve kterém se klient nachází.

2.4.2 Chess

Tato třída je hlavní třídou aplikace. Při zapnutí nastaví veškeré potřebné proměnné klienta, stageController a naváže spojení se serverem.

2.4.3 StageController

Třída *StageController* je hlavní třídou, která se stará o veškeré GUI. Uživateli zobrazuje všechna okna a ovládací prvky aplikace.

2.4.4 ConnectionThread

Vlákno *ConnectionThread* slouží ke stejnému účelu jako struktura ConnectionThread na serveru. Hlídá jestli je spojení stále navázáno. Pokud není, zareaguje na klientu.

2.4.5 Ostatní třídy

Klientská aplikace obsahuje spoustu dalších tříd, které dopomáhají funkčnosti aplikace. Patří mezi ně výčtové typy, konstanty a další pomocné třídy, které reprezentují hru.

2.4.6 Zprávy

Zde lze najíst kompletní výčet zpráv, které klient rozpoznává a popis jeho reakce.

- **PING;** - kontrola ztráty spojení mezi serverem a klientem
- **RECONNECT;X;** - klient se znovu připojil, X značí stav, do kterého se má dostat
- **CONNECT;OK;** - klient se připojil
- **CONNECT;FAIL;** - klientovi se nezdařilo připojení
- **CREATE_MATCH;OK;** - hra byla úspěšně vytvořena
- **FOUND_MATCH;X-....-Y** - seznam her, kde X-...-Y označují hry, ke kterým se lze připojit
- **JOIN_MATCH;OK;** - připojení do hry bylo úspěšné
- **JOIN_MATCH;FAIL;** - do vybrané hry se nelze připojit
- **PAUSE;** - protihráč ztratil spojení a je na něj třeba počkat
- **BACK;** protihráč opětovně navázal spojení a hraje dál
- **MOVE;OK;sRow;sCol;dRow;dCol** pohyb figurky souhlasí s pravidly a je posunuta ze souřadnis {sRow,sCol} na souřadnice {dRow,dCol}
- **MOVE;FAIL;** pohyb figurky nesouhlasí s pravidly a je třeba hrát znovu
- **GAME;WIN;** klient zvítězil ve hře

- **GAME;LOST;** klient ve hře prohrál
- **YIELD;** protihráč se vzdal a opustil tedy hru
- **LEFT;** protihráč se odpojil od serveru
- **PIECE;X-...-Y** vykreslí figurku na souřadnice s parametry označené v části X-...-Y
- **REPLAY;YES;** oba hráči žádají o novou hru a je tedy vytvořena
- **REPLAY;NO;** protihráč nechce hrát znovu

2.5 Implementace

V této sekci bude popsáno chování obou aplikací v různých případech užití

2.5.1 Přihlášení

Klient zadá jméno, které je odesláno na server. Server klienta připojí a jeho instanci přidá do hashmapy připojených kientů. Současně je vytvořeno vlákno ConnectionThread, které každou 1 sekundu posílá zprávu ping a kontroluje tak, zda je klient stále připojen.

2.6 Vytvoření hry

Při vytvoření hry se na serveru vytvoří instance hry, která obsahuje dva odkazy na klienta. První odkaz bude odkazovat na klienta který hru vytvořil a jako ID této hry bude použito označení socketu hráče který ji vytvořil. Toto ID slouží jako klíč v hashmapě vytvořených her. Současně je všem klientům odeslána zpráva s novým seznamem dostupných her, které se mu zobrazí v okně lobby.

2.7 Připojení ke hře

Klient se může připojit k libovolné hře. Server zkontroluje, zda je hra volná a přidá odkaz na druhého hráče. V tuto chvíli hráči mohou začít hrát. Tato hra již není validní k připojení hráče a je tudíž odeslána nová zpráva všech dostupných her, která překreslí list dostupných her.

2.8 Hra

Klient kliknutím na hrací pole odešle zprávu serveru, která pohyb zkontroluje. Pokud je pohyb validní, odešle oboum hráčům zprávu na základě které se překreslí hrací pole. Pokud pohyb validní není, je hráč vyzván k novému pohybu figurky.

2.9 Výsledky

Po zabití krále je oboum hráčům odeslána zpráva GAME, kde druhý parametr této zprávy označuje, zda klient vyhrál nebo prohrál. Na základě této zprávy je jim zobrazena hláška o výsledcích hry a je jim nabídnuta možnost hru hrát znovu. Pokud budou chtít hrát znovu, server znovu nainicalizuje hru a hra začne od začátku. Pokud hrát nechtějí, jsou oba dva odpojeni od příslušné hry, jsou jim nastaveny správné parametry a hra je zničena a odebrána z hashmapy dostupných her.

2.10 Ukončení

Hráč může v jakémkoliv stavu hru validně ukončit křížkem. Tato akce vyvolá na serveru validní odpojení hráče. Server klienta vymaže ze své sady socketů, vymaže ho z paměti a oznámí odpojení ostatním uživatelům, které tento klient ovlivňoval.

2.11 Řízení chyb

V aplikaci může v jakémkoliv stavu nastat přerušení spojení z různých důvodů. Pokud toto přerušení nerozpozná sám select, který na základě nevalidních zpráv hráče odstřihne, odhalí přerušení spojení vlákna ConnectionThread na klientské i serverové straně. Toto vlákno čeká dvě minuty. Pokud se do dvou minut hráč znovu nepřipojí, jen na serverové straně vymazán z paměti a ostatním hráčům odeslána zpráva, která značí jeho odpojení. Pokud se do dvou minut hráč znovu připojí, je mu odeslána zpráva RECONNECT, kde druhý parametr zprávy je stav ve kterém se klient má nacházet. Na základě tohoto stavu jsou mu následně odeslány další zprávy, které zajistí návrat do stejného stavu, jako před odpojením. Pokud během hry z nějakého důvodu spadne server, je klientům odeslána zpráva a jejich aplikace validně uzavřena.

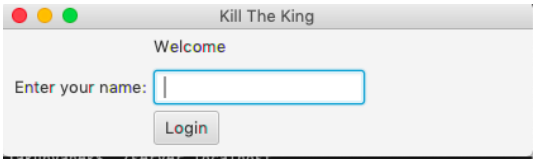
3 Uživatelská dokumentace

V této části bude popsána kompilace a spuštění jednotlivých částí aplikace.

3.1 Klient

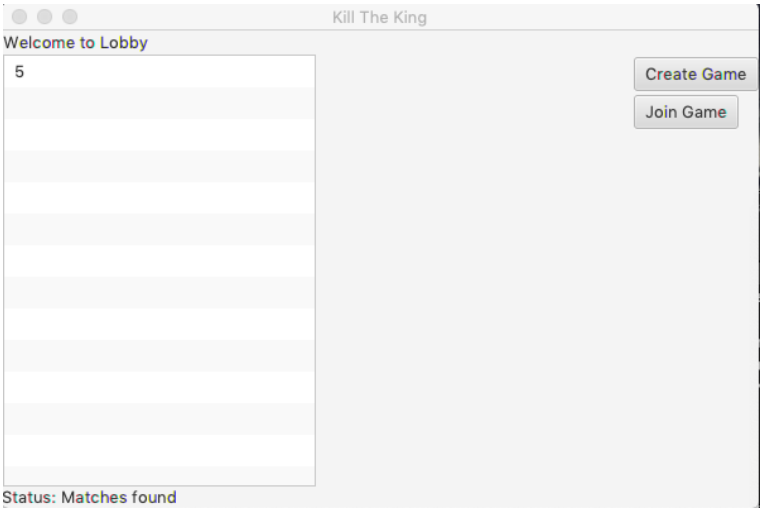
K překladu tříd do spustitelného souboru *.jar* je využit nástroj Maven. Maven je nástroj, který dokáže přeložit a spravovat projekty psané v Javě. Tento nástroj přeloží projekt pomocí příkazu *mvn clean install* ve složce projektu. Vytvořené soubory uloží do nově vytvořené složky target. Do této složky je potřeba nakopírovat složku images, která obsahuje obrázky figurek.

Po přeložení je klient spuštěn příkazem *java -jar [adresa] [port]* kde adresa je validní IPv4 adresa, na které bude aplikace poslouchat. Následně je zobrazeno přihlašovací okno. Do tohoto okna hráč zadá svou přezdívku a je připojen do hry.



Obrázek 2: Přihlašovací obrazovka

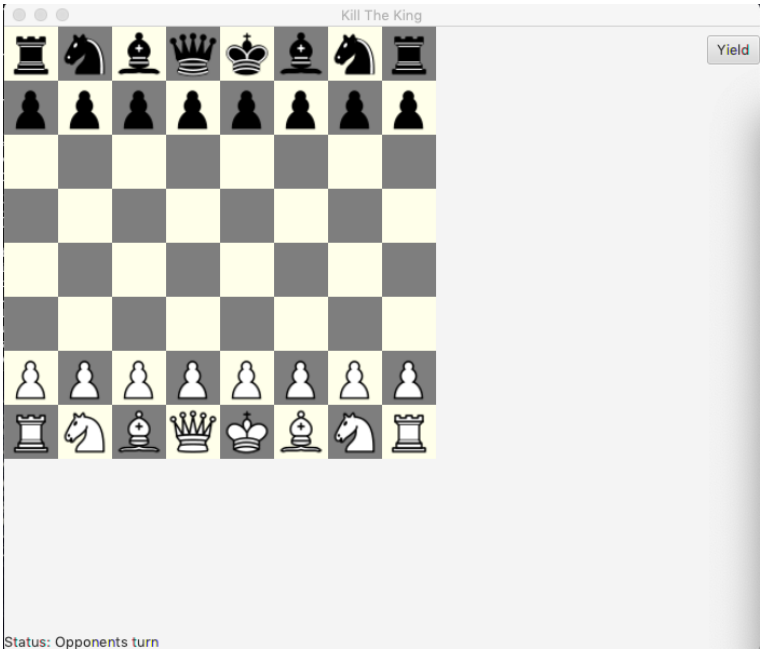
V tuto chvíli se ocitne v Lobby, kde může pomocí tlačítek vytvořit hru, nebo se připojit do již existující. Existující hry jsou k nalezení v levé části okna.



Obrázek 3: Hráč se může připojit nebo vytvořit hru

Obrázek 4: Hráč čeká na připojení protihráče

Po připojení do hry hráč hraje hru. Hra je ovládána myší. Prvním stiskem tlačítka myši hráč vybere figurku, kterou chce táhnout a druhým kliknutím určí místo, kam s ní chce pohnout.



Obrázek 5: Hráč je ve hře

Hra končí pokud jeden hráč druhému sebere krále.

Obrázek 6: Výsledky hry

Hráč se během hry také může vzdát stiskem tlačítka a tím je vrácen zpět do Lobby. Po skončení hry je hráči nabídnuto hru opakovat a nebo se vrátit zpět do Lobby. Hra může být v jakékoliv chvíli ukončena stisknutím křížku.

3.2 Server

K překladu souborů je využit makefile, který přeloží všechny soubory a vytvoří spustitelný soubor. Překlad je prováděn z terminálu ve složce se zdrojovými soubory aplikace příkazem *make*. Server je následně spuštěn příkazem *./server [adresa] [port]*, kde adresa je validní IPv4 adresa, na kterém bude server poslouchat. Port je možné vybrat jakýkoliv z rozsahu 0-65535. Server je uzavřen z příkazové řádky stiskem tlačítek *CTRL+C*.

4 Závěr

Aplikace splňuje veškeré požadavky semestrální práce.

Během jejího vypracování jsem zjistil, jak moc důležitá je analýza problému před začátkem implementace. Během implementace jsem se setkal s několika obtížemi, které se vyskytly hlavně kvůli špatnému návrhu aplikace. Při důkladnější analýze by tato situace nemusela nastat a implementace by se značně zkrátila.

Velká pozornost nebyla věnována tvorbě GUI, které nebylo hlavním cílem aplikace.