FAKULTA APLIKOVANÝCH VĚD
ZÁPADOČESKÉ UNIVERZITY
V PLZNI

Fakulta aplikovaných věd

Semestrální práce z předmětu KIV/UPS

Realizace hry Blackjack

15. ledna 2024

Obsah

Zadání semestrální práce z předmětu KIV/UPS

- síťová hra pro více hráčů, architektura server-klient (1:N), PC
- server: C/C++
- klient: Java, C# (i např. Unity), Kotlin nebo jiný vysokoúrovňový jazyk (musí schválit cvičící)

Varianty zadání:

- tahová hra (prší, mariáš, šachy, piškvorky, ...)
- real-time hra (různé skákačky a střílečky, tanky, "Bulánci", ...)
- pseudo-real-time (Pong, Arkanoid, ...)

Požadavky na protokol:

- textový protokol nad transportním protokolem TCP nebo UDP
    - při shodě zadání v rámci cvičení (max. 2) bude každý student používat jiný
- bez šifrování
- využijte znalostí ze cvičení při návrhu (např. transparentnost při přenosu dat, apod.)
- když se nic neděje (žádný hráč nic nedělá), nic se neposílá
    - výjimkou může být občasná ping zpráva
- na každý požadavek přijde nějaká reakce (byť by šlo pouze o jednoznakové potvrzení, že se operace podařila)

Legenda:

- 🔴 červeně jsou označeny body, jejichž nesplnění automaticky vede k vrácení práce
- 🟡 oranžově jsou označeny body, které nemusí vést k vrácení práce (stále jsou ale povinné)
- 🔵 modře jsou označeny body, kvůli kterým práce již nebude vracena

Požadavky na aplikace:

- 🔴 aplikace jsou při odevzdání přeloženy standardním nástrojem pro automatický překlad (make, ant, maven, scons, ...; nikoliv bash skript, ani ručně gcc/javac, ani přes IDE)
    - pokud potřebujete nějakou knihovnu, verzi Javy, .. před prezentací si ji zajistěte a nainstalujte
- 🔴 je zakázáno využití jakékoliv knihovny pro síťovou komunikaci a serializace zpráv - to řeší váš kód sám pouze s BSD sockety (server) a nativní podporou ve standardní knihovně (klient; Java, C#, ..); není dovoleno použít ani C++2y networking rozhraní
- 🟡 kód aplikací je vhodně strukturovaný do modulů, resp. tříd
- 🟡 kód je dostatečně a rozumně dokumentovaný komentáři
- 🔴 aplikace (server i klient) jsou stabilní, nepadají na segfaultu (a jiných), všechny výjimky jsou ošetřené, aplikace se nezasekávají (např. deadlock)
- 🔴 počet hráčů ve hře je omezen pouze pravidly dané hry; vždy by však měla jít dohrát ve 2 lidech (abychom ji mohli testovat)
- 🟡 po vstupu se hráč dostane do "lobby" s místnostmi, hráč má možnost si vybrat místnost a vstoupit do ní (pokud nepřesahuje limit hráčů); případně je zařazen do fronty a čeká na naplnění herní místnosti
- 🔴 hra umožňuje zotavení po výpadku způsobené nečekaným ukončením klienta, krátkodobou nebo dlouhodobou síťovou nedostupností
    - dle pravidel hry se pak buď:
        - čeká na návrat hráče (hra se pozastaví)

        - nečeká na návrat hráče, hra pokračuje a po obnovení se hráč připojí do následujícího kola hry (ale hráči je stále po připojení obnoven stav)
        - nečeká na návrat hráče, hra pokračuje (ale hráči je stále po připojení obnoven stav)
    - krátkodobá nedostupnost nesmí nutit hráče k manuálnímu pokusu o připojení (klient vše provede automaticky)
    - dlouhodobá nedostupnost už by naopak měla (i s příslušnou zprávou hráči)
    - všichni hráči v dané hře musí vědět o výpadku protihráče (krátkodobém, dlouhodobém)
    - hráč, který je nedostupný dlouhodobě, je odebrán ze hry; hra pak může místnost ukončit (dle pravidel, většinou to ani jinak nejde) a vrátit aktivního protihráče zpět do lobby
- ● hráči jsou po skončení hry přesunuti zpět do "lobby"
- ● obě aplikace musí běžet bez nutnosti je restartovat (např. po několika odehraných hrách)
- ● obě aplikace ošetřují nevalidní síťové zprávy; protistranu odpojí při chybě
    - náhodná data nevyhovující protokolu (např. z /dev/urandom)
    - zprávy, které formátu protokolu vyhovují, ale obsahují očividně neplatná data (např. tah figurky na pole -1)
    - zprávy ve špatném stavu hry (např. tah, když hráč není ve hře/na tahu, apod.)
    - zprávy s nevalidními vstupy dle pravidel hry (např. šachy – diagonální tah věží)
    - ● aplikace můžou obsahovat počítadlo nevalidních zpráv a neodpojovat hned po první nevalidní zprávě, až po např. třech
- ● obě aplikace mají nějakou formu záznamu (log)
    - zaznamenávají se informace o stavech hráčů, her, popř. chybové hlášení, apod.

Server:

- ● server je schopen paralelně obsluhovat několik herních místností, aniž by se navzájem ovlivňovaly (jak ve smyslu hry, tak např. synchronizace)
- ● počet místností (limit) je nastavitelný při spouštění serveru, popř. v nějakém konfiguračním souboru
- ● celkový limit hráčů (dohromady ve hře a v lobby) je omezen; rovněž se dá nastavit při spuštění serveru nebo konfigurací
- ● stejně tak lze nastavit IP adresu a port, na které bude server naslouchat (parametr nebo konfigurační soubor; ne hardcoded)

Klient:

- ● klient implementuje grafické uživatelské rozhraní (Swing, JavaFX, Unity, popř. jiné dle možností zvoleného jazyka a prostředí) (ne konzole)
- ● klient umožní zadání adresy (IP nebo hostname) a portu pro připojení k serveru
- ● uživatelské rozhraní není závislé na odezvě protistrany - nezasekává se v průběhu např. připojení na server nebo odesílání zprávy/čekání na odpověď
- ● hráč a klient je jednoznačně identifikovaný přezdívkou (neřešíme kolize)
    - [nepovinné] chcete-li, můžete implementovat i jednoduchou registraci (přezdívka + heslo), aby se kolize vyřešily
- ● všechny uživatelské vstupy jsou ošetřeny na nevalidní hodnoty
    - totéž platí pro např. ošetření tahů ve hře (např. šachy, aby věž nemohla diagonálně, apod.)

- ● klient vždy ukazuje aktuální stav hry - aktuální hrací pole, přezdívky ostatních hráčů, kdo je na tahu, zda není nějaký hráč nedostupný, atp.
- ● klient viditelně informuje o nedostupnosti serveru - při startu hry, v lobby, ve hře
- ● klient viditelně informuje o nedostupnosti protihráče - ve hře

Dokumentace obsahuje:

- základní zkrácený popis hry, ve variantě, ve které jste se rozhodli ji implementovat
- popis protokolu dostatečný pro implementaci alternativního klienta/serveru:
    - formát zpráv
    - přenášené struktury, datové typy
    - význam přenášených dat a kódů
    - omezení vstupních hodnot a validaci dat (omezení na hodnotu, apod.)
    - návaznost zpráv, např. formou stavového diagramu
    - chybové stavy a jejich hlášení (kdy, co znamenají)
- popis implementace klienta a serveru (programátorská dokumentace)
    - dekompozice do modulů/tříd
    - rozvrstvení aplikace
    - použité knihovny, verze prostředí (Java), apod.
    - metoda paralelizace (select, vlákna, procesy)
- požadavky na překlad, spuštění a běh aplikace (verze Javy, gcc, ...)
- postup překladu
- závěr, zhodnocení dosažených výsledků

Průběžné odevzdání během semestru za bonusové body:

1) (cca 5. týden) popis protokolu - stavový diagram a přenášené zprávy (formát, apod.)
2) (cca 9. týden) kostra serveru, volitelně i klienta - připojení, elementární výměna zpráv, např. vylistování seznamu místností a možnost založit novou
3) (cca 12. týden) vyrovnání serveru/klienta s nevalidními stavy, řešení výpadků

Závěrečné odevzdání:

- minimálně je nutné získat alespoň 15 bodů, maximálně je možné získat až 30 bodů
- do termínu stanoveného cvičícím lze získat plný počet bodů (obvykle polovina ledna); poté je za každý den zpoždění (vč. sobot a nedělí) odečten 1 bod z celkového hodnocení práce
- odevzdání musí proběhnout nejpozději do konce ledna (mezní termín)
- student předvede funkčnost řešení na PC v laboratoři UC-326
- server je spuštěn na jednom PC s GNU/Linux, klient na jednom PC s GNU/Linux a druhém PC s MS Windows
- před odevzdáním si student připraví prostředí, aby předvádění mělo hladký průběh - ověří, zda se obě aplikace úspěšně přeloží na obou prostředích, zda je lze spustit a propojit
    - laboratoř je vám k dispozici, pokud v ní zrovna neprobíhá výuka, zkouška nebo jiná akce
- průběh odevzdání bude určitě zahrnovat (v režii studenta):
    - překlad klienta a serveru
    - spuštění s různými parametry
    - odehrání jedné celé hry bez výpadků (jejich simulace) a bez nevalidních dat
    - schopnost reagovat na výpadky (obě aplikace; dle zadání)

- schopnost vyrovnat se s nevalidními daty (obě aplikace; dle zadání)
- ověření náročnosti na systémové prostředky

Užitečné příkazy, tipy a triky:

- server, co naslouchá na 127.0.0.1:10000 a produkuje náhodná data:
  - cat /dev/urandom | nc -l 127.0.0.1 -p 10000
- klient, co se připojí na 127.0.0.1:10000 a produkuje náhodná data:
  - cat /dev/urandom | nc 127.0.0.1 10000
- simulace výpadku klienta/serveru (pouze vzdáleně):
  - stylem DROP (zahazuje pakety):
    - iptables -A INPUT -p tcp --dport 10000 -j DROP
  - stylem REJECT (odmítá pakety a odpovídá patřičnou ICMP zprávou):
    - iptables -A INPUT -p tcp --dport 10000 -j REJECT
  - odebrání pravidel - záměna -A za -D
- je vhodné (nikoliv povinné) zkusit, zda neuniká na serveru nějaká paměť (valgrind)
  - aby se zobrazila čísla řádku, kompilujte (gcc, clang) s přepínačem –g
  - opravením chyb, které valgrind vypíše, můžete výrazně snížit riziko „náhodných“ pádů a chyb v době odevzdání
- dodatečné verze Javy a knihoven neinstalujte do svého home – hrozí vyčerpání místa na vašem diskovém prostoru v AFS; místo toho použijte na Linuxu např. lokální složku /tmp
- pokud server padá na cílovém prostředí (Linux) a není jasné kde, přeložte rovněž s přepínačem -g a před jeho spuštěním použijte příkaz "ulimit -c unlimited" (popř. místo ‘unlimited‘ použijte nějaké rozumně velké množství paměti)
  - po pádu se vygeneruje soubor s názvem "core" - ten lze analyzovat nástrojem gdb, např. "gdb -c core muj-server" (za předpokladu, že zkompilovaná binárka serveru se jmenuje "muj-server")
  - v gdb můžete vypsat aktuální zanoření (zde v době pádu) příkazem "bt" ("backtrace")
  - případně můžete rovnou spouštět server uvnitř GDB ("gdb muj-server", příkaz "r")
  - podrobnější dokumentace nástroje GDB a jeho ovládání je například zde: http://sourceware.org/gdb/current/onlinedocs/gdb/

1 Úvod

1.1 Cíl

Hlavním cílem této semestrální práce je vytvoření a implementace serverově-klientní architektury pro hru Blackjack. Zahrnuje to vývoj stabilního a robustního serveru v jazyce C++, který bude schopen zvládat současný provoz více herních lobby, a vývoj klienta v jazyce Java s využitím JavaFX pro grafické uživatelské rozhraní.

Zvláštní pozornost je věnována schopnosti jak serveru, tak klienta, vyrovnat se s neočekávaným odpojením, s cílem zpracovávat všechny situace bez chyb a snažit se obnovit předchozí stav hry před ztrátou spojení. Klient by měl být schopen automaticky se znovu přihlásit na server, pokud je to možné; v případě delšího přerušení spojení by se měl uživatel přihlásit manuálně.

Dále je důležité, aby všichni účastníci hry nebo lobby byli informováni v případě, že někdo ze hry odpojí. Každá chyba nebo upozornění by mělo jasně signalizovat, co se děje, aby bylo uživatelům poskytnuto jasné a srozumitelné rozhraní.

Cílem je vytvořit efektivní, uživatelsky přívětivý a bezpečný systém pro hru Blackjack, který bude technicky odolný a schopný efektivně řešit všechny výzvy spojené s online hraním.

1.2 Pravidla hry

Hra Blackjack, kterou tato práce implementuje, vychází z klasických pravidel oblíbené karetní hry, ale s několika modifikacemi. Hlavní rozdíl spočívá v tom, že hráči nehrají proti dealerovi, jelikož tato možnost ve hře není zahrnuta. Dále hra nezahrnuje žádné sázky, protože jako autor práce nepodporuji hazardní hraní.

Na začátku hry každý hráč obdrží dvě karty. Hraje se s kompletním balíčkem 52 karet bez žolíků. Karty s hodnotami od 2 do 10 mají hodnotu odpovídající svému číslu bez ohledu na barvu. Karty jako jsou kluk, dáma a král mají hodnotu 10, zatímco eso má hodnotu 11.

Cílem hry je dosáhnout součtu hodnot karet co nejbližšího 21. Pokud hráč dosáhne hodnoty menší než 21 a není v hře žádný jiný hráč s vyšším součtem, který by také byl menší nebo roven 21, vyhrává tento hráč. Pokud hráč překročí hodnotu 21, ale není v lobby žádný hráč, kdo by měl 21 nebo méně, a zároveň není nikdo, kdo by měl více než 21, ale méně než překročený součet prvního hráče, první hráč vyhrává. Může být více vítězů, pokud mají stejný počet bodů, a tento počet se považuje za vítězný.

Hráči mají během svého tahu dvě možnosti. První možností je vzít si kartu z balíčku, čímž zvýší celkovou hodnotu svých karet v ruce s cílem přiblížit se k součtu 21. Druhou možností je odmítnout další kartu a tím 'pasovat', což znamená, že hráč v tomto kole

již dále nezvyšuje hodnotu svých karet. Toto rozhodnutí je důležité, jelikož hráč musí pečlivě zvážit riziko překročení součtu 21, které by znamenalo prohru v tomto kole.

Hra pokračuje, dokud všichni hráči v hře neřeknou "pas". Jakmile to všichni hráči učiní, je provedeno vyhodnocení vítězů. Počet hráčů v lobby je omezen na maximálně 26 osob, což souvisí s tím, že se hraje s 52 kartami a každý hráč obdrží dvě karty.

2 Popis komunikačního protokolu

2.1 Základní struktura zpráv

V této sekci jsou popsána základní pravidla komunikačního protokolu použitého v aplikaci Blackjack. Každá zpráva odeslaná mezi klientem a serverem musí následovat specifický formát pro správné zpracování.

2.1.1 Prefix zprávy

Každá zpráva začíná prefixem "AABJCK". Pokud tento prefix není přítomen, zpráva nebude akceptována a klient, který zprávu odeslal, bude odpojen.

2.1.2 Operační kód

Následuje operační kód (operational code), který je vždy dvouciferný a rozpětí hodnot má od "00"do "99". Operační kód určuje typ operace nebo akce, která má být provedena. Jakákoliv zpráva s neplatným nebo chybějícím operačním kódem povede k odpojení klienta.

2.1.3 Délka zprávy

Po operačním kódu následuje čtyřciferná hodnota udávající délku zprávy. Hodnota začíná na "0000"a končí na "9999". Tato délka musí zahrnovat i případné znaky konce řádku. Nesprávná délka zprávy nebo nesoulad s reálnou délkou obsahu zprávy rovněž povede k odpojení klienta.

2.1.4 Formát a zpracování zpráv

Následující obsah zprávy závisí na operačním kódu a je validován podle něj. Server je schopen zpracovat různé formáty zpráv, včetně řídících znaků pro konec řádku.

2.2 Podporované operační kódy

V této aplikaci jsou momentálně podporovány následující operační kódy:

- **00 - Registrace:** Tento kód je použit pro registraci nového uživatele v systému. Klient odesílá požadované údaje a server zpracuje registraci.
- **01 - Přihlášení:** Umožňuje klientovi přihlásit se k existujícímu účtu. Klient posílá své přihlašovací údaje a server ověřuje jejich správnost.
- **02 - Odhlášení nebo opuštění lobby:** Tento kód je použit pro odhlášení uživatele ze systému nebo pro opuštění herní místnosti (lobby).
- **03 - Vytvoření lobby:** Umožňuje klientovi vytvořit novou herní místnost, do které se mohou ostatní hráči připojit.

- **04 - Připojení do lobby:** Slouží k připojení klienta do již existující herní místnosti.
- **05 - Smazání lobby:** Umožňuje vlastníkovi herní místnosti tuto místnost smazat.
- **06 - Herní akce:** Tento kód je použit pro různé akce v rámci hry, jako je například tahání karty nebo pasování.
- **07 - Ping:** Jedná se o jednoduchý test spojení mezi klientem a serverem, který slouží k ověření, zda je spojení stále aktivní.

V této struktuře protokolu je každá operace přiřazena k určitému kódu, který jednoznačně definuje typ požadavku nebo odpovědi. Tímto způsobem je komunikace mezi klientem a serverem udržována přehledná a efektivní.

2.3 Operační kód 00 - Registrace

Operační kód 00 je určen pro proces registrace nového uživatele. Příklad zprávy vypadá takto: `AABJCK00XXXXaa11dre;123456;Andrei;Akhramchuk`, kde `XXXX` označuje délku zprávy. Formát zprávy je následující:

- **Login:** Uveden na prvním místě a oddělen středníkem (`;`).
- **Heslo:** Uvedeno na druhém místě, opět odděleno středníkem.
- **Jméno:** Uvedeno na třetím místě.
- **Příjmení:** Uvedeno na čtvrtém místě.

Je nezbytné, aby žádná z těchto pozic nebyla prázdná a počet pozic musí být přesně čtyři. Jakékoli odchylky vedou k chybě a vrácení zprávy s chybovým kódem.

V reakci na tuto zprávu server odpovídá zprávou, která začíná číslem `1` pro úspěšnou registraci nebo `0` v případě chyby. V případě úspěšné registrace je formát odpovědi `1;[aa11dre;321678;Andrei;Akhramchuk]`, kde jsou údaje uživatele vraceny ve formátu podobném odeslanému.

Pokud dojde k chybě, odpověď začíná číslem `0`, následuje středník a dále text chyby v kapitálkách, například `0;USER_ALREADY_EXISTS`.

Mezi možné chybové kódy patří:

- **INVALID_MESSAGE_FORMAT:** Neplatný formát zprávy.
- **USER_ALREADY_EXISTS:** Uživatel již existuje.

Tento mechanismus zajišťuje, že proces registrace je bezpečný, přesný a uživatelsky přívětivý.

2.4 Operační kód 01 - Přihlášení

Operační kód 01 je použit pro proces přihlášení uživatele. Příklad odeslané zprávy: `AABJCK01XXXXaa11dre;321678`, kde `XXXX` označuje délku zprávy. Formát zprávy obsahuje:

- **Login:** Uveden na prvním místě a oddělen středníkem (`;`).
- **Heslo:** Uvedeno na druhém místě, opět odděleno středníkem.

V reakci na tuto zprávu server odpovídá zprávou, která začíná číslem `1` pro úspěšné přihlášení nebo `0` v případě chyby. V případě chyby následuje středník a text chyby v kapitálkách, například `0;INVALID_LOGIN_CREDENTIALS`.

Možné chybové kódy:

- **INVALID_MESSAGE_FORMAT:** Neplatný formát zprávy.
- **YOU_ARE_ALREADY_LOGGED_IN:** Uživatel je již přihlášen.
- **USER_ALREADY_LOGGED_IN_BY_ANOTHER_CLIENT:** Uživatel již přihlášen na jiném klientu.
- **USER_NOT_FOUND:** Uživatel nebyl nalezen.
- **INVALID_LOGIN_CREDENTIALS:** Neplatné přihlašovací údaje.

V případě úspěchu následuje informace o uživateli ve formátu `1;[login;name;surname;online]`, kde `online` je `1` nebo `0` indikující, zda je uživatel online. Po uživatelských údajích následuje středník a další informace podle aktuálního stavu uživatele, buď `MENU`, `LOBBY` nebo `GAME`, které určují další kontext komunikace.

MENU

V případě, že po údajích o uživateli a středníku (`;`) následuje slovo `MENU`, identifikujeme, že se uživatel nachází v menu. Po tomto slově následuje obal seznamu (`[]`), ve kterém jsou umístěny objekty lobby. Každý objekt lobby je také obalen ve `[]` a oddělen středníkem.

Struktura objektu lobby je následující:

- **ID Lobby:** První hodnota představuje ID lobby.
- **Název Lobby:** Druhá hodnota je název lobby.
- **Maximální počet hráčů:** Třetí hodnota udává maximální počet hráčů v lobby.
- **Přítomnost hesla:** Čtvrtá hodnota (0 nebo 1) určuje, zda lobby má heslo.
- **Počet hráčů:** Pátá hodnota udává aktuální počet hráčů.
- **Admin Lobby:** Následuje objekt uživatele, který může být prázdný, identifikující admina lobby. Tento objekt obsahuje login, jméno, příjmení a indikátor online stavu (0 nebo 1).

- **Vlastník Lobby:** Další objekt uživatele, který identifikuje vlastníka lobby s podobnými údaji jako u admina.
- **Stav hry:** Poslední hodnota (0 nebo 1) určuje, zda je hra v lobby již spuštěna.

Příklad struktury lobby: `[1;lobby2;4;0;0;[];[aa11dre;Andrei;Akhramchuk;1];0]`, kde první hodnota je ID lobby, druhá je název, třetí maximální počet hráčů, čtvrtá indikuje přítomnost hesla, pátá je počet hráčů, šestá a sedmá jsou údaje o adminovi a vlastníkovi lobby a poslední hodnota udává stav hry.

LOBBY

Pokud po údajích o uživateli a středníku (`;`) následuje slovo `LOBBY`, identifikujeme, že se uživatel nachází v herní místnosti (lobby). Struktura zprávy je pak rozšířena o seznam aktuálních hráčů v lobby.

Struktura objektu lobby v tomto případě je následující:

- **ID Lobby, Název, Maximální počet hráčů, Přítomnost hesla, Počet hráčů:** Stejné jako v předchozím příkladu.
- **Admin a Vlastník Lobby:** Objekty uživatelů identifikující admina a vlastníka lobby.
- **Seznam hráčů:** Obalený ve `[]`, obsahuje seznam aktuálních hráčů v lobby, kde každý hráč je reprezentován objektem `[login;name;surname;online]`.
- **Stav hry:** Hodnota 0 indikuje, že hra v lobby ještě nezačala. Hodnota 1 by zde neměla být, protože bychom dostali předponu `GAME` místo `LOBBY`.

Příklad struktury lobby s hráči: `[1;lobby2;4;0;2;[aa11dre;Andrei;Akhramchuk;1];[aa11dre;Andrei;Akhramchuk;1];[[aa11dre;Andrei;Akhramchuk;1];[1;1;1;0]];0]`, kde seznam hráčů představuje aktuální přítomné hráče v lobby a poslední hodnota určuje, zda hra již byla zahájena.

Tento formát zprávy umožňuje uživatelům přehledně vidět, kdo je v lobby, a zda hra již byla nebo nebyla zahájena.

GAME

Pokud po údajích o uživateli a středníku (`;`) následuje slovo `GAME`, identifikujeme, že se uživatel nachází ve hře. Struktura zprávy v tomto případě obsahuje následující prvky:

- **ID Lobby, Název, Maximální počet hráčů, Přítomnost hesla, Počet hráčů, Admin a Vlastník Lobby:** Stejné jako v příkladu lobby.
- **Stav hry:** Hodnota `1` indikuje, že hra je aktivní.
- **Seznam hráčů:** Obalený ve `[]`, obsahuje seznam hráčů ve hře. První objekt v seznamu představuje hráče, který je na řadě, s údaji `login;name;surname;online status`.

- **Detaily hráčů:** Další objekty hráčů obsahují `login;name;surname;online status` a následně `1` nebo `0`. Hodnota `1` indikuje, že karty hráče jsou viditelné a následuje seznam karet hráče obalený ve `[]`, kde každá karta je reprezentována dvoupísmenným kódem (např. `S7` pro sedm pik, `HJ` pro Jack srdcí) spolu s celkovou hodnotou karet. Hodnota `0` indikuje, že karty hráče jsou skryté, a zobrazuje pouze počet skrytých karet bez specifikace.

Příklad struktury zprávy ve hře: `[0;lobby2;4;0;2[aa11dre;Andrei;Akhramchuk;1];[aa11dre;Andrei;Akhramchuk;1];[[aa11dre;Andrei;Akhramchuk;1];[1;1;1;1]];1;[[aa11dre;Andrei;Akhramchuk;1];[aa11dre;Andrei;Akhramchuk;1;1;[D7;S2];9];[1;1;1;1;0;2]]`, kde první část identifikuje lobby a druhá část poskytuje informace o hráčích a hře, včetně viditelných a skrytých karet.

2.5 Operační kód 02 - Odhlášení nebo Opuštění lobby

Operační kód 02 slouží buď pro odhlášení uživatele ze systému (LOGOUT) nebo pro opuštění herní místnosti (EXITLOBBY). Odeslaná zpráva obsahuje pouze konstantní řetězec, buď `LOGOUT` nebo `EXITLOBBY`.

LOGOUT

V případě `LOGOUT` je odpověď serveru ve formátu, který začíná `1;` pro úspěšné odhlášení nebo `0;` v případě chyby. Příklad úspěšné odpovědi je `1;LOGOUT_SUCCESS`.

EXITLOBBY

V případě použití `EXITLOBBY` a neúspěchu v procesu, server odpovídá zprávou začínající `0;`, následovanou textem chyby. Možné chyby zahrnují:

- **USER_NOT_AUTHENTICATED:** Uživatel není ověřen.
- **NOT_IN_A_LOBBY:** Uživatel není v lobby.

V případě úspěšného opuštění lobby je odpověď serveru ve formátu odpovídajícím stavu menu, jak je popsáno v sekci `MENU` operačního kódu 01. Například: `AABJCK1200661;MENU[[0;lobby2;4;0;1;[1;1;1;0];[aa11dre;Andrei;Akhramchuk;1];0]]`. Toto umožňuje uživateli přehledně vidět stav menu po opuštění lobby.

Tento přístup k operačnímu kódu 02 umožňuje jasnou a efektivní komunikaci jak pro akce odhlášení, tak pro opuštění herní místnosti, a zajišťuje, že uživatel má přehled o svém aktuálním stavu v aplikaci.

2.6 Operační kód 03 - Vytvoření lobby

Operační kód 03 je použit pro vytvoření nové herní místnosti (lobby). Formát zprávy pro vytvoření lobby je `JMENO;početHráčů;přítomnostHesla`, kde:

- **JMENO:** Název lobby.

- **Počet hráčů:** Maximální počet hráčů v lobby.
- **Přítomnost hesla:** Hodnota `0` nebo `1`, indikující, zda lobby má heslo. Pokud je hodnota `1`, očekává se následování hesla.

Příklad zprávy pro vytvoření lobby s heslem: `AABJCK030018JMENO;0;1;123321`.
V reakci na tuto zprávu server odpovídá zprávou začínající `1;` pro úspěšné vytvoření nebo `0;` v případě chyby. V případě chyby může následovat jedna z následujících konstantních zpráv:

- **USER_NOT_AUTHENTICATED:** Uživatel není ověřen.
- **INVALID_MESSAGE_FORMAT:** Neplatný formát zprávy.
- **INVALID_MAX_PLAYERS_SHOULD_BE_BETWEEN_1_AND_26:** Neplatný maximální počet hráčů, měl by být mezi 1 a 26.
- **LOBBY_WITH_THE_SAME_NAME_ALREADY_EXISTS:** Lobby se stejným jménem již existuje.

V případě úspěšného vytvoření lobby server odpovídá zprávou, která začíná `1;` a následuje stav menu. Příklad úspěšné odpovědi může vypadat takto: `AABJCK1301181;MENU[[1;lobby4;4;0;0;[];[aa11dre;Andrei;Akhramchuk;1];0];[0;lobby2;4;0;1;[1;1;1;0];[aa11dre;Andrei;Akhramchuk;1];0]]`.

Tato odpověď zahrnuje informace o všech dostupných herních místnostech v menu, včetně nově vytvořené lobby. Struktura každého objektu lobby v seznamu obsahuje:

- **ID Lobby:** Jedinečný identifikátor lobby.
- **Název Lobby:** Název herní místnosti.
- **Maximální počet hráčů:** Maximální kapacita hráčů v lobby.
- **Přítomnost hesla:** Indikátor, zda je lobby chráněna heslem.
- **Počet hráčů:** Aktuální počet hráčů v lobby.
- **Admin Lobby:** Informace o adminovi lobby.
- **Stav hry:** Indikuje, zda je hra v lobby již zahájena.

Tento přístup k operačnímu kódu 03 umožňuje uživatelům vytvářet herní místnosti s jasnou a efektivní komunikací o výsledku tohoto procesu.

2.7 Operační kód 04 - Vstup do lobby

Operační kód 04 je použit pro připojení uživatele k existující herní místnosti (lobby). Formát zprávy pro vstup do lobby je `JMENO;přítomnostHesla`, kde:

- **JMENO:** Název lobby.
- **Přítomnost hesla:** Hodnota `0` nebo `1`, indikující, zda je pro vstup do lobby vyžadováno heslo. Pokud je hodnota `1`, očekává se následování hesla.

Příklad zprávy pro vstup do lobby s heslem: `AABJCK040016JMENO;1;123321`.

V reakci na tuto zprávu server odpovídá zprávou začínající `1;` pro úspěšný vstup nebo `0;` v případě chyby. V případě chyby může následovat jedna z následujících konstantních zpráv:

- **USER_NOT_AUTHENTICATED:** Uživatel není ověřen.
- **INVALID_MESSAGE_FORMAT:** Neplatný formát zprávy.
- **LOBBY_NOT_FOUND:** Lobby nenalezena.
- **CANNOT_ENTER_LOBBY_WITH_ACTIVE_GAME:** Nelze vstoupit do lobby s aktivní hrou.
- **YOU_ARE_ALREADY_IN_THIS_LOBBY:** Uživatel je již v této lobby.
- **YOU_ARE_ALREADY_IN_ANOTHER_LOBBY:** Uživatel je již v jiné lobby.
- **LOBBY_IS_FULL:** Lobby je plná.
- **PASSWORD_REQUIRED_TO_JOIN_THIS_LOBBY:** Vyžadováno heslo pro vstup do této lobby.
- **INCORRECT_PASSWORD:** Nesprávné heslo.

V případě úspěchu je odpověď serveru ve formátu odpovídajícím stavu v lobby, podobně jako v sekci `LOBBY` operačního kódu 01. Například: `AABJCK1401171;LOBBY[1;lobby4;4;0;1;[aa11dre;Andrei;Akhramchuk;1];[aa11dre;Andrei;Akhramchuk;1];[aa11dre;Andrei;Akhramchuk;1]];0]`.

Tato odpověď poskytuje uživateli informace o lobby, do které se právě připojil, včetně seznamu hráčů a stavu hry. Tímto způsobem je uživatelům umožněno efektivně se orientovat v aktuálním prostředí herní místnosti a získat přehled o dostupných možnostech a hráčích v ní.

Tento přístup k operačnímu kódu 04 umožňuje uživatelům plynule se připojovat k herním místnostem a efektivně komunikovat s herním serverem při vstupu do lobby.

2.8 Operační kód 05 - Odstranění lobby

Operační kód 05 je použit pro odstranění existující herní místnosti. Formát zprávy pro odstranění lobby je `JMENO;přítomnostHesla`, kde:

- **JMENO:** Název lobby k odstranění.
- **Přítomnost hesla:** Hodnota `0` nebo `1`, indikující, zda je pro odstranění lobby vyžadováno heslo. Pokud je hodnota `1`, očekává se následování hesla.

Příklad zprávy pro odstranění lobby s heslem: `AABJCK050016JMENO;1;123321`.

V reakci na tuto zprávu server odpovídá zprávou začínající `1;` pro úspěšné odstranění nebo `0;` v případě chyby. V případě chyby může následovat jedna z následujících konstantních zpráv:

- **USER_NOT_AUTHENTICATED:** Uživatel není ověřen.
- **INVALID_MESSAGE_FORMAT:** Neplatný formát zprávy.
- **LOBBY_NOT_FOUND:** Lobby nenalezena.
- **ONLY_THE_LOBBY_CREATOR_CAN_DELETE_THE_LOBBY:** Pouze tvůrce lobby může lobby odstranit.
- **PASSWORD_REQUIRED_TO_DELETE_THIS_LOBBY:** Pro odstranění této lobby je vyžadováno heslo.
- **CANNOT_DELETE_LOBBY_WITH_ACTIVE_GAME:** Nelze odstranit lobby s aktivní hrou.
- **INCORRECT_PASSWORD:** Nesprávné heslo.

V případě úspěchu je odpověď serveru ve formátu odpovídajícím stavu menu, podobně jako v sekci `MENU` operačního kódu 01. Například: `AABJCK1500591;MENU[[1;lobby4;4;0;0;[];[aa11dre;Andrei;Akhramchuk;1];0]]`.

Tato odpověď poskytuje uživateli potvrzení o úspěšném odstranění vybrané lobby a zároveň aktualizovaný seznam dostupných herních místností v menu.

Tento přístup k operačnímu kódu 05 umožňuje uživatelům efektivně spravovat své herní místnosti a zajišťuje jasnou komunikaci o výsledcích jejich akcí.

2.9 Operační kód 06 - Herní akce

Operační kód 06 zahrnuje akce `START`, `PASS`, a `TAKE`, přičemž každá zpráva obsahuje pouze jedno slovo určující akci.

Chybové zprávy

Server odpovídá zprávou začínající `0;` následovanou chybovou zprávou v případě chyby:

- **USER_NOT_AUTHENTICATED:** Uživatel není ověřen.
- **NOT_IN_A_LOBBY:** Uživatel není v lobby.
- **INVALID_MESSAGE_FORMAT:** Neplatný formát zprávy.
- **UNKNOWN_SUBCOMMAND:** Neznámý podpříkaz.
- **ITS_NOT_YOUR_TURN:** Není váš tah (pro `PASS` a `TAKE`).

Pro `START` mohou být specifické chyby:

- **NOT_ALL_PLAYERS_HAVE_CONNECTED:** Ne všichni hráči jsou připojeni.
- **GAME_ALREADY_STARTED:** Hra již byla zahájena.
- **ONLY_THE_LOBBY_ADMIN_CAN_START_A_GAME:** Pouze admin lobby může hru zahájit.

Iterativní zprávy pro `START`

Server posílá iterativní zprávy s informacemi o kartách hráčů, strukturované podobně jako v sekci `GAME` operačního kódu 01, například:

- `START;2;[aa11dre;Andrei;Akhramchuk;1;0;2];[1;1;1;1;1;[]]`
- `START;2;[aa11dre;Andrei;Akhramchuk;1;0;2];[1;1;1;1;1;[C6;S6];12]`

Tyto zprávy informují hráče o rozdávání karet a aktuálním stavu hry.

Zprávy `TURN`

Kromě `START`, `PASS` a `TAKE`, server rovněž odesílá zprávy `TURN`, které indikují, čí tah je. Formát zprávy je `TURN[0/1];loginHráče`, kde `0` nebo `1` označuje, zda je tah příjemce zprávy.

Akce `PASS` a `TAKE`

Pro akce `PASS` a `TAKE` mohou být zprávy podobné `START`, ale s odpovídajícími akcemi. Příklady zpráv:

- `TAKE;2;[aa11dre;Andrei;Akhramchuk;1;0;3];[1;1;1;1;1;[C6;S6];12]`
- `PASS;2;[aa11dre;Andrei;Akhramchuk;1;0;2];[1;1;1;1;1;[C6;S6];12]`

Tyto zprávy reflektují změny ve hře po provedení akcí `PASS` nebo `TAKE` jedním z hráčů.

Konec hry - `END`

Když všichni hráči zvolí `PASS`, server odesílá zprávu `END`, která signalizuje konec hry. Struktura zprávy `END` je následující:

- **Počet hráčů:** Udává počet hráčů ve hře.
- **Informace o hráčích:** Obsahuje detaily o hráčích, včetně jejich karet, které jsou nyní všechny odhalené.
- **Počet vítězů:** Udává, kolik hráčů vyhrálo hru.
- **Seznam vítězů:** Obsahuje přihlašovací jména vítězů, obalená ve `[]` a oddělená středníky.

Příklad zprávy `END`:

`END;2;[aa11dre;Andrei;Akhramchuk;1;1;[S3;C5;CK];18];[1;1;1;1;1;[C6;S6];12];1;[aa11dre]`

Tato zpráva poskytuje hráčům kompletní informace o výsledku hry, včetně detailů o kartách všech hráčů a identifikaci vítězů.

Tento komplexní přístup k operačnímu kódu 06 umožňuje hráčům získat jasný a úplný přehled o průběhu a výsledku každé hry v Blackjacku.

2.10 Operační kód 07 - PING

Operační kód 07, reprezentovaný zprávou `PING`, je jednoduchý, ale klíčový pro správnou funkci celé serverově-klientní architektury. Tato zpráva slouží jako kontrolní mechanismus pro ověření aktivního spojení mezi klientem a serverem.

Formát zprávy

Zpráva `PING` je velmi jednoduchá a obsahuje pouze slovo `PING`. Příklad zprávy: `AABJCK070004PING`.

Odpověď serveru

Očekávaná odpověď od serveru začíná číslem `1`, následovanou středníkem a slovem `PONG`. Tato odpověď potvrzuje, že spojení mezi serverem a klientem je funkční. Například: `1;PONG`.

Význam a chování

- **Validační PING:** Po spuštění klient provádí validační PING, na který očekává odpověď od serveru. Pokud odpověď není správná, nebo není obdržena, klient může ukončit komunikaci a hru.
- **Detekce odpojení:** Pokud klient nepošle zprávu `PING` během 5 sekund a nachází se v lobby, ostatní hráči v lobby obdrží upozornění o jeho odpojení. Pokud klient nepošle `PING` po dobu 15 sekund a nachází se ve hře, hra bude ukončena, hráči budou přesunuti zpět do lobby a obdrží odpovídající zprávu podle stavu `LOBBY` operačního kódu 01.

Tento jednoduchý, ale zásadní mechanismus zajišťuje stabilní a spolehlivou komunikaci mezi serverem a klienty během celé hry.

2.11

Vizualizace komunikačního protokolu Pro lepší vizuální pochopení struktury a toku zpráv v komunikačním protokolu Blackjack aplikace je v příloze A přiložen detailní SVG obrázek. Tento obrázek poskytuje grafický přehled návaznosti zpráv mezi serverem a klientem, což umožňuje hlubší vhled do fungování celého systému. Vizualizaci najdete v příloze.

3 BlackJack Server

3.1 Přehled

BlackJack Server představuje jádro celé hry Blackjack a je napsán v jazyce C++. Jedná se o vícevláknovou aplikaci, kde počet vláken je přímo závislý na počtu uživatelů připojených k serveru. Server kompletně zpracovává celou logiku aplikace, počínaje registrací až po samotnou hru.

3.2 Architektura

Server není připojen k databázi, což znamená, že v případě jeho pádu dojde ke ztrátě všech dat. Tato skutečnost vyžaduje pečlivou správu a zálohování dat, aby se minimalizovalo riziko ztráty.

3.3 Zpracování požadavků a stresová odolnost

Důraz je kladen na stresovou odolnost serveru vůči různým nevalidním vstupům. Server je navržen tak, aby mohl efektivně zpracovávat různé druhy požadavků a v případě potřeby nevalidní požadavky odmítnout a příslušného klienta odpojit. Tento přístup zajišťuje stabilitu a bezpečnost systému při zachování vysoké úrovně výkonu.

3.4 Struktura Tříd

3.4.1 Utils

Třída `Utils` je navržena jako pomocná třída, která zahrnuje funkce běžně používané v různých částech programu. Tímto přístupem se snižuje redundantnost kódu a udržuje se jeho čistota a srozumitelnost. Třída obsahuje metody jako `generatePositiveResponse`, `generateNegativeResponse`, `areAllComponentsValid`, `trimCompare` a další, které zjednodušují a optimalizují celkovou strukturu programu.

3.4.2 User

Třída `User` reprezentuje uživatele v systému. Tato třída je navržena tak, aby spojovala informace o uživateli s dalšími třídami, jako jsou `ClientInfo` a `Lobby`. Třída `User` obsahuje atributy jako `login`, `password`, `name`, `surname` a `disconnectCounter`. Dále zahrnuje slabé reference (`weak_ptr`) na třídy `Lobby` a `ClientInfo`, které představují lobby, ve které se uživatel nachází, a klienta, který je s uživatelem spojen.

3.4.3 Socket

Třída `Socket` představuje abstrakci socketu v rámci serveru. Její hlavní úlohou je správa socketových deskriptorů, což zahrnuje vytváření, uzavírání a manipulaci se socketem.

Konstruktor třídy přijímá parametry jako `domain`, `type` a `protocol`, a v případě selhání její vytvoření vyvolává výjimku `std::runtime_error`. Třída dále zajišťuje, že socket bude správně uzavřen při zničení objektu.

3.4.4 ClientInfo

Třída `ClientInfo` slouží jako kontejner pro informace o klientovi. Obsahuje důležité údaje jako `client_socket` a `addr`, které představují socketové číslo a adresu klienta. Třída také udržuje slabou referenci (`weak_ptr`) na objekt `User`, což umožňuje přiřadit konkrétního uživatele k danému klientovi. Metody `setClientSocket`, `getAddr`, `setUser` a `getUser` poskytují rozhraní pro manipulaci s těmito údaji.

3.4.5 Card

Třída `Card` reprezentuje hrací kartu, která má určitý znak (suit) a hodnotu (rank). Tato třída je základním stavebním prvkem pro reprezentaci karet v hře. Enumerace `Suit` a `Rank` definují možné znaky a hodnoty karet. Metoda `value()` je zásadní pro určení hodnoty karty v kontextu hry, což je klíčové pro pravidla a logiku hry.

3.4.6 ClientGuard

Třída `ClientGuard` slouží jako ochranný mechanismus pro správu klientů v kontextu vícevláknového přístupu. Tato třída zajišťuje synchronizaci přístupu k seznamu klientů (`clients`) pomocí mutexu (`client_mutex`). Konstruktor přijímá seznam klientů, jednoho klienta a mutex, a destruktor zajišťuje uvolnění zámku při zničení objektu. Třída je klíčová pro zajištění bezpečnosti a konzistence dat ve vícevláknovém prostředí.

3.4.7 Constants

Třída `Constants` obsahuje konstanty a sdílené proměnné používané v celém serveru. Tato třída zahrnuje konstanty jako `DATALEN`, `HEADER_SIZE`, různé operační kódy a konstantní řetězce používané pro komunikační protokol a chybové zprávy. Definice těchto konstant zjednodušuje úpravy a správu kódu, protože hodnoty používané na více místech lze snadno měnit z jednoho místa.

3.4.8 Main

Hlavní třída `Main` slouží jako vstupní bod aplikace. Je zodpovědná za inicializaci serveru, zpracování konfiguračních souborů, a vytvoření socketů pro komunikaci s klienty. Třída obsahuje metody pro zavření klientova připojení, čtení konfigurace serveru, správu a zpracování připojení klientů, a kontrolu časových limitů klientů. Hlavní metoda této třídy spustí server, poslouchá příchozí připojení a spravuje klientovy vlákna pro zpracování zpráv.

3.4.9 MessageProcessor

Třída `MessageProcessor` je navržena k zpracování příchozích zpráv od klientů. Obsahuje metody pro zpracování různých typů zpráv (např. registrace, přihlášení, odhlášení, správa lobby, herní akce) a vrací odpověď na základě výsledku. Každá metoda přijímá operační

kód, obsah zprávy, a reference na `ClientInfo`, `ClientManager` a `LobbyManager` pro provedení potřebných akcí. Tato třída je klíčová pro rozhraní mezi klientem a serverem a zajišťuje správné zpracování a odpověď na různé typy požadavků.

3.4.10 Lobby

Třída `Lobby` reprezentuje herní místnost, ve které se hráči shromažďují a hrají. Obsahuje informace o lobby, jako je název, maximální počet hráčů, přítomnost hesla, správu hráčů a hru. Metody této třídy umožňují přidávání a odebírání hráčů, zahájení a ukončení hry, odesílání zpráv všem hráčům v lobby, a konverzi informací o lobby do textového formátu pro různé účely. `Lobby` je klíčová pro správu skupin hráčů a interakci mezi nimi v rámci hry.

3.4.11 LobbyManager

Třída `LobbyManager` slouží k správě herních místností (lobby) na serveru. Obsahuje neuspořádanou mapu (`unordered_map`) pro ukládání lobby podle jejich názvů. Třída nabízí metody pro vytváření nových lobby, přidávání hráčů do stávajících lobby, mazání lobby a konverzi informací o lobby do textové podoby. Metody pracují s mutexem pro zajištění bezpečnosti a konzistence dat ve vícevláknovém prostředí.

3.4.12 ClientManager

Třída `ClientManager` je zodpovědná za správu klientů a uživatelů připojených k serveru. Obsahuje seznam (`list`) všech připojených klientů a mapu (`unordered_map`) uživatelů. Tato třída poskytuje metody pro přidávání nových klientů, pokusy o přihlášení uživatelů, registraci nových uživatelů, odhlášení a opuštění lobby. Metody zpracovávají požadavky klientů a reagují na ně podle aktuálního stavu a pravidel hry.

3.4.13 BlackjackGame

Třída `BlackjackGame` představuje logiku a stav hry Blackjack. Obsahuje balíček karet (`vector<Card>`), mapu hráčů a jejich karet, a odkazy na související lobby a hráče. Třída zajišťuje inicializaci balíčku karet, míchání karet, rozdávání karet hráčům, a zpracování herních akcí jako jsou tahy a pasy. Dále umožňuje výpočet a deklaraci vítěze a správu stavu hry, včetně zasílání aktualizovaných informací o hře všem hráčům.

3.4.14 Vzájemná Interakce Tříd

V aplikaci BlackJack Server jsou třídy pečlivě navrženy tak, aby spolu efektivně interagovaly a poskytovaly koherentní a robustní řešení pro serverovou část hry Blackjack.

Main a MessageProcessor

Hlavní třída `Main` funguje jako neustále naslouchající uzel, který očekává příchozí zprávy od klientů. Jakmile je zpráva přijata, `Main` zkontroluje, zda odpovídá definovanému formátu komunikačního protokolu. Pokud zpráva neodpovídá, klient, který zprávu poslal, je

odpojen. V opačném případě je zpráva předána `MessageProcessoru`, který provádí požadovanou logiku podle obsahu zprávy, využívaje přitom metody z tříd jako `ClientManager`, `LobbyManager` a `BlackjackGame`.

Interakce s LobbyManagerem a ClientManagerem

Jakékoliv akce týkající se lobby, hráčů nebo klientů jsou spravovány bezpečnostními třídami "manažery". `LobbyManager` řídí vytváření, správu a mazání herních místností (lobby), zatímco `ClientManager` zajišťuje registraci, přihlášení a správu klientů a uživatelů. Tyto třídy poskytují abstrakci a zabezpečení pro manipulaci se stavem hry a uživateli.

BlackjackGame

Logika samotné hry Blackjack je implementována v třídě `BlackjackGame`. Tato třída spravuje herní průběh, včetně míchání karet, rozdávání, tahů hráčů a určení vítěze. `BlackjackGame` pracuje v těsné spolupráci s `Lobby`, kde každá lobby může mít asociovanou instanci hry.

3.5 Použité Knihovny a Verze Prostředí

Projekt BlackJack Server využívá následující C++ knihovny a funkcionality:

- Standardní knihovna C++ (STL) - pro základní funkce a datové struktury.
- Síťové knihovny: `<sys/socket.h>`, `<netinet/in.h>`, `<arpa/inet.h>` - pro práci se síťovými sokety.
- Chrono - pro manipulaci s časem a časovými intervaly.
- IOStream a FStream - pro práci s vstupem a výstupem a souborovým systémem.
- Stringstream - pro manipulaci s textovými daty.

3.6 Komunikace s Klienty

Server má schopnost aktivně komunikovat se klienty, což zahrnuje odesílání zpráv klientům, aby je informoval o změnách ve hře nebo v lobby. To je zvláště důležité pro udržení klientů informovaných o stavu hry, příchodu nebo odchodu hráčů a dalších důležitých událostech.

Metoda Paralelizace

Projekt BlackJack Server implementuje paralelizaci pomocí vláken (threads) v C++:

- Vlákna jsou vytvářena pro každého klienta, aby nezávisle zpracovávala příchozí požadavky.
- Síťová komunikace a zpracování požadavků klientů probíhá v oddělených vláknech.

- Pro synchronizaci a bezpečný přístup k sdíleným zdrojům je použit mutex.
- Periodická kontrola časových limitů klientů je prováděna v samostatném vlákně.

3.7 Požadavky na Překlad, Spuštění a Běh Aplikace

Pro sestavení a spuštění serverové aplikace Blackjack Server jsou potřebné následující komponenty a nastavení:

- Operační systém: Linux
- Kompilátor: g++ podporující standard C++17
- CMake: Minimální verze 3.22
- Knihovny: Standardní knihovna C++ (STL), síťové knihovny (`sys/socket.h`, `netinet/in.h`, `arpa/inet.h`), Chrono, IOStream, FStream, a Stringstream

Všechny závislosti musí být předem nainstalovány v systému, aby bylo možné projekt bez problémů sestavit.

3.8 Postup Překladu

Pro sestavení projektu je potřeba provést následující kroky:

3.9 Sestavení pomocí CMake

V kořenovém adresáři projektu spusťte CMake pro generování Makefile:

```
cmake .
```

3.9.1 Sestavení pomocí Makefile

Po úspěšném vygenerování Makefile sestavte projekt pomocí nástroje make:

```
make
```

Tento příkaz zkomiluje zdrojové soubory a vytvoří spustitelný soubor serveru. Je důležité, že se příkaz make musí spustit v adresáři, kde se nachází vygenerovaný Makefile.

3.9.2 Spuštění aplikace

Po sestavení projektu můžete spustit serverovou aplikaci následujícím příkazem:

```
./BlackjackServer
```

Tento příkaz spustí server, který bude naslouchat na předem definovaném portu v configs.txt pro připojení klientů.

4 BlackJack Client

4.1 Přehled

BlackJack Client je klientská část hry Blackjack, napsaná v Javě s využitím frameworku JavaFX. Aplikace implementuje MVC (Model-View-Controller) architekturu, což zajišťuje čisté oddělení logiky, uživatelského rozhraní a datové vrstvy.

4.2 Model-View-Controller Architektura

MVC architektura je návrhový vzor, který rozděluje aplikaci do tří hlavních komponent:

- **Model** - Reprezentuje data a logiku aplikace. Model je zodpovědný za správu dat, pravidel hry a komunikaci se serverem.
- **View** - Zajišťuje zobrazení informací uživateli. V aplikaci BlackJack Client jsou views definovány pomocí FXML a stylizovány CSS specifickým pro JavaFX.
- **Controller** - Slouží jako propojení mezi View a Modelem. Controller zpracovává vstupy od uživatele, aktualizuje model a odráží změny ve view.

4.3 Grafické Rozhraní

Uživatelské rozhraní aplikace je vytvořeno pomocí SceneBuilder, což umožňuje vizuální návrh layoutů ve formátu FXML. Stylizace UI je prováděna pomocí CSS, které je upraveno pro specifika JavaFX, poskytující atraktivní a interaktivní uživatelské rozhraní.

4.4 Vícevláknové Zpracování

Aplikace využívá vícevláknového zpracování pro efektivní běh a reaktivitu:

- **UI Thread** - Spravuje uživatelské rozhraní a interakci s uživatelem.
- **Příchozí Zprávy** - Oddělené vlákno pro naslouchání a zpracování zpráv od serveru.
- **Odchozí Zprávy** - Vlákno pro odesílání akcí a požadavků na server.
- **Pingy a Odpovědi** - Vlákno pro udržování síťového spojení a zpracování pingů.

4.5 Struktura Tříd

4.5.1 Kontrolery

V rámci MVC architektury jsou kontrolery zodpovědné za manipulaci s uživatelským rozhraním a reakci na uživatelské akce. Níže je uveden přehled hlavních kontrolerů a jejich funkcí v aplikaci:

EnterPasswordController

`EnterPasswordController` spravuje dialog pro zadání hesla, které je požadováno pro připojení k lobby nebo pro její smazání. Tento kontroler zachytává akce spojené s tlačítky pro potvrzení nebo zrušení a ověřuje zadané heslo.

GameController

`GameController` řídí hlavní herní obrazovku. Zobrazuje informace o hráčích, jejich karty a umožňuje hráčům interagovat s hrou prostřednictvím tlačítek pro hraní, odchod z lobby nebo zahájení hry.

GameEndController

`GameEndController` se stará o obrazovku ukončení hry. Zobrazuje výsledky hry, včetně seznamu hráčů, jejich konečných hodnot karet a určuje, kdo vyhrál. Umožňuje hráčům zavřít herní obrazovku a vrátit se do lobby.

LobbyCreationController

`LobbyCreationController` umožňuje hráčům vytvořit novou herní lobby. Poskytuje UI pro zadání názvu lobby, nastavení maximálního počtu hráčů a volbu, zda bude lobby chráněna heslem.

LobbyMenuController

`LobbyMenuController` je zodpovědný za obrazovku menu lobby, kde hráči mohou vidět seznam dostupných lobby a připojit se k nim nebo vytvořit novou lobby.

LoginController

`LoginController` řídí přihlašovací obrazovku, kde uživatelé mohou zadat své přihlašovací údaje pro vstup do aplikace nebo se zaregistrovat.

MainMenuController

`MainMenuController` spravuje hlavní menu aplikace, poskytuje hráčům přístup k různým funkcím, jako je připojení k hře, nastavení a odhlášení.

RegistrationController

`RegistrationController` zajišťuje registraci nových uživatelů, kde si mohou vytvořit nový účet pro přístup do hry.

4.5.2 FxContainer

`FxContainer` je kontejner pro JavaFX scény a okna. Obsahuje statické metody pro nastavení a získání aktuálních scén a okenních instancí. Třída zajišťuje, že přechody mezi různými částmi aplikace jsou spravovány centrálně.

4.5.3 MainContainer

`MainContainer` je centrální uchovávací místo pro sdílené zdroje aplikace. Spravuje uživatelské informace, fronty zpráv pro komunikaci s serverem a synchronizační mechanismy pro různé stavy aplikace. Třída využívá blokující fronty pro zpracování příchozích a odchozích zpráv, což umožňuje bezpečnou komunikaci mezi vlákny v mnohovláknové aplikaci. AtomicLong a volatile proměnné se používají pro sledování stavů jako čas poslední odezvy nebo stav připojení.

Třída MainContainer taktéž obsahuje metody pro nastavení a získání různých stavů uživatele, jako je to, zda je uživatel v lobby, ve hře, nebo v menu po skončení hry. Tento přístup umožňuje ostatním částem aplikace získávat informace o stavu a reagovat na změny. Metody jsou chráněny pomocí synchronizovaných bloků, což zajišťuje, že přístup k těmto sdíleným zdrojům je bezpečný a koordinovaný napříč různými vlákny.

4.5.4 FxManager

Třída `FxManager` slouží jako správce pro ovládání uživatelského rozhraní JavaFX. Umožňuje vytváření modálních oken, přepínání scén a dynamické načítání FXML souborů, které definují layouty uživatelského rozhraní. Metody této třídy umožňují snadnou manipulaci s grafickými prvkami aplikace, jako jsou okna pro registraci, přihlášení nebo hru.

4.5.5 Objects

ConnectionObject

`ConnectionObject` je třída obsahující síťové komponenty potřebné pro komunikaci s serverem. Zahrnuje soket, nástroje pro čtení a zápis dat a konfigurační objekt, který obsahuje nastavení aplikace.

DeserializedMessage

Třída `DeserializedMessage` reprezentuje deserializovanou zprávu. Uchovává informace o úspěchu zprávy, jejím obsahu, operačním kódu a určuje, zda se jedná o herní zprávu.

GameObject

`GameObject` je třída reprezentující herní objekt, který obsahuje seznam hráčů a vítězů hry. Umožňuje aktualizaci a správu informací o hráčích v kontextu hry.

GamePlayer

`GamePlayer` je třída popisující hráče ve hře. Obsahuje vlastnosti jako jméno uživatele, skóre, zda je na tahu, a seznam karet. Poskytuje metody pro aktualizaci a získání těchto vlastností.

Lobby

`Lobby` představuje herní lobby s atributy jako ID lobby, jméno, maximální počet hráčů a stav hry. Třída nabízí funkce pro správu uživatelů lobby a herního stavu.

LobbyManager

`LobbyManager` je statická třída spravující všechny lobby. Uchovává mapu a seznam lobby a poskytuje metody pro aktualizaci, přidávání nebo odebírání lobby. Také spravuje aktuálně vybrané lobby a synchronizuje informace o stavu hry v lobby.

User

Třída `User` reprezentuje uživatele aplikace s atributy jako uživatelské jméno, jméno, příjmení a online stav. Tato třída poskytuje metody pro nastavení a získání uživatelských informací, stejně jako pro vygenerování řetězců pro přihlášení nebo registraci.

4.5.6 ActionUtils

Třída `ActionUtils` poskytuje metody pro vykonávání různých akcí v aplikaci, jako je odhlášení, přihlášení, tvorba a správa lobby a herních akcí. Akce zahrnují odesílání specifických zpráv na server a zpracování odpovědí. Metoda `logout` odhlásí uživatele a vrátí ho na přihlašovací scénu. Metoda `login` zpracovává přihlášení a případné automatické přihlášení po přerušení internetového připojení. `createLobby` vytvoří novou lobby a odesílá odpovídající zprávu serveru. `actionLobby` umožňuje uživatelům vstoupit do nebo opustit lobby a `leaveLobby` zpracovává opuštění lobby a návrat do hlavního menu.

4.5.7 FxUtils

Třída `FxUtils` poskytuje sadu utilitních funkcí pro práci s uživatelským rozhraním JavaFX, jako jsou dialogová okna pro zobrazení upozornění a nastavení pozadí pro grafické komponenty. Mezi klíčové metody patří `createErrorAlert`, `createWarningAlert`, a `createInformationAlert`, které vytvářejí různé typy dialogových oken pro interakci s uživatelem. Dále třída obsahuje funkce pro zobrazování specifických upozornění v různých situacích, například když je pole pro heslo prázdné nebo dojde k odpojení uživatele. Metody `setBackgroundImage` aplikují obrázky na pozadí kontejnerů, jako jsou `VBox` a `BorderPane`, a pomocné funkce `applyValidation` a `showTooltip` se používají pro validaci vstupů a zobrazování tooltipů.

4.5.8 GameUtils

Třída `GameUtils` se zaměřuje na herní akce, jako je zahájení hry, předání tahu, nebo provedení tahu v hře. `startGame`, `passAction`, a `takeAction` jsou metody, které posílají herní příkazy na server a reagují na odpovědi. Pomocná metoda `evaluateGameAction` hodnotí herní akce a správně zpracovává odpovědi serveru. `endGame` zpracovává ukončení hry a aktualizuje stav hry a seznam vítězů.

4.5.9 ServerUtils

Třída **ServerUtils** obsahuje metody pro správu síťových operací, jako je připojení k serveru, odesílání periodických *ping* zpráv pro udržení aktivního spojení a reakce na *pong* zprávy pro potvrzení dostupnosti serveru. Dále zpracovává logiku pro opětovné připojení a obsahuje metody pro zastavení a spuštění posluchačů a plánovačů spojení.

4.5.10 Utils

Třída **Utils** poskytuje pomocné funkce pro zpracování zpráv, vytváření požadavků na server, serializaci a deserializaci dat, a další obecné úkony. Funkce jsou obvykle statické a slouží jako nástroje pro ostatní komponenty aplikace pro práci se síťovými komunikacemi.

4.5.11 FxManager

FxManager je třída odpovědná za správu grafického uživatelského rozhraní. Řídí změny scén, vytváří modální okna a obstarává přechody mezi různými částmi aplikace. Třída využívá JavaFX framework pro manipulaci s UI.

4.5.12 BlackJackApplication

Hlavní třída **BlackJackApplication** dědící od *Application* je vstupním bodem programu. Řídí inicializaci aplikace, připojení k serveru a zahajuje hlavní událostní smyčku JavaFX. Obsahuje metody pro spuštění a ukončení aplikace a udržuje hlavní stage.

4.5.13 SuperMain

Třída `SuperMain` slouží jako hlavní třída pro spuštění JavaFX aplikace a je řešením problému s javafx-maven-plugin. Třída obsahuje pouze jednu metodu `main`, která předává řízení aplikace třídě `BlackJackApplication`. Toto uspořádání umožňuje obejít některé technické potíže spojené s kompilací a spouštěním JavaFX aplikací skrze Maven a zajišťuje hladký start aplikace.

4.6 Komunikace se Serverem

Komunikace mezi klientem a serverem je realizována prostřednictvím definovaného síťového protokolu, přičemž klient posílá požadavky a přijímá odpovědi od serveru, které jsou následně zpracovány a zobrazeny v uživatelském rozhraní.

4.6.1 Síťová Komunikace

Klíčovým aspektem aplikace je síťová komunikace mezi klientem a serverem. Tento mechanismus je založen na standardních Java síťových knihovnách:

- **java.net.Socket** - pro TCP síťovou komunikaci.
- **java.io** - pro vstupní a výstupní streamy pro přenos dat.

Tyto knihovny jsou použity pro implementaci asynchronního poslouchání serverových zpráv a zpracování klientských požadavků.

4.7 Požadavky na sestavení a běh aplikace

Pro úspěšné sestavení a spuštění klientské aplikace BlackJack je potřeba mít nainstalované následující komponenty:

4.7.1 Prostředí

- Operační systém: Linux nebo Windows
- Java Runtime Environment (JRE) verze 11 nebo novější pro běh aplikace.
- Apache Maven, preferovaná verze 3.6.0 nebo novější, pro sestavení a balení projektu.
- JavaFX SDK kompatibilní s vaší verzí JRE pro běh grafických komponent.

4.7.2 Sestavení projektu

Pro sestavení projektu pomocí Maven proveďte následující kroky v terminálu nebo příkazové řádce:

```
mvn clean package
```

Tento příkaz provede čištění projektu (odstranění předchozích sestavení), sestaví projekt a vytvoří spustitelný balíček (JAR soubor)

4.8 Spuštění aplikace

Po úspěšném sestavení můžete aplikaci spustit pomocí následujícího příkazu, který využívá sestavený JAR soubor:

```
java -jar cesta/k/vasemu/sestavenemu.jar
```

Zkontrolujte, že cesta k sestavenému JAR souboru je správná. Tento příkaz by měl být spuštěn v prostředí, kde je dostupné JRE verze 11 nebo novější.

4.9 Specifické požadavky a řešení problémů pro Linux

Následující kroky popisují, jak připravit Linuxové prostředí pro sestavení a spuštění aplikace BlackJack Client a jak řešit běžné problémy spojené s JavaFX.

4.9.1 Instalace JRE a Maven na Linuxu

Pro spuštění aplikace je potřebné mít nainstalované JRE a Maven. Instalaci lze provést pomocí správce balíčků distribuce. Například pro Debian nebo Ubuntu:

```
sudo apt update
sudo apt install default-jre
sudo apt install maven
```

Ujistěte se, že verze JRE odpovídá požadavkům vaší aplikace.

4.9.2 Instalace JavaFX na Linuxu

JavaFX lze na Linuxu instalovat samostatně nebo jako součást JDK. Pro samostatnou instalaci použijte následující příkazy:

```
sudo apt update
sudo apt install openjfx
```

Toto nainstaluje nejnovější dostupnou verzi JavaFX kompatibilní s vaším systémem.

4.10 Řešení problémů Linux

Chyba 'gdk_is_x11_display (display)' failed může být způsobena konfliktem mezi JavaFX a používaným grafickým serverem. Pro vyřešení:

1. Ujistěte se, že máte nainstalovány všechny potřebné knihovny pro JavaFX a GTK, které jsou kompatibilní s X11.
2. Přepněte na používání X11 místo Wayland, pokud je to možné.
3. Spusťte aplikaci s přídavnými parametry pro Java, které vynutí použití starší verze GTK nebo softwarové vykreslování.
4. Nastavte proměnné prostředí GDK_BACKEND na x11 před spuštěním aplikace.

4.11 Specifické požadavky a instrukce pro Windows

Pro uživatele systému Windows zahrnují specifické požadavky na sestavení a spuštění aplikace instalaci následujících komponent:

4.11.1 Instalace JRE a Maven

- JRE: Java Runtime Environment je nutné pro běh Java aplikací. Stáhněte si odpovídající verzi JRE z oficiálních stránek Oracle nebo OpenJDK.
- Maven: Apache Maven je nástroj pro správu a sestavení projektů v Javě. Stáhněte a nainstalujte Maven z oficiálních stránek Apache Maven.

4.11.2 Instalace JavaFX

- JavaFX SDK: Pro použití grafických komponent JavaFX v aplikaci je potřeba stáhnout a nainstalovat JavaFX SDK kompatibilní s verzí JRE. JavaFX SDK můžete stáhnout z webu Gluon nebo přímo z OpenJFX.

4.11.3 Konfigurace a spuštění aplikace

Po instalaci výše uvedených komponent můžete aplikaci sestavit a spustit pomocí stejných příkazů mvn clean package a java -jar, jak je uvedeno výše.

5 Závěr

5.1 Závěr BlackJack Serveru

Vývoj BlackJack Serveru představoval významný krok v oblasti implementace serverových aplikací. Podařilo se vytvořit stabilní, vícevláknovou aplikaci. Výzvy spojené s bezpečností, synchronizací a správou klientů byly úspěšně zvládnuty díky pečlivému návrhu architektury a efektivnímu využití C++ knihoven a standardů.

Bezpečnost serveru byla klíčovým aspektem, přičemž se podařilo zajistit odolnost proti nevalidním zprávám a potenciálním bezpečnostním hrozbám. Budoucí rozvoj serveru by mohl zahrnovat rozšíření funkcionalit, včetně možnosti hraní proti dealerovi, další optimalizace výkonu a implementace dalších pokročilých herních funkcí. Celkově byla práce na serveru úspěšná a poskytla pevný základ pro budoucí rozšiřování a vylepšení.

5.2 Závěr BlackJack Klienta

Vývoj klientské části hry BlackJack byl úspěšným projektem, který demonstroval možnosti a flexibilitu JavaFX ve vytváření interaktivního uživatelského rozhraní. Při implementaci byla dodržena architektura MVC (Model-View-Controller) a základní principy objektově orientovaného programování (OOP). Klient nabízí intuitivní a uživatelsky přívětivé rozhraní, což zajišťuje plynulý a příjemný herní zážitek.

Efektivní využití JavaFX ve vývoji UI umožnilo vytvořit esteticky přitažlivé a funkční prostředí. Klientská aplikace je stabilní, dobře strukturovaná a umožňuje snadnou navigaci a interakci během hry. Celkově lze říci, že vývoj klienta byl úspěšný a plně odpovídá požadavkům a očekáváním.

Navzdory úspěchu existují oblasti pro další zlepšování, včetně optimalizace výkonu, rozšíření funkčnosti uživatelského rozhraní a integrace dodatečných funkcí pro zvýšení uživatelského komfortu a interaktivity.

5.3 Celkový závěr semestrální práce

V rámci projektu se nám podařilo vytvořit stabilní a vzájemně propojené aplikace - server a klient pro hru Blackjack. Hlavním úspěchem byla implementace spolehlivých mechanismů pro obnovu spojení. V případě odpojení serveru klient aktivně usiluje o obnovení připojení a snaží se vrátit k předchozímu stavu hry. Podobně, když klient ztratí spojení, server adekvátně informuje ostatní hráče v lobby a řádně řeší konec hry, pokud je to potřebné.

Vývoj byl podpořen rozsáhlou dokumentací, která detailně popisuje komunikační protokol a interakce mezi serverem a klientem. Tento přístup umožňuje snadné přidávání nových funkcí a optimalizací, stejně jako nezávislý vývoj klienta nebo serveru. Celkově projekt představuje pevný základ pro další rozvoj.