

Semestrální práce z předmětu KIV/UPS

## Síťová hra Větší bere

Max Nonfried  
A19B0601P

31. ledna 2022

# Obsah

<b>1</b>	<b>Zadání</b>	<b>3</b>
<b>2</b>	<b>Pravidla hry</b>	<b>4</b>
<b>3</b>	<b>Komunikační protokol</b>	<b>5</b>
3.1	Přijímání zpráv - omezení vstupních hodnot . . . . .	5
3.2	Seznam zpráv . . . . .	6
3.2.1	Připojení do lobby . . . . .	6
3.2.2	Připojení do hrací místnosti - začátek hry . . . . .	6
3.2.3	Průběh kola . . . . .	6
3.2.4	Konec hry - odpojení z hrací místnosti, návrat do lobby . . .	7
3.2.5	Opětovné připojení hráče . . . . .	7
3.2.6	Ping . . . . .	7
3.2.7	Odpojení klienta . . . . .	7
<b>4</b>	<b>Popis implementace serveru</b>	<b>8</b>
4.1	Použité prostředky . . . . .	8
4.2	Popis jednotlivých modulů . . . . .	8
4.2.1	definitions.h . . . . .	8
4.2.2	main.c . . . . .	8
4.2.3	communication.c . . . . .	8
4.2.4	recv_msg_handlers.c . . . . .	9
4.2.5	connection_stuff.c . . . . .	9
4.2.6	player_functions.c . . . . .	9
4.2.7	game_functions.c . . . . .	9
4.3	Popis paralelizace . . . . .	9
<b>5</b>	<b>Popis implementace klienta</b>	<b>11</b>
5.1	Použité prostředky . . . . .	11
5.2	Popis jednotlivých modulů . . . . .	11
5.2.1	Main.java . . . . .	11
5.2.2	Gui.java . . . . .	11
5.2.3	GuiReactions.java . . . . .	11
5.2.4	MouseListener.java . . . . .	11
5.2.5	MyWindowListener.java . . . . .	11
5.2.6	Communication.java . . . . .	11
5.2.7	ListeningThread.java . . . . .	12

5.2.8	RecvMsgThread.java . . . . .	12
5.2.9	GameHandler.java . . . . .	12
5.2.10	PingThread.java . . . . .	12
5.2.11	ReconnectThread.java . . . . .	12
5.3	Popis paralelizace . . . . .	12
<b>6</b>	<b>Spuštění aplikace</b>	<b>13</b>
6.1	Server . . . . .	13
6.2	Klient . . . . .	13
<b>7</b>	<b>Závěr</b>	<b>14</b>

# Zadání

Zadáním této semestrální bylo vytvořit síťovou hru pro více hráčů formou server – klient. Podmínkou bylo, že server bude napsán v programovacím jazyce C (případně v C++) a spuštěn na operačním systému Linux. Klient musí být napsán v programovacím jazyce Java (případně C#, Unity, Kotlin) a musí být určen jak pro operační systém Windows, tak i Linux.

Podrobný popis zadání je v dokumentu *PozadavkyUPS.pdf*.

Já jsem si vybral tahovou hru Větší bere, jejíž pravidla jsem podle požadavků semestrální práce upravil, viz kapitola 2. Server jsem napsal v programovacím jazyce C, klient je napsán v jazyce Java.

# Pravidla hry

Hra se hraje s mariášovými kartami (tj. 32 karet). Hra je pro dva hráče. Každému hráčovi se rozdají 4 karty. Každé kolo si hráč lízá jednu kartu. Hráč, který je na tahu, vyloží jednu kartu. Následně druhý vyloží také jednu kartu. Hráč, který má kartu s vyšší hodnotou, získává 1 bod. V případě stejných hodnot vyložených karet získávají bod oba dva hráči. Vyhrává hráč, který nasbírá nejvíc bodů.

# Komunikační protokol

Komunikační protokol definuje zprávy, které slouží ke komunikaci mezi serverem a klientem. Návaznost zpráv je znázorněna na diagramu v souboru *navaznost\_zprav.png*.

Formát zprávy:

```
"hlavička|jaké funkcionality se zpráva týká|samotná zpráva\n"
```

Konkrétní příklad:

```
"vb|round|playedcard|<played_card>\n"
```

Použité výrazy:

- "vb"- hlavička zprávy,
- "|"- oddělovač částí zprávy,
- "\n"- konec zprávy.

Základní typy zprávy, tj. čeho se daná zpráva týká:

- "nick"- identifikuje zprávy týkající se přezdívky,
- "connection"- identifikuje zprávy týkající se spojení,
- "round"- identifikuje zprávy týkající se kola hry,
- "game"- identifikuje zprávy týkající se hry celkově.

## 3.1 Přijímání zpráv - omezení vstupních hodnot

Zprávy jsou přijímány po jednom znaku (po jednom bytu). Přijatý znak je uložen jako datový typ `integer`. Následně probíhá kontrola, zda je ASCII hodnota přijatého znaku v intervalu `[0,127]`. U prvních tří znaků se také kontroluje, zda odpovídají prvním třem znakům definovaných protokolem, tj. "v", "b" a "|". Pokud se třikrát přijme nevalidní znak, je přijímací cyklus přerušen a klient odpojen. Pokud je zpráva v pořádku, vypíše se do konzole a vytvoří se vlákno pro její zpracování. Do vlákna je předána ve formě `*char` u serveru, resp. ve formě `string` u klienta.

Po rozdělení zprávy na části oddělené "|" se překontroluje, zda zpráva obsahuje hlavičku. V sérii podmínek se rozhodne, čeho se zpráva týká (nick, connection, round, game) a je předána podrobnější sérii podmínek pro daný typ zprávy. Pokud v kterékoliv fázi zpráva nevyhovuje protokolu, či je poslána v nevhodný čas (např. hráč není na tahu), server i klient to pozná. V případě serveru si server zaznamená u klienta nevalidní zprávu a po třech nevalidních zprávách klienta odpojí, klient takovou zprávu ignoruje.

## 3.2 Seznam zpráv

C představuje klienta, S server, "<>" značí proměnnou.

### 3.2.1 Připojení do lobby

C: "vb|nick|<nick>\n"- poslání přezdívky (pokud je v pořádku, hráč pokračuje do lobby).

S: "vb|nick|ok\n"- potvrzení, že přezdívka je v pořádku.

S: "vb|nick|duplicate\n"- odmítnutí přezdívky (již existuje).

### 3.2.2 Připojení do hrací místnosti - začátek hry

C: "vb|game|play\n"- žádost o připojení do hry.

S: "vb|game|full\n"- počet herních místností je plný.

S: "vb|game|wait\n"- čekej, hledám protihráče.

S: "vb|game|wait|error\n"- protihráč nenalezen.

S: "vb|game|prepare|<opponents\_nickname>|<card1>|<card2>|<card3>|<card4>\n"- protihráč nalezen, posílám informace potřebné k započetí hry.

S: "vb|game|play|<active\_player>\n"- hra začíná, na tahu je "<active\_player>".

### 3.2.3 Průběh kola

C: "vb|round|playedcard|<played\_card>\n"- klient posílá kartu, kterou zahrál.

S: "vb|round|oppplayedcard|<opponents\_played\_card>\n"- protihráč zahrál tuto kartu.

S: "vb|round|roundresult|nickname|<nickname>|<my\_points>|<opponents\_points>\n"- výsledek kola, proměnné: vítěz kola, moje body, body protihráče.

S: "vb|round|roundresult|tie|<my\_points>|<opponents\_points>\n"- výsledek kola, remíza, proměnné: moje body, body protihráče.

S: "vb|round|lastcard\n"- poslední karta v balíčku karet (příkaz pro zneviditelnění balíčku karet).

S: "vb|round|start|<card\_to\_draw>|<active\_player>\n"- začátek kola, proměnné: jakou kartu si má hráč líznout, kdo je na tahu.

### 3.2.4 Konec hry - odpojení z hrací místnosti, návrat do lobby

Reakce na zprávu „vb|round|playedcard|<played\_card>“ od klienta, když zahrál poslední kartu.

S: "vb|game|winner|forced|<winner\_nickname>\n"- neplánované ukončení hry (protihráč je nedostupný déle než 1 minutu, protihráč posílal serveru nevalidní zprávy).

S: "vb|game|winner|<winner\_nickname>\n"- plánované ukončení hry.

S: "vb|game|end|<number\_of\_won\_games>\n"- konec hry, příkaz pro návrat do lobby, proměnná: celkový počet vyhraných her.

### 3.2.5 Opětovné připojení hráče

S: "vb|connection|opppdisconnected\n"- zpráva hráčovi, že se protihráč je nedostupný. Pokud se protihráč nepřipojí do 1 minuty, je hra ukončena.

S: "vb|connection|oppconnected\n"- zpráva hráčovi, že se protihráč opět připojil do hry.

C: "vb|connection|reconnect|<nickname>\n"- žádost klienta o reconnect, identifikuje se přezdívkou.

S: "vb|connection|reconnect|<card1>|<card2>|<card3>|<card4>|<turn>|<my\_points>|<opp\_points>|<played\_card>\n"- informace pro klienta po znovu připojení, pokračuje ve hře. Proměnná «played\_card» je protihráčem zahraná karta.

S: "vb|connection|reconnect|lobby|<number\_of\_won\_games>\n"- informace pro klienta po znovu připojení, příkaz pro zobrazení lobby stránky.

### 3.2.6 Ping

C: "vb|connection|ping\n"- pingovací zpráva serveru.

S: "vb|connection|ping|ok\n"- odpověď na pingovací zprávu.

S: "vb|connection|ping\n"- pingovací zpráva klientovi.

C: "vb|connection|ping|ok\n"- odpověď na pingovací zprávu.

### 3.2.7 Odpojení klienta

C: "vb|connection|disconnect\n"- klient se odpojil.



# Popis implementace serveru

## 4.1 Použité prostředky

Server je naprogramován v programovacím jazyce C.

## 4.2 Popis jednotlivých modulů

### 4.2.1 definitions.h

Hlavičkový soubor, který je importovaný do všech céčkových souborů. Obsahuje importované knihovny, definuje konstanty, deklaruje globální proměnné a importované hlavičkové soubory jednotlivých céčkových souborů.

### Popis důležitých struktur

Struktura `game` obsahuje proměnné potřebné k zajištění chodu hry. Hry jsou uloženy v poli ukazatelů `game **game_pointers`. Struktura `player` obsahuje proměnné potřebné k zajištění chodu hry. Jako unikátní klíč je určena přezdívka. K vyhledání hráče se ovšem často používá socket. Hráči jsou uloženi v poli ukazatelů `player **players`.

### 4.2.2 main.c

Soubor obsahuje funkci `main()`, funkci pro kontrolu argumentů, funkci pro inicializaci proměnných. Pokud příprava proběhne v pořádku, ve funkci `main()` se vytvoří vlákno pro pingování a spustí se funkce `communicate(...)`, která obsahuje nekonečný cyklus pro přijímání spojení a zpráv.

### 4.2.3 communication.c

Soubor obsahuje funkce potřebné ke komunikaci serveru s klienty. Vytvoření serverového socketu, přijímání spojení a přijímání zpráv zajišťuje funkce `communicate(...)`. Po přijetí zprávy se vytvoří vlákno pro její zpracování (funkce `serve_communicate_thread(...)`). Zpráva se rozdělí funkcí `str_split(...)` a předá se funkci `handle_recv_msg(...)`, která obsahuje sérii podmínek pro rozdělení zprávy podle typu.

Soubor dále obsahuje funkci pro odeslání zprávy `send_msg(...)`; funkci `deal_with_invalid_msg(...)`, která každému klientovi počítá nevalidní zprávy. Zavírání socketů probíhá v nekonečné smyčce funkce `communicate(...)` voláním funkce `close_sockets()`, která zavře sockety, které byly určeny k zavření. Funkce `add_socket_to_close(...)` označí socket jako socket určený k zavření.

#### 4.2.4 `recv_msg_handlers.c`

Soubor obsahuje funkce pro zpracování konkrétních typů zpráv, tj. `handle_nick_msgs(...)`, `handle_connection_msgs(...)`, `handle_game_msgs(...)`, `handle_round_msgs(...)`.

Dále obsahuje pomocné funkce pro zajištění chodu a logiky hry.

#### 4.2.5 `connection_stuff.c`

Soubor obsahuje funkci `ping(...)`, která běží v samostatném vlákně a v určitých časových intervalech rozesílá klientům ping zprávy. Pokud klient neodpoví do určitého času, funkce ho označí za odpojeného a vytvoří vlákno `handle_disconnected_player_thread(...)`, které čeká na znovu připojení klienta. Pokud klient během této čekací doby pošle `reconnect` zprávu, funkce `reconnect_player(...)` zajistí jeho znovu připojení.

#### 4.2.6 `player_functions.c`

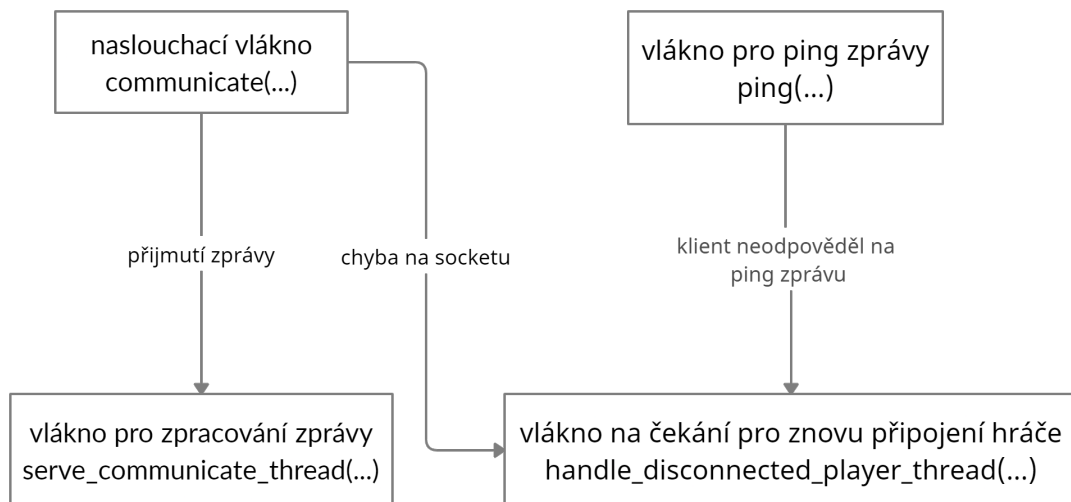
Soubor obsahuje funkce týkající se přímo hráče (struktury `player`), tedy např. `create_player(...)`, `get_player_index(...)`, `delete_player(...)`.

#### 4.2.7 `game_functions.c`

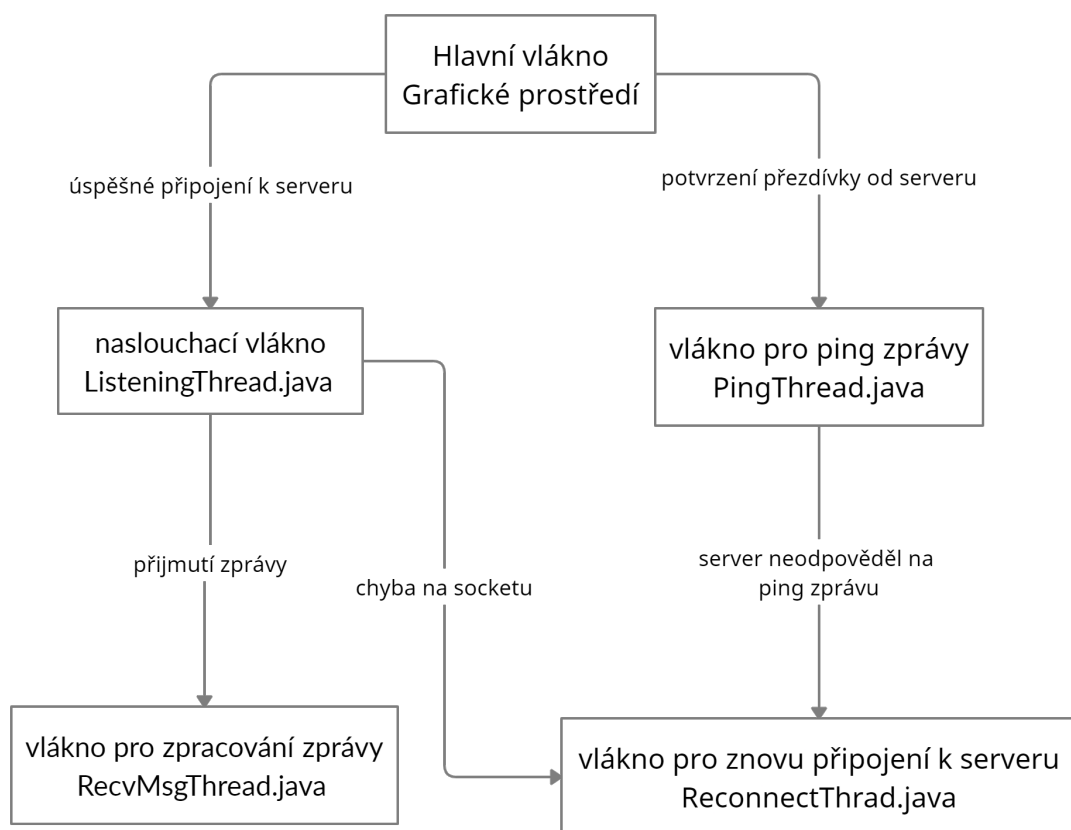
Soubor obsahuje funkce týkající se přímo hry (struktury `game`), tedy např. `create_new_game(...)`, `prepare_game(...)`, `get_game_index(...)`.

### 4.3 Popis paralelizace

Souběh vláken je znázorněn na obrázku 4.1.



Obrázek 4.1: Graf vláken na serveru



Obrázek 4.2: Graf vláken na klientské straně

# Popis implementace klienta

## 5.1 Použité prostředky

Klient je naimplementován v programovém jazyce Java. Pro grafické prostředí je použit balíček `javax.swing`.

## 5.2 Popis jednotlivých modulů

### 5.2.1 `Main.java`

Soubor obsahuje metodu `main(...)`. Ta vytvoří instance tříd, zajistí jejich přípravu metodami `prepare...(...)` a zobrazí grafické prostředí.

### 5.2.2 `Gui.java`

Soubor obsahuje veškeré prvky grafického prostředí. Po vytvoření instance se musí zavolat metoda `prepareGui(...)`, která zajistí vytvoření a nastavení všech prvků grafického prostředí. Grafické prostředí obsahuje tři panely `openPanel`, `lobbyPanel`, `gamePanel` mezi kterými je podle potřeb přepínáno.

### 5.2.3 `GuiReactions.java`

Soubor zajišťuje reakce po kliknutí na tlačítka.

### 5.2.4 `MyMouseListener.java`

Soubor zajišťuje reakci po kliknutí na kartu při hře.

### 5.2.5 `MyWindowListener.java`

Soubor zajišťuje korektní zavření při zavření grafického prostředí křížkem.

### 5.2.6 `Communication.java`

Soubor obsahuj metodu `connect()` pro vytvoření socketu, připojení k serveru a inicializování čtecího a psacího bufferu. Metoda `sendMsg(...)` zajišťuje posílání

zpráv. Dále soubor obsahuje proměnné, ve kterých jsou uloženy informace k připojení.

### 5.2.7 `ListeningThread.java`

Soubor obsahuje metodu `run(...)`, která běží v samostaném vlákne. V metodě běží nekonečný cyklus pro přijímání zpráv. Když se přijme nová zpráva, vytvoří se vlákno pro její zpracování. Pokud se na soketu vyskytne chyba, vytvoří se vlákno pro znovu připojení k serveru. Pokud se přijme nevalidní znak, zavře se socket, ukončí se ping vlákno a vlákno se ukončí. Klient se následně musí ručně připojit.

### 5.2.8 `RecvMsgThread.java`

Soubor obsahuje metodu `run(...)`, která běží v samostaném vlákne. Metoda má sérii podmínek pro rozdělení zpráv do čtyř typů (`nick`, `connection`, `round`, `game`). Podle typu zprávy se předá určité metodě pro další zpracování.

### 5.2.9 `GameHandler.java`

Soubor obsahuje metody pro zpracování jednotlivých typů zpráv. Zprávy jsou metodám předány od `RecvMsgThread.java`. Tento modul zajišťuje logiku a chod hry.

### 5.2.10 `PingThread.java`

Soubor zajišťuje posílání ping zpráv serveru v určitých časových intervalech. Pokud server do daného času neodpoví, vytvoří se vlákno `ReconnectThread.java` pro znovu připojení k serveru.

### 5.2.11 `ReconnectThread.java`

Soubor zajišťuje znovu připojení klienta k serveru v samostaném vláknu. Po spuštění vlákna se klient v časových intervalech pokouší znovu připojit k serveru. Vlákno se ukončí po úspěšném znovu připojení, nebo po doběhnutí časového limitu.

## 5.3 Popis paralelizace

Souběh vláken je znázorněn na obrázku 4.2.

# Spuštění aplikace

## 6.1 Server

Program se spouští se čtyřmi parametry. maximální počet připojených klientů; maximální počet her; port, na kterém bude server naslouchat; IP adresa, na které bude server naslouchat. Program se překládá nástrojem Makefile. Pro správný překlad je zapotřebí překladač gcc verze 10.2.1. Příkaz pro přeložení:

```
make
```

Příkaz pro spuštění (parametry jsou uvedeny v <>):

```
./main <maximální počet hráčů> <maximální počet her> <port> <IP  
adresa>
```

## 6.2 Klient

Správný chod klienta zaručuje Java verze 11. Klient je k dispozici jako spustitelný soubor JAR. Spouští se stejným příkazem jak na operačním systému Windows, tak i na operačním systému Linux:

```
java -jar <ups_client>.jar
```

# Závěr

Server i klienta se mi podařilo s drobnými nedostatky naimplementovat. Vzhledem k tomu, že to byl můj první větší programátorský projekt, jsem získal mnoho nových zkušeností. Seznámil jsem se s implementací grafického prostředí, s používáním vláken a organizováním jejich souběhu a s využitím socketů (protokolem TCP/IP).

Hlavní nedostatek a tedy prostor pro vylepšení se nachází v serveru. Vyhledávání hráčů by bylo ideální naimplementovat podle ID, nikoliv podle socketů, přezdívky. Předešlo by se tím různým komplikacím při implementaci.

# Seznam obrázků

4.1	Graf vláken na serveru . . . . .	10
4.2	Graf vláken na klientském . . . . .	10