# 

12.12.2024 Tomáš Kopelent (A23B0120P)

**Semestrální práce z předmětu KIV/UPS**

Hra piškvorky nad protokolem TCP

**Západočeská univerzita v Plzni**

**Fakulta aplikovaných věd**

# Obsah

[Obsah 1](#_Toc181257169)

[Zadání 2](#_Toc181257170)

[Analýza úlohy 3](#_Toc181257171)

[Implementace 4](#_Toc181257172)

[1. Soubor fat.h a fat.c 4](#_Toc181257173)

[2. Soubory commands.h a commands.c 5](#_Toc181257174)

[3. Soubor constants.h 5](#_Toc181257175)

[4. Soubory input\_controller.h a input\_controller.c 5](#_Toc181257176)

[5. Soubory main.h a main.c 5](#_Toc181257177)

[Uživatelská příručka 6](#_Toc181257178)

[První spuštění programu 6](#_Toc181257179)

[Dostupné příkazy 6](#_Toc181257180)

[Závěr 7](#_Toc181257181)

# Popis hry

Rozhodl jsem se implementovat hru piškvorky. Klientskou část jsem naprogramoval v jazyce Java (verze 17) a serverovou v jazyce C. Server přijímá klienty, kteří s ním komunikují příslušným protokolem a vytváří pro ně herní místnosti, tak, že spojí vždy dva čekající hráče do jedné herní místnosti. Hrací deska má rozměr 5x5 políček a vyhrává hráč, který jako první dosáhne spojení čtyř znaků.

Pokud se jeden z hráčů v průběhu hry odpojí a nevrátí se zpět do hry (v požadovaném čase), je vyhlášen za vítěze připojený hráč. Dojde-li k zaplnění hrací plochy, aniž by některý z hráčů dokázal spojit čtyři znaky, pak hra končí remízou. Po skončení hry je vždy oběma hráčům (jsou-li připojeni) nabídnuto opustit hru nebo spustit další hru.

# Popis protokolu

Klientská a serverová část aplikace spolu komunikují přes protokol TCP. Zprávy, které si vzájemně posílají jsou shrnuty v následující tabulce:

|  |  |  |
| --- | --- | --- |
| **ZPRÁVY VYSKYTUJÍCÍ SE CHRONOLOGICKY ZA SEBOU** | | |
| **Zpráva** | **Odesílatel** | **Význam** |
| LOGIN;<name>\n | klient | Přihlašovací zpráva od nového klienta |
| LOGIN;<name>\n | Server | Odpověď na přihlášení klienta |
| WANT\_GAME;\n | Klient | Klient chce novou hru (není ve hře) |
| WANT\_GAME;<char>\n | Server | Odpověď pro klienta se znakem, který bude mít klient přidělen ve hře |
| START\_GAME;<opp\_name>;  <opp\_char>;<my\_turn >\n | Server | Oznámení startu hry  <opp\_name> = jméno protihráče  <opp\_char> = znak protihráče  <my\_turn > = 1 (hráč je na tahu), 0 (hráč není na tahu) |
| MOVE;x;y\n | Klient | Odeslání tahu klienta |
| * MOVE;0;x;y\n * MOVE;5;0;0\n * MOVE;6;0;0\n * MOVE;7;0;0\n * MOVE;8;0;0\n | Server | * Tah prošel validací (návratová hodnota 0 a souřadnice tahu) * Hra nenalezena * Hráč není na tahu * Neplatný tah * Zabrané pole |
| OPP\_MOVE;x;y\n | Server | Odeslání validního tahu protihráči |
| * GAME\_STATUS;<name>\n * GAME\_STATUS;DRAW\n * GAME\_STATUS;  OPP\_DISCONNECTED\n | Server | * Vyhrál hráč <name> * Hra skončila remízou * Hra skončila protože se jeden z hráčů odpojil |
| **ZPRÁVY, KTERÉ SE VYSKYTUJÍ NECHRONILOGICKY** | | |
| **Zpráva** | **Odesílatel** | **Význam** |
| LOGOUT;\n | Klient | Klient se vědomě odpojuje |
| OPP\_DISCONNECTED;\n | Server | Zpráva pro hráče, že se jeho protihráč odpojil |
| * OPP\_DISCONNECTED;WAIT\n * OPP\_DISCONNECTED;NOT\_WAIT\n | Klient | * Čeká na odpojeného protihráče * Nečeká na odpojeného protihráče |
| PING;\n | Server | Kontrolní ping zpráva klientovi |
| PONG;\n | Klient | Odpověď klienta na ping zprávu |
| RECONNECT;<board XO>;  <player\_name>;  <opponent\_name>;  <player\_char>\n | Server | Obnovovací zpráva po připojení odpojeného hráče  <board XO> = hrací plocha velikosti n krát n, kde jsou v jedné řádce položky X, O, „mezera“  <player\_name> = jméno hráče, kterému je posílána obnovovací zpráva  <opponent\_name> = jméno protihráče  <player\_char> = hrací znak hráče |

# Implementace serverové části

VLÁKNA + KNIHOVNY + JAZYK

## Konfigurační soubor config.h

Tento soubor obsahuje všechny konstanty, které jsou v programu využívány. Dovoluje tedy měnit konstanty přehledně na jednom místě. Dále jsou zde i struktury *client* (reprezentuje klienta připojeného na server)*, game* (reprezentuje rozehranou hru mezi dvěma klienty)*, server\_address* (reprezentuje dvojici IP adresa a port pro efektivnější zpracování programových argumentů).

## Modul main.c

Obsahuje vstupní bod programu, zpracovává vstupní parametry (ip adresu a port, pokud jsou zadány). Dále je zde vytvořeno vlákno pro běh serveru a vlákno pro zasílání ping zpráv klientům. Nosnou metodou tohoto modulu je metoda *run\_server()*, která inicializuje server a obsahuje nekonečnou while smyčku pro příjem nových klientů, žádajících o připojení.

## Modul communication.h a communication.c

Tento modul obsahuje funkce pro příjem a komunikaci připojeného klienta se serverem. Metoda *receive\_messages()* obsahuje nekonečnou while smyčku, která odchytává zprávy poslané od klienta a v závislosti na typu zprávy (rozhoduje metoda *serve\_message()*) příslušně zareaguje. V modulu jsou také funkce pro zaslání zpráv vyhovujících protokolu klientovi.

## Model client\_manager.h a client\_manager.c

Modul obsahuje globální proměnnou *clients*, což je pole ukazatelů na strukturu *client*, velikosti MAX\_CLIENTS (konstanta měnitelná v config.h). Do této globální proměnné jsou pomocí funkce *add\_client()* přidáváni nově připojení členové. Aby byl zajištěný bezpečný přístup k proměnné *clients* napříč vlákny, obsahuje modul i mutex *clients\_mutex*. Při práci s kritickou sekcí tedy vždy dojde nejdříve k uzamčení mutexu a po provedení operace k jeho odemčení. Dále jsou v modulu metody a funkce pro nastavení nebo úpravu dat klientů.

## Model game\_manager.h a game\_manager.c

Zde nalezneme globální proměnnou *games*, která je polem ukazatelů na strukturu *game* a má velikost MAX\_GAMES. Přístup ke sdílené paměti chrání mutex *mutex\_games*. V modulu se nachází metody a funkce pro založení a zrušení hry nebo funkce pro výpis a získání konkrétní hry (ať už pro získání hry, kterou daný klient hraje nebo hry podle jejího vytvořeného ID – je hře přiděleno při jejím vytvoření).

## Model game\_logic.h a game\_logic.c

Zde je obsažena všechna logika hry piškvorky. Po každém tahu je volána funkce z tohoto modulu *validate\_move()*, která nejdříve vyhodnotí, zda je hráč na tahu a zda jeho tah má smysl v kontextu hry a dále, pokud byl tah validní, zkontroluje, zda v důsledku tahu nedošlo k dosažení výherní délky a ukončení hry (pokud ano nastaví příslušné atributy struktury *game*, která je uložená v globální proměnné *games* a navrátí příslušnou návratovou hodnotu).

# Implementace klientské části

Pro implementaci klientské části jsem se rozhodl použít Javu verze 17. Při návrhu klientské části semestrální práce jsem vycházel z architektonického vzoru MVC (model-view-controller). Cílem bylo oddělit logiku a zpracování dat od uživatelského rozhraní. Pro síťové připojení jsem použil knihovny Net a Io. GUI využívá knihovnu Java Swing a Awt.

## Třída Constants

Obsahuje statické konstanty, které jsou použité jinde v kódu. Je tedy možné stejně jako u serveru měnit hodnoty konstant na jednom místě.

## Třídy Player a TicTacToeModel

Obsahují modely, které klientská aplikace využívá. *Player* obsahuje pouze atributy *name* (jméno hráče) a *playerChar* (znak hráče X nebo O). TicTacToeModel obsahuje jako atributy *myPlayer* a *opponentPlayer* (oba typu *Player*) a dále dvourozměrné pole *board* typu char a velikosti TIC\_TAC\_TOE\_SIZE x TIC\_TAC\_TOE\_SIZE, které reprezentuje hrací plochu. Jsou zde navíc metody pro nastavení atributů a aktualizaci hrací plochy dle serverem validovaného tahu.

## Třída TicTacToeView

## Třída TicTacToeController

## Třída ServerClient

# Překlad programu

# Závěr