FAKULTA
APLIKOVANÝCH VĚD
ZÁPADOČESKÉ
UNIVERZITY
V PLZNI

Semestrální práce - hra pro více hráčů (Prší)

Úvod do počítačových sítí

Ondřej Drtina
A17B0202P
drtinao@students.zcu.cz

Obsah

1 Zadání

Cílem semestrální práce v rámci předmětu UPS bylo vytvořit síťovou hru pro více hráčů. Konkrétní hru si mohl student po konzultaci s příslušným cvičícím zvolit, avšak i přesto byla stanovena určitá pravidla pro její tvorbu. Vzhledem k tomu, že práce zahrnuje tvorbu herního serveru i klienta, lze požadavky rozdělit do dvou sekcí.

1.1 Požadavky na server

Herní server musí splňovat zejména následující požadavky:

- musí být napsán v nízkoúrovňovém programovacím jazyce (C / C++)
- je schopný obsluhy více klientů současně
- možnost znovu navázet komunikaci s klientem po výpadku spojení a umožnění pokračování ve hře

1.2 Požadavky na klienta

Herní klient musí splňovat následující požadavky:

- psán ve vysokoúrovňovém programovacím jazyce (Java / Unity)
- disponování grafickým uživatelským rozhraním
- kompatibilita s Windows i Linux
- možnost práce s herními místnostmi (lobby)

1.3 Společné požadavky

Některé z požadavků jsou shodné pro serverovou i klientskou část aplikace, například:

- komunikace pomocí protokolu TCP (bez zabezpečení)
- řešení výpadků komunikace (oznámení a patřičná reakce při odpojení klienta / serveru)

- možnost automatického překladu (make, Ant, Maven)
- odolnost vůči nevalidním zprávám

1.4 Celé znění úlohy

Celé zadání úlohy je možno nalézt online zde.

2 Analýza problému

Řešení úlohy vyžadovalo především nastudování způsobu komunikace pomocí soketů. Následně pak bylo nutno vyhodnotit možné způsoby implementace tohoto typu komunikace v rámci jazyků, jež byly použity pro tvorbu hry (v mém případě Java a C). Další kroky učiněné v rámci analýzy lze rozdělit do dvou kategorií - viz podkapitoly).

2.1 Herní klient

Tvorba herního klienta nevyžadovala příliš analytické činnosti, jelikož jsem se klienta rozhodl vytvořit v jazyce Java (s vyžitím knihoven AWT / Swing). Zmíněný programovací jazyk jsem již pro pár prací použil a jsem tedy obeznámen se syntaxí běžně užívaných příkazů tohoto jazyka. Zřejmě jediné možné úskalí jsem tedy v době analýzy viděl pouze v komunikaci prostřednictvím soketů. Rychle se však ukázalo, že v případě Javy je řešení síťové komunikace poměrně jednoduchou záležitostí i pro začátečníka a vyžaduje minimální režii ze strany programátora.

2.2 Herní server

Ve srovnání s herním klientem vyžadovala tvorba herního serveru o poznání delší čas strávený analýzou. Tento fakt přikládám zejména skutečnosti, že herní server musel být vytvořen v nízkoúrovňovém jazyce (např. C), se kterým pracuji pouze výjimečně a mnoho konstrukcí jazyka jsem si musel před zahájením tvorby programu připomenout. Samotnou kapitolu lze pak vyčlenit pro síťovou komunikaci, jejíž implementace je v případě jazyka C o poznání složitější záležitostí (ve srovnání s Javou).

2.2.1 Síťová komunikace

Již při analýze jsem zjistil, že obsluhu požadavků klientů bude možno v rámci jazyka C řešit dvěma způsoby. Jedno z možných řešení spočívalo ve využití příkazu fork() (vytvoření nového vlákna), jež by byl volán při každém připojení nového klienta. Tento způsob se mi příliš nezamlouval, jelikož při větším počtu připojených klientů by velice pravděpodobně vedl k velkému vytížení serveru (hodně vláken běžících současně) a mohl by vést k nepřehlednosti během ladění programu.

Rozhodl jsem se tedy využít příkazu select(), jež nevyžaduje vytváření dalšího vlákna při připojení nového klienta a umožňuje obsluhu více klientů v rámci jednoho vlákna. Více informací viz kapitola zabývající se implementací.

3 Popis implementace (programátorská dokumentace)

Jelikož během práce byl vytvářen herní server i klient (každý v jiném programovacím jazyce), rozhodl jsem se popis implementace rozdělit do tří částí. V rámci první části je popsán protokol, jež jsem pro hru vytvořil a pomocí kterého si klient vyměňuje zprávy s herním serverem. Druhá sekce textu obsahuje popis implementace serveru - zde je například popsán způsob ochrany serveru proti narušitelům atd. Další část textu pojednává o klientské části aplikace.

3.1 Komunikační protokol

Při návrhu komunikačního protokolu jsem snažil snoubit bezpečnost s jednoduchostí - validní zprávy mají jasně definovanou délku, ve většině případů i obsah. Každá validní zpráva v rámci programu musí začínat znakem "#"a končit znakem "?".

3.1.1 Rozdělení hry na fáze

Hra je z pohledu serveru dělena do čtyř fází. Server si uchovává pro každého připojeného hráče strukturu, v rámci níž jsou uvedeny informace o fázi hry, ve které se hráč nachází. Ve struktuře je uvedena i IP adresa hráče, která slouží jako jednoznačný identifikační údaj.

3.1.2 Ochrana vůči narušitelům

Rozklad hry na jednotlivé fáze je také základní stavební kámen obranného mechanismu mého serveru. Každá fáze hry má totiž jasně definované zprávy, které jsou v dané fázi validní. Tuto skutečnost je možno rovněž chápat jako bezpečnostní opatření, jelikož pro každou fázi hry jsou jasně definované validní zprávy, které mohou od klienta přijít. Pokud přijde od libovolného klienta zpráva, jež neodpovídá požadavkům (délka či obsah), pak dojde k odpojení daného klienta.

3.1.3 Zprávy předávané v 1. fázi hry

V první fázi hry se uživatel nachází, pokud klient úspěšně navázal spojení se serverem. V této fázi je za validní považován pouze jeden typ zpráv.

Zpráva s délkou 17B - přezdívka

Pokud klient v první fázi pošle zprávu o délce 17B, je její obsah požadován za jméno hráče (15 znaků).

Požadavky: jméno hráče musí být uvedeno bez diakritiky a nesmí obsahovat znak procenta (procenta doplňuje klientská aplikace do délky 15 znaků).

Formát zprávy: "#xxxxxxxxxxxxxxx?", kde x = písmeno obsažené v přezdívce uživatele

Odpověď serveru:

- SERVER_WRITE_RECEIVED_OK ("#99?") = server jméno přijal, úspěch => 2. fáze hry
- SERVER_WRITE_RECEIVED_FAIL ("#98?") = zpráva od klienta je nevalidní (chybí počáteční nebo ukončovací znak), chyba

3.1.4 Zprávy předávané ve 2. fázi hry

V druhé fázi hry se uživatel nachází, pokud klient úspěšné prošel první fází (tzn. připojil se k serveru, úspěšně navázal spojení se serverem a odeslal přezdívku). V druhé fázi si uživatel typicky vybírá herní místnost, k níž se připojí. Do této fáze se řadí i vytvoření nové herní místnosti. V dané fázi jsou za validní považovány pouze tři typy zpráv.

Zpráva s délkou 13B končící "0"- připojení k existující hře

Pokud klient pošle zprávu o délce 13B, je považována za validní a předpokládá se, že obsahuje název místnosti, ke které se chce klient připojit (10 znaků) a za ní "0". Poslední znak serveru říká, že hra by již měla být vytvořena a nemá se pokoušet o její založení.

Požadavky: název herní místnosti musí být bez diakritiky a nesmí obsahovat znak procenta.

Formát zprávy: "#xxxxxxxxxx0?", kde x = písmeno obsažené v názvu místnosti

Odpověď serveru:

- SERVER_WRITE_RECEIVED_OK ("#99?") = hra existuje, uživatel připojen => 3. fáze hry

- SERVER_WRITE_RECEIVED_FAIL ("#98?") = zpráva od klienta je nevalidní, chyba
- SERVER_STAGE2_WRITE_ROOM_DOES_NOT_EXIST ("#96?") = zvolená herní místnost neexistuje
- SERVER_STAGE2_WRITE_ROOM_RUNNING ("#95?") = ve zvolené místnosti již běží hra, jejíž součástí není daný uživatel

Zpráva s délkou 13B končící "1"- vytvoření nové hry

Pokud klient pošle zprávu o délce 13B, je považována za validní a pokud končí "1", předpokládá se, že obsahem zprávy je název místnosti, jež chce uživatel založit (10 znaků) a za ní "1". Poslední znak značí, že chce uživatel vytvořit novou místnost (ne se připojit k existující).

Požadavky: název herní místnosti musí být bez diakritiky a nesmí obsahovat znak procenta.

Formát zprávy: "#xxxxxxxxxx1?", kde x = písmeno obsažené v názvu místnosti

Odpověď serveru:

- SERVER_WRITE_RECEIVED_OK ("#99?") = hra úspěšně založena => 3. fáze hry
- SERVER_WRITE_RECEIVED_FAIL ("#98?") = zpráva od klienta je nevalidní, chyba
- SERVER_STAGE2_WRITE_ROOM_EXISTS ("#97?") = herní místnost s daným názvem již existuje, chyba

Zpráva s délkou 3B ("#1?") - získání seznamu herních místností

Pokud klient pošle kód SERVER_STAGE2_GET_GAME_ROOM_LIST ("#1?"), pak je to chápáno jako požadavek o odeslání seznamu existujících herních místností.

Formát zprávy: "#1?"

Odpověď serveru:

- zpráva o délce 52B obsahující seznam herních místností (5 místností maximum * 10B pro název místnosti)

3.1.5 Zprávy předávané ve 3. fázi hry

Ve třetí fázi hry se uživatel nachází, pokud klient úspěšně prošel předcházejícími fázemi (tzn. je připojen, odeslal své jméno a nachází se v některé z herních místností). V této fázi uživatel čeká v herní místnosti na připojení dalšího hráče - má možnost hru odstartovat, či se vrátit do herního lobby. Následuje seznam očekávaných zpráv a možných reakcí na ně.

Zpráva s délkou 4B ("#5?") - odstartování hry

Pokud klient pošle kód SERVER_STAGE3_START_GAME ("#5?") je tato zpráva požadována za validní požadavek o start hry uvnitř herní místnosti, v níž se daný hráč nachází.

Formát zprávy: "#5?"

Odpověď serveru:

- SERVER_WRITE_RECEIVED_OK ("#99?") = požadavek na start hry přijat => 4. fáze hry
- SERVER_WRITE_RECEIVED_FAIL ("#98?") = zpráva od klienta je nevalidní, chyba
- SERVER_STAGE3_WRITE_GAME_CANNOT_START ("#11?") = v místnosti je pouze jeden hráč, chyba

Zpráva s délkou 4B ("#6?") - návrat do lobby

Pokud klient pošle kód SERVER_STAGE3_BACK_LOBBY ("#6?") je tato zpráva požadována za validní požadavek o návrat do lobby. Po zpracování takové zprávy je klient přesunut zpět do fáze 2.

Formát zprávy: "#6?"

Odpověď serveru:

- SERVER_WRITE_RECEIVED_OK ("#99?") = požadavek na přesun do lobby přijat

Zpráva s délkou 4B ("#7?") - zjištění stavu hry

Pokud klient pošle kód SERVER_STAGE3_GAME_STATE ("#7?"), dojde k vyhodnocení stavu hry uvnitř herní místnost (hra běží / neběží). Tento kód je klienty odesílán v periodickém čas. intervalu a klienti tak zjišťují, zda již byla hra odstartována, či nikoliv.

Formát zprávy: "#7?"

Odpověď serveru:

- SERVER_WRITE_RECEIVED_OK ("#99?") = server obdržel zprávu, hra neběží
- SERVER_STAGE3_WRITE_GAME_STARTED ("#10?") = server obdržel zprávu, hra byla spuštěna => 3. fáze hry

3.1.6 Zprávy předávané ve 4. fázi hry

Ve čtvrté fázi hry se uživatel nachází, pokud klient prošel všemi předcházejícími fázemi (tzn. připojen, odeslal přezdívku, vybral místnost a někdo odstartoval hru). Takový hráč se tedy typicky nachází uvnitř některé z herních místností a jsou od něj očekávány následující zprávy.

Zpráva s délkou 4B ("#50?") - získání jmen hráčů

Na požadavek SERVER_STAGE4_GET_PLAYER_NAMES ("#50?") server zareaguje odesláním přezdívek hráčů, jež se nacházejí v herní místnosti společně s hráčem, který tuto zprávu odeslal.

Formát zprávy: "#50?"

Odpověď serveru:

- zpráva o délce 47B obsahující jména hráčů, jež se nacházejí v dané herní místnosti (3 hráči maximum * 15B pro jednu přezdívku)

Zpráva s délkou 4B ("#51?") - získání přidělených karet

Při příchodu kódu SERVER_STAGE4_GET_CARDS ("#51?") server odešle seznam karet, které má uživatel přidělen. Pokud zpráva přijde od hráče, který se do místnosti nově připojil, předchází odeslání seznamu karet ještě přidělení náhodných karet uživateli.

Seznam obsahuje pouze "0"(pokud karta, jež odpovídá danému indexu nebyla přidělena) a "1"(pokud byla přidělena). Indexy pole s kartami jsou shodné pro obě strany - server i klient. Indexy pole karet - viz další kapitola.

Formát zprávy: "#51?"

Odpověď serveru:

- zpráva o délce 34B reprezentující karty přidělené uživateli

Zpráva s délkou 4B ("#01?"- "#44?") - použití přidělené karty

Kódy v rozmezí "#01?"- "#44?"jsou chápány jako identifikační kódy karet a příchod takového kódu je ze strany serveru chápán jako pokus o použití karty ve hře.

Kódy "#01?"- "#32?"jsou přiřazeny běžným kartám (32 karet v balíčku), zbývající kódy jsou pak v rámci aplikace vyhrazeny pro další potřebné informace o kartě, kterou chce uživatel umístit. Kódy "#33?"- "#44?"jsou tedy ze strany klienta odeslány, pokud chce uživatel umístit kartu, jejíž efekt nemusí být ze vzhledu karty patrný => jedná se o "měniče". Uživatel může chtít umístit například červeného měniče a přitom změnit barvu na zelenou - v takovém připadě by byl např. odeslán kód "#37?". Kompletní seznam kódu karet je dostupný níže.

Nejprve je zkontrolována platnost daného úkonu (zejména zjištění, zda je možno danou kartu umístit na předchozí). Následně se zkontroluje, zda je odesílatel na tahu. Závěrem proběhne kontrola, zda uživatel danou kartu opravdu vlastnil (tzn. byla mu serverem přidělena a nejedná se o podvodníka). Od výsledku těchto kontrol se odvíjí odpověď serveru.

Formát zprávy: "#xx?", kde xx = číslo příslušné karty

Kódové označení karet:

- "(#01?") = koule 7
- "(#02?") = koule 8
- "(#03?") = koule 9
- "(#04?") = koule 10
- "(#05?") = koule spodek
- "(#06?") = koule menic
- "(#07?") = koule kral
- "(#08?") = koule eso
- "(#09?") = cervena 7
- "(#10?") = cervena 8
- "(#11?") = cervena 9
- "(#12?") = cervena 10
- "(#13?") = cervena spodek
- "(#14?") = cervena menic
- "(#15?") = cervena kral

- "(#16?") = cervena eso
- "(#17?") = zelena 7
- "(#18?") = zelena 8
- "(#19?") = zelena 9
- "(#20?") = zelena 10
- "(#21?") = zelena spodek
- "(#22?") = zelena menic
- "(#23?") = zelena kral
- "(#24?") = zelena eso
- "(#25?") = zaludy 7
- "(#26?") = zaludy 8
- "(#27?") = zaludy 9
- "(#28?") = zaludy 10
- "(#29?") = zaludy spodek
- "(#30?") = zaludy menic
- "(#31?") = zaludy kral
- "(#32?") = zaludy eso
- "(#33?") = koule měnič => červená
- "(#34?") = koule měnič => zelená
- "(#35?") = koule měnič => žaludy
- "(#36?") = červená měnič => koule
- "(#37?") = červená měnič => zelená
- "(#38?") = červená měnič => žaludy
- "(#39?") = zelená měnič => koule
- "(#40?") = zelená měnič => červená

- "(#41?") = zelená měnič => žaludy
- "(#42?") = žaludy měnič => koule
- "(#43?") = žaludy měnič => červená
- "(#44?") = žaludy měnič => zelená

Odpověď serveru:

- SERVER_WRITE_RECEIVED_OK ("#99?") = karta úspěšně umístěna
- SERVER_WRITE_RECEIVED_FAIL ("#98?") = zpráva od klienta je nevalidní, chyba
- SERVER_STAGE4_WRITE_CANNOT_PLACE ("#69?") = daná karta nemůže být umístěna na předchozí, chyba
- SERVER_STAGE4_WRITE_NOT_TURN_PLACE ("#68?") = odesílatel není na tahu, chyba
- SERVER_STAGE4_MUST_TAKE_1_CARDS ("#67?") = uživatel si musí vzít 1 kartu (umístěna karta 7), chyba
- SERVER_STAGE4_MUST_TAKE_2_CARDS ("#66?") = uživatel si musí vzít 2 karty (umístěna karta 7), chyba
- SERVER_STAGE4_MUST_TAKE_3_CARDS ("#65?") = uživatel si musí vzít 3 karty (umístěna karta 7), chyba
- SERVER_STAGE4_MUST_TAKE_4_CARDS ("#64?") = uživatel si musí vzít 4 karty (umístěna karta 7), chyba
- SERVER_STAGE4_MUST_TAKE_5_CARDS ("#63?") = uživatel si musí vzít 5 karet (umístěna karta 7), chyba
- SERVER_STAGE4_MUST_TAKE_6_CARDS ("#62?") = uživatel si musí vzít 6 karet (umístěna karta 7), chyba
- SERVER_STAGE4_MUST_TAKE_7_CARDS ("#61?") = uživatel si musí vzít 7 karet (umístěna karta 7), chyba
- odpojení klienta od serveru = pokud použita jakákoliv karta, jež uživatel nemá přidělenou -> pokus o hack serveru

Zpráva s délkou 4B ("#45?") - vynechání kola / žádost o kartu

Pokud přijde kód SERVER_STAGE4_TAKE_STAND ("#45?") server jej vyhodnotí jako:

- a) žádost o vynechání kola - pokud je aktuální kartou eso, uživatel "stojí"
- b) žádost o novou kartu - uživateli přidělena další karta z balíčku dostupných karet

Formát zprávy: "#45?"
Odpověď serveru:

- kód karty ("#0?"- "#31?") = karta přidělena, pořadí přidělené karty v seznamu kódů karet, úspěch
- SERVER_WRITE_RECEIVED_FAIL ("#98?") = zpráva od klienta je nevalidní, chyba
- SERVER_STAGE4_WRITE_NOT_TURN_TAKE ("#79?") = karta nebyla přidělena, uživatel není na tahu, chyba
- SERVER_STAGE4_WRITE_NO_FREE_CARDS ("#78?") = karta nebyla přidělena, v balíčku nejsou žádné volné karty, chyba

Zpráva s délkou 3B ("#0?") - žádost o stav hry

Na kód SERVER_PING_CHAR ("#0?") server reaguje odesláním zprávy velikosti 6B ve formátu XYYZ, kde:

- X = pořadí hráče, jež je aktuálně na tahu (v rámci herní místnosti)
- YY = kódové označení aktuální karty ("#01?"- "#44?") NEBO "#45?"- pokud uživatel "stojí"/ bere si další kartu
- Z = pokud umístěná karta "platí"na dalšího uživatele (např. eso) - "#1?"pokud aktivní, jinak "#0?"

Zasílání tohoto kódu je klientem prováděno periodicky, aby byl vždy obeznámen s aktuálním stavem hry.
Formát zprávy: "#0?"
Odpověď serveru:

- zpráva velikosti 6B, specifikace viz výše

Zpráva s délkou 4B ("#52?") - žádost o návrat do lobby

Pokud klient pošle kód SERVER_STAGE4_BACK_LOBBY ("#52?") je tato zpráva požadována za validní požadavek o návrat do lobby. Po zpracování takové zprávy je klient přesunut zpět do fáze 2.

Formát zprávy: "#52?"

Odpověď serveru:

- SERVER_WRITE_RECEIVED_OK ("#99?") = požadavek na přesun do lobby přijat => 2. fáze hry
- SERVER_WRITE_RECEIVED_FAIL ("#98?") = zpráva od klienta je nevalidní, chyba

3.2 Server

3.2.1 Technologie využité během vývoje

Serverová část aplikace byla vytvořena v nízkoúrovňovém jazyce C. Vývoj probíhal zejména ve vývojovém prostředí CLion, jež bylo spuštěno v rámci Linuxového operačního systému Ubuntu.

3.2.2 Požadavky pro spuštění

Vzhledem k volbě jazyka je pro překlad programu nutno mít na cílovém zařízení, na němž má být server spuštěn, nainstalovano GCC či jiný překladač schopný přeložit zdrojové soubory jazyka C. Požadavky na výkon procesoru a velikost paměti RAM jsou v případě mojí aplikace zanedbatelné, s jejím spuštěním by neměl mít problém žádný moderní PC. Je však nutno poznamenat, že požadavky na výkon rostou s počtem připojených klientů a pokud bychom chtěli server reálně nasadit, musíme počítat s obsluhou tisíců klientů a výkonný hardware by byl v takovém případě nevyhnutelný. Testování serveru probíhalo zejména na následující konfiguraci: 12GB RAM, i5-5200U (mobilní CPU), Ubuntu. Server byl při testování na daném stroji stabilní a nevykazoval známky zahlcení ze strany klientů (rychlé reakce na zprávy klientů atp.).

3.2.3 Reakce na zprávy od klienta

Informace o aktuální fázi je pro server důležitá, jelikož od aktuální fáze hráče se odvíjí reakce serveru na zprávu daného uživatele. Pokud je daná

zpráva validní, server patřičně zareaguje (dle protokolu - viz výše). V případě nevalidní zprávy je reakce predikovatelná, klient je zkrátka odpojen.

3.2.4 Popis struktur použitých v rámci aplikace

Zdrojový kód serverové části aplikace disponuje dostatečným množstvím komentářů a proto nepovažuji za vhodné popisovat jednotlivé struktury v rámci této dokumentace.

3.3 Klient

3.3.1 Technologie využité během vývoje

Klient byl (dle požadavků) vytvořen ve vysokoúrovňovém jazyce Java. Vývoj probíhal zejména ve vývojovém prostředí IntelliJ IDEA, jež bylo spuštěno v rámci Linuxového operačního systému Ubuntu.

3.3.2 Požadavky pro spuštění

Na cílovém zařízení je potřeba mít nainstalovanou Javu minimálně ve verzi 11. Je možno použít verzi od firmy Oracle, či volně dostupnou OpenJDK. Systémové požadavky klienta jsou minimální a v dnešní době s během klienta nebude mít problém prakticky žádné zařízení. Nejslabší PC, na kterém byl klient testován disponovalo následující konfigurací: Intel Atom z3735f, 1GB RAM. Jelikož jsem během provozování klienta na zmíněné konfiguraci nepozorval žádné problémy, troufám si tvrdit, že je poměrně dobře optimalizovaný.

Rychlost odezvy klienta je dána zejména tím, že téměr veškeré výpočty, jež program koná (náhodný výběr karet, atp.), jsou vykonávány na straně serveru.

3.3.3 Zasílání zpráv serveru

Použití mnou vytvořeného klienta hráči zaručí, že na server budou odeslány pouze validní zprávy a při hraní by tak neměl narazit na žádný problém. Zasílány jsou zprávy z protokolu (viz výše). I přes tuto skutečnost jsem se serverem zkoušel komunikovat prostřednictvím terminálu a při jeho použití jsem nenarazil na žádný problém.

3.3.4 Popis struktur použitých v rámci aplikace

Klientská část aplikace, podobně jako server, je opatřena mnoha komentáři (javadoc) a zde tedy nebudu strukturu aplikace rozepisovat.

4 Uživatelská dokumentace

Níže je popsán postup, který je nutno provést pro přeložení a následné spuštění serverové i klientské části aplikace (z pohledu uživatele).

4.1 Pravidla hry

Vzhledem k faktu, že jsem vytvořil hru prší s klasickými pravidly, níže jmenuji pouze základní. Kompletní pravidla lze v případě potřeby nalézt online, např. zde.

Pravidla:

- na začátku hry každý hráč dostane čtyři karty
- každá karta má určitou hodnotu a barvu, na kartu lze položit pouze kartu se stejnou hodnotou / barvou
- pokud hráč nemá kartu, jež by byl schopen vyložit a nemůže "stát"(viz dále), pak si musí vzít další kartu
- hráč použije kartu 7 = další hráč musí dát 7 NEBO bere dvě karty
- hráč použije eso = další hráč musí dát eso NEBO "stojí"(nebere si kartu, hraje další v pořadí)
- hráč použije měniče = další hráč musí dát měniče NEBO kartu s barvou, na níž měnič změnil
- hra končí, pokud jeden z hráčů již nemá žádné karty - takový hráč je vítěz

4.2 Herní klient

K překladu a spuštění je potřeba mít nainstalovanou Javu ve verzi 11. Překlad lze provést příkazem "make", jež bude proveden v rámci adresáře s klientskou částí aplikace. Následné spuštění pak lze provést pomocí příkazu "make run".

4.3 Server

K překladu ja potřeba mít nainstalovaný překladač jazyka C. Překlad lze provést příkazem "make", jež bude proveden v rámci adresáře se serverovou částí aplikace. Následné spuštění pak lze provést pomocí příkazu "./gameserver xxxxx", kde x = číslo portu, na kterém server poběží.

5 Závěr

5.1 Funkčnost programu a jeho přínos

Funkčnost programu jsem ověřil a pracuje dle očekávání. Jednalo se o první semestrální práci, v rámci níž jsem řešil síťovou komunikaci. Díky tomuto faktu došlo k rozšíření mých znalostí v rámci jazyka Java i C.

5.2 Zhodnocení zadání

Zadání úlohy se mi zpočátku zdálo velice triviální, ale nakonec se ukázalo, že naprogramování této úlohy vyžaduje nastudování několika algoritmů, z nichž některé jsou, minimálně pro začátečníka v jazyce C, poměrně náročné na implementaci. Po prostudování algoritmů (spojených zejména se síť. komunikací) a vyřešení několika drobných problémů se mi však úlohu podařilo vyřešit.