

**Hra Pexeso n hráčov**

SEMESTRÁLNA PRÁCA

Vypracoval: **Martin Uhrina, Matej Puchý**

Študijná skupina: **5ZYP31, 5ZYR31**

Predmet: **Princípy operačných systémov**

Cvičiaci: **doc. Ing. Miroslav Kvaššay PhD.**

Obsah

[1. Programátorská dokumentácia 3](#_Toc155615318)

[1.1. Štruktúra projektu 3](#_Toc155615319)

[1.1.1. Pexesso 3](#_Toc155615320)

[1.1.2. Player 3](#_Toc155615321)

[1.1.3. Client 3](#_Toc155615322)

[1.1.4. Hrac1 4](#_Toc155615323)

[1.1.5. Hrac2 4](#_Toc155615324)

[1.1.6. MainServer 4](#_Toc155615325)

[1.2. Sockety 5](#_Toc155615326)

[1.3. Vlákna 5](#_Toc155615327)

[1.4. Synchronizačný problém 5](#_Toc155615328)

[2. Používateľská dokumentácia 6](#_Toc155615329)

Zoznam obrázok

[Obrázok 1 Úvodné zobrazenie zo strany klienta 6](#_Toc155649811)

[Obrázok 2 Po zadaní správy hraj 7](#_Toc155649812)

[Obrázok 3 Zadanie znaku na potvrdenie 7](#_Toc155649813)

[Obrázok 4 Zadanie súradníc 8](#_Toc155649814)

[Obrázok 5 Odhalenie znakov 8](#_Toc155649815)

[Obrázok 6 UML diagram 9](#_Toc155649816)

# Programátorská dokumentácia

## Štruktúra projektu

### Pexesso

* stará sa o logiku pexesa, čiže inicializuje objekt hry s daným počtom riadkov a stĺpcov, nastavuje náhodne umiestnené hodnoty na hracej doske
* void initialization()- inicializuje hraciu dosku náhodne premiešanými hodnotami
* bool makeGuess(int row1, int col1, int row2, int col2)-porovnáva 2 vzory na hracej doske na základe zadaných súradníc, ak sú zhodné pridá ich do nájdených párov ak nie nechá ich v stave v akom sa nachádzali
* void characters()-vypíše aktuálny stav hracej dosky na konzolu
* bool allPairsFound()-metóda, ktorá overí či boli všetky páry(dvojice) nájdené
* void revealPair(int row, int col)- odhalí vzor na hracej doske na základe zadaných súradníc
* char getCharAt(int row, int col) const – vráti znak hracej plochy na danej pozícii na základe stavu hry
* void resetRevealedPairs()-zakryje všetky otočené vzory
* std::string getPexesso()-vráti reťazec s hracou doskou a jeho aktuálnym stavom

### Player

* vytvára objekt hráča s daným menom, skóre, mutexom a poradím
* std::string getName() const – vráti meno hráča
* int getScore() const – vráti aktuálne skóre hráča
* void setName(std::string names)-nastavuje meno hráča
* void setScore(int scores)-nastavuje skóre hráča podľa zadaného čísla
* std::mutex& getMutex()-vráti referenciu na mutex
* int getPoradie()-vráti poradie hráča
* void updateScore(int extraPoints)-zvyšuje skóre hráča
* std::string playerInfo() const – vytvára textový reťazec obsahujúci informácie o hráčovi, jeho meno a aktuálne skóre

### Client

* trieda Client predstavuje komunikáciu klienta, ktorý sa pripája k serveru v komunikačnom projekte
* void connectToServer(const char\* ipAddress, int port)-vytvorenie spojenia so serverom na základne IP adresy a portu
* void sendMessage(const char\* message)-táto metóda slúži na odosielanie správ na server
* ~Client()- uzavrie socket clienta

### Hrac1

* trieda Hrac1 vytvára inštanciu triedy Client
* klient sa pripája k serveru pomocou metódy connectToServer
* hráč zadáva správu pre server, kým nezadá príkaz “hraj“
* po zadaní príkazu hráč vstupuje do hlavnej časti hry
* v nekonečnej slučke hráč prijíma správy od servera a posiela správy na server
* po skončení hry program vypisuje príslušné hlásenia a končí

### Hrac2

* obe triedy Hrac1, Hrac2 dedia triedu Client

### Main

* trieda Server predstavuje implementáciu hlavného servera, ktorý komunikuje so všetkými klientami
* void clientHandler(int clientSocket, Pexesso& pexeso)- táto metóda je jediná metóda súboru main, ktorá vykonáva viacero funkcií: obsluhuje každého pripojeného klienta, komunikuje so všetkými klientmi, spracováva prijaté správy od klientov, koordinuje hru, pracuje s vláknami...
* int main()-hlavá funkcia programu, inicializuje pexesso s rozmermi 5x4 a nastaví znaky pre hru, inicializuje serverový socket a nastavuje jeho parametre, umožňuje pripojenie sa klientom k serveru, spúšťa pre každého klienta nové vlákno, spúšťa všetky vlákna, nakoniec čaká na ukončenie všetkých vlákien klienta

## Sockety

Sockety sú využité na komunikáciu medzi serverom a klientom. Server vytvára socket a nastavuje ho na naslúchanie na určenom porte pomocou blind() a listen(). Po akceptovaní klienta server získa nový socket na komunikáciu so vzdialeným klientom. Klient je pridaný do zoznamu klientov a je spúštané nové vlákno. Z pohľadu klienta, klienti vytváraju socket a pripájajú sa na server pomocou IP a portu. Komunikácia medzi serverom a klientom prebieha pomocou send() a recv(), kde klienti posielajú správy serveru a prijímajú odpovede. Každý klient má svoj vlastný socket, ktorý používa na komunikáciu s klientom.

## Vlákna

V našej aplikácii sme používali vlákna pre jednotlivých klientov na servery. Každé vlákno malo na starosti komunikáciu medzi klientom a serverom.

Hlavná funkcia main v triede Server vytvára socket, akceptuje pripojenia a spúšťa vlákno pre každé pripojenie v metóde clientHandler. V triedach Hrac1 a Hrac2 sú vlákna využité pomocou knižnice std::thread na spustenie samostatného vlákna pre každého klienta, čo umožňuje serveru súčasne obsluhovať viacerých klientov bez blokovania.

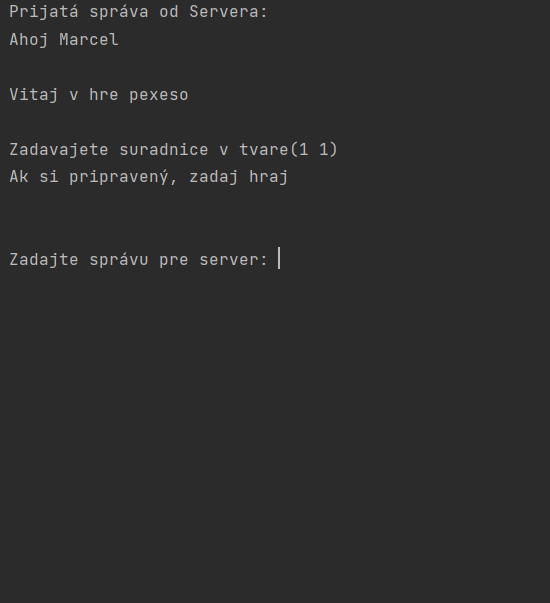
## Synchronizačný problém

V našom projekte sa synchronizačné problémy vyskytli pri súčasnom pripojení viacerých hráčov k hracej doske a pri operáciách vyberania kariet.

Problém nastáva pri tom, že server v aktuálnom čase musí vykonávať rôzne akcie, pre hráča ktorý je na rade a pre hráčov, ktorí čakajú na rad. Hráč ktorý je na rada posiela serveru správu so súradnicami, ktoré sa majú odokryť. Hráčom ktorí nie sú na rade server posiela aktuálnu verziu pexesa, takže iba čakajú na prijímanie správ, ktoré vypisujú. Na serveri sme si vytvorili metódu clientHandler, ktorá sa spúšťa ako nové vlákno pre každého klienta, v ktorom beží jeho program. Na začiatku metódy sa vykoná kontrola, či je hráč na rade ak nie je tak jeho metóda sa uzavrie cez condotion\_variable. Hráč ktorý je na rade pošle všetkým ostatným hráčom správu, ktorá začína “$” + aktuálne pexeso a dolár znamená že majú čakať na svoj ťah. Klientovi ktorý je na rade pošle správu s “%” na začiatku, čo znamená že bude načítavať súradnice. Po prijatí súradníc na serveri sa vykoná algoritmus na vyhodnotenie súradníc. Ak je zhoda, tak sa pridajú body, a potom sa ďalší klient dostáva na radu. Táto časť kódu sa opakuje až pokiaľ nie je odhalené celé pexeso alebo klient zadá “end”.

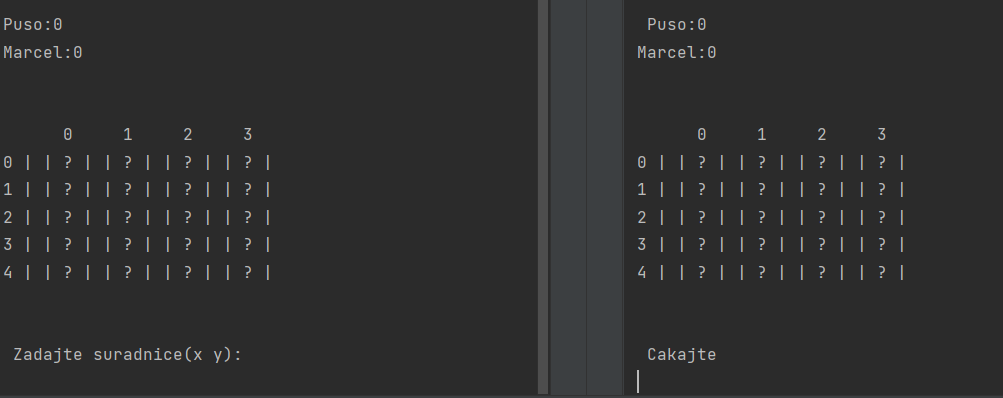
# Používateľská dokumentácia

Zobrazenie zo strany klienta po pripojení na server a následné uvítanie klienta.



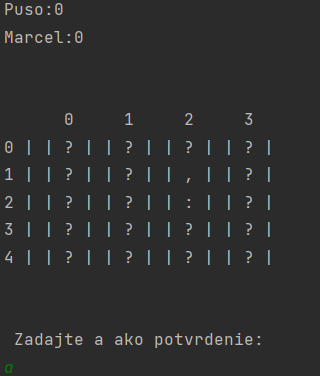
Obrázok Úvodné zobrazenie zo strany klienta

Zobrazenie pexesa v termináli, kde jeden klient je na rade, ktorý zadáva súradnice v podobe (x y) a druhý klient čaká na radu.



Obrázok Po zadaní správy hraj

Po zadaní súradníc v termináli vypíše správu pre potvrdenie zakliknutia súradníc prostredníctvom písmena “a”.



Obrázok Zadanie znaku na potvrdenie

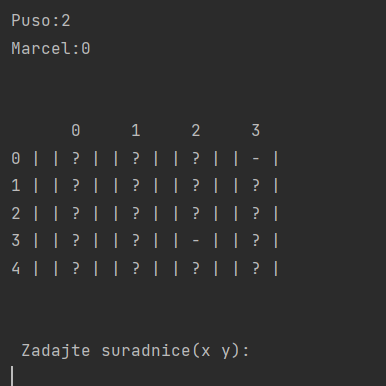
Po neuhádnutí zadaných súradníc prvého klienta odomkne druhého klienta, ktorý je na rade a má zadať súradnice. Týmto krokom automaticky zamkne prvého klienta a vypíše mu do konzoly správu Cakajte.

Obrázok, na ktorom je elektronika, snímka obrazovky, elektronické zariadenie, text

Automaticky generovaný popis

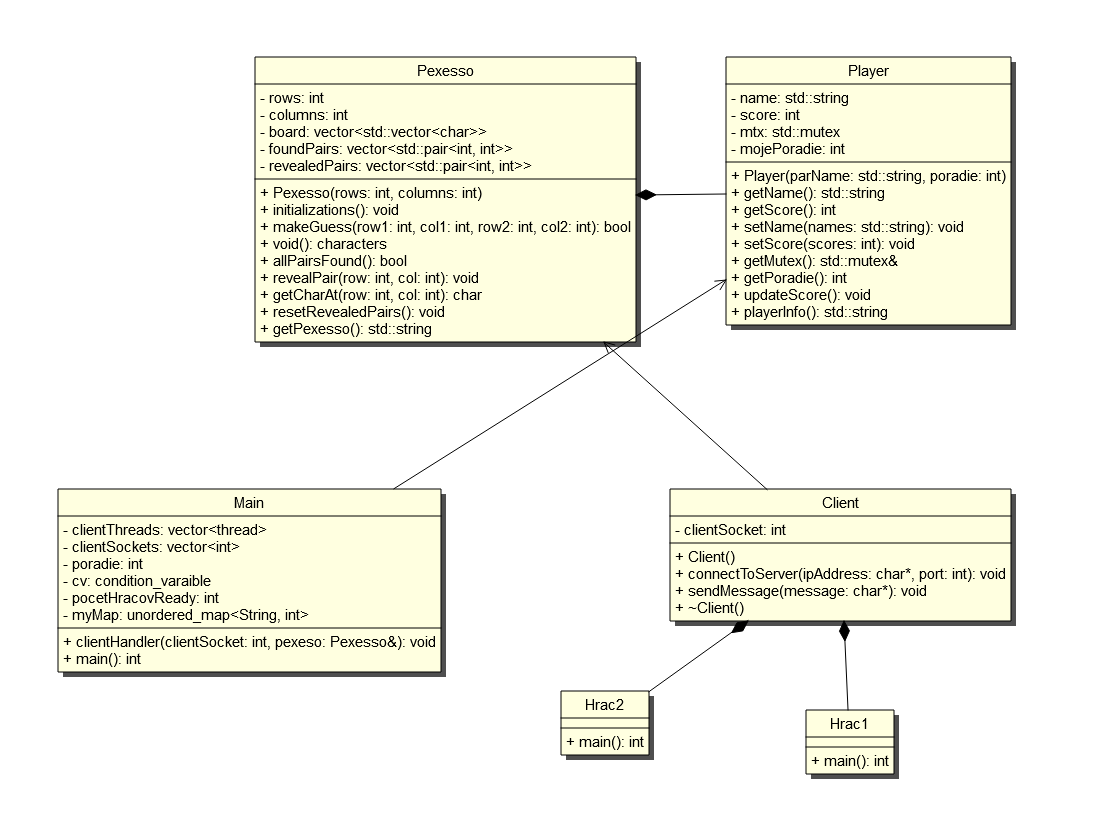
Obrázok Zadanie súradníc

Po zadaní súradníc, ktoré boli správne a obsahovali rovnaký znak pridá klientovi body a zadané súradnice zmení na znak “-”.



Obrázok Odhalenie znakov

# UML



Obrázok UML diagram