**Vyšší odborná škola**

**a Střední průmyslová škola elektrotechnická**

**Plzeň, Koterovská 85**

**Dlouhodobá maturitní práce s obhajobou**

Téma: **Piškvorky**

**Autor práce: Vojtěch Brázda**

**Třída: 4. I**

**Vedoucí práce: Mgr. Alan Koukol**

**Dne: 22. 4. 2022**

**Hodnocení:**

**Vyšší odborná škola**

**a Střední průmyslová škola elektrotechnická**

**Plzeň, Koterovská 85**

**Zadání dlouhodobé maturitní práce**

**Žák: Simon Valeš, Vojtěch Brázda**

**Třída: 4. i**

**Studijní obor:** **18-20-m/01 Informační technologie**

**Zaměření:** *Vývoj aplikací*

**Školní rok:** **2021 - 2022**

*Téma práce:* ***Piškvorky***

***Pokyny k obsahu a rozsahu práce:***

* ***Hrací plocha minimálně 15x15 polí***
* ***Možnost hraní proti živému hráči v rámci LAN***
* ***Možnost hraní proti AI (minimálně dvě úrovně obtížnosti)***
* ***Možnost volby počtu vítězných kol***

**Určení částí tématu zpracovávaných jednotlivými žáky:**

***Simon Valeš:***

*Vytvoření tlačítek na hrací ploše, dále pak základního menu pro výběr možností hraní proti AI nebo hráči a následně lze vybrat počet vítězných kol. Než hráči začnou hrát, připojí se do připojovacího lobby a následně se hra zapne. Vytvoření základní úrovně Ai.*

***Vojtěch Brázda:***

*Vytvoření základní hrací plochy, dále vyhodnocování výhry a grafickou úpravu základního pole. Zajistit bodovací systém pro hráče, aby věděli, kdo vyhrává. Udělat aplikaci možnost hraní přes LAN, mezi dvěma soupeři. Základní úroveň AI, poté alpha verzi pro testování, a nakonec vytvoření rozšířené AI.*

***Požadavek na počet vyhotovení maturitní práce:*** *2 výtisky*

*Termín odevzdání:* ***22. dubna 2022***

*Čas obhajoby:* ***15 minut***

Vedoucí práce: **Mgr.Alan Koukol**

Projednáno v **katedře VTT, ODP, PPE** a schváleno ředitelkou školy.

V Plzni dne: 30. října 2021 Ing. Naděžda Mauleová, MBA, v.r.

*ředitelka školy*

**Anotace**

V této maturitní práci se pokoušíme o vytvoření hry Piškvorky, podle zadaných kritérií. Po spuštění hry se objeví menu a začne hrát námi vybraná skladba, která nás doprovází po celou dobu chodu programu, pokud ji tedy v pravém dolním rohu nevypneme. V menu jsou nastavena 2 základní tlačítka HRA PROTI PC (námi vytvořeného AI, které má 2 úrovně) a HRA PRO VÍCE HRÁČŮ (myšleno tedy jen pro 2 hráče). Pokud vybereme hru proti AI, tak si zvolíme jednu nebo druhou úroveň, a taktéž počet kol. Pokud zvolíme hru proti hráči, tak si ještě musíme zvolit, kdo bude hru hostovat a kdo se připojí. Hostitel, neboli server nastavuje, na kolik kol se hra bude odehrávat. Tak či tak, po zvolení herního módu (AI, LAN) se vygeneruje pole tlačítek 15x15, kde následně po kliknutí na dané tlačítko, se objeví buď znak X nebo O. V horní části obrazovky, se přepíná, kdo zrovna má v daný okamžik hrát a napravo je vidět skóre, kdo má kolik vyhraných kol. Po výhře vyskočí okno s informací, kdo vyhrál.

„Prohlašuji, že jsem tuto práci vypracoval samostatně a použil(a) literárních pramenů a informací, které cituji a uvádím v seznamu použité literatury a zdrojů informací.“

„Souhlasím s využitím mé práce učiteli VOŠ a SPŠE Plzeň k výuce.“

V Plzni dne: …..................... Podpis: ….....................................

# Obsah práce

Maturitní práci jsme vytvářeli v programu InteliJ. Po vytvoření tříd jsme je rozdělily do balíčků, podle jejich funkčnosti v kódu, abychom se lépe orientovali v kódu. Máme tedy oddělené balíčky, jako například třídy, které pracují s AI jsou v balíčku AI, LAN třídy v balíčku LAN atd. Práce obsahuje nejen třídy, ale také rozhraní, které jsou ve balíčku Rozhraní. Také jsme vytvořili balíček pro obrázky, které v práci používáme, například pro tlačítka v menu.

Na začátku práce jsme vytvořili herní plochu, a až poté spouštěcí menu. V menu je umístěný uprostřed nahoře nápis Piškvorky, v levém horní rohu logo TicTac a v dolním pravém rohu tlačítko na zapnutí a vypnutí hudby. Pod nápisem Piškvorky jsou umístěné 2 tlačítka. První tlačítko je pro zapnutí hry proti AI. Druhé tlačítko Hra pro více hráčů, nám přepne tlačítka na výběr serveru, nebo clienta. Při výběru tlačítka Server, vyskočí okno, kde je zobrazena IP adresa počítače, na kterém je server spuštěn a výběr kol. Server zakládá hru jako hostitel, a client se k němu následně připojí. Při výběru tlačítka Client, vyskočí okno, kde uživatel musí zadat IP adresu serveru do textového pole, a následně se připojí. Před zapnutím každé hry se musí vybrat počet kol z JRadioButtonu, na které se bude daná hra hrát. Na výběr je 1, 3, nebo 5, přičemž je v základu nastaveno 1 kolo. Při spuštění hry z menu se tedy vygeneruje herní pole tlačítek 15x15, automaticky se nám herní okno přesune na střed naší obrazovky o poloviční šířce monitoru. Nad herní pole se nastaví label, na kterém se mění, kdo bude hrát v daný moment. Buď X nebo O. V pravo od něj je zobrazený stav vyhraných kol pro oba znaky. Po každé výhře se přičte bod pro znak, který vyhrál. Po celé odehrané hře vyhrává ten, jenž má více bodů a zobrazí se vyskakovací okno s informací kdo vyhrál.

Pro ošetření toho, aby se nemohli spojit znaky, které jsou v poli indexované hned za sebou, ale v herní ploše je jedno na konci řádky, a to druhé na začátku řádky hned pod, jsme vytvořili rozhraní, které tomuto zabraňuje. Také zabraňuje AI aby prováděla podobné úkony.

# Vlastní dokumentace

## Struktura

### Úvod

Tuto maturitní práci jsme si vybrali, protože máme rádi programovací jazyk Java, a taky nás oba vždy bavilo hrát piškvorky na papíru. Tak proč je neudělat programově?

Práci máme dokončenou, dle požadavků, které jsme měli dodržet. Naše aplikace obsahuje hrací pole 15 x 15, je zde možnost hrát proti komukoli v rámci LAN a také výběr 2 úrovní naší ručně vytvořené umělé inteligence. Tohle vše by měla naše maturitní práce umět. Jelikož je Java objektově orientovaný jazyk, tak jsme na vytvoření maturitní práce využili vlastnosti OOP, jako je dědění tříd, zapouzdření, polymorfismus. Rozvrhli jsme třídy do balíčků pro větší přehlednost ve struktuře kódu.

1. Obsah obrázku text

   Popis byl vytvořen automaticky

Nejprve jsme vytvořili hrací plochu pomocí tříd Start a Vykreslení. Dále jsme vytvořili menu před spuštěním hry, a to ve třídě Piškvorky. Doplnili jsme práci o rozhraní: Music, FirstTurn a Podmínky. Poté jsme začali pracovat na LAN komunikaci, pro kterou jsme vytvořili 4 třídy, a to 2 pro funkčnost: Server a Client a 2 pro připojovací lobby: SettingUpServer a SettingUpClient. Nakonec jsme se snažili popasovat s AI, pro kterou jsme vytvořili 3 třídy AI, CheckAi a MenuAi. Máme tam i třídu pro úpravu tlačítek a pak balíček ve kterém jsou obrázky, které jsme v naší aplikaci použili.

Ne vždy šlo vše hladce, a i některé malé problémy jsme řešili pár hodin. I přes to všechno jsme byli schopni tyto problémy vyřešit a podařilo se nám tuto práci dokončit. Teď už nemusíme hrát piškvorky na papíru, ale stačí nám jen zapnout naší aplikaci.

V úvodu student zdůvodní volbu tématu, stručně naznačí stav řešené problematiky v odborné literatuře, vytyčí hypotézu, cíl práce, zmíní metody a postupy práce, poukáže na aktuálnost řešení, případně na jeho přínos. Představí členění MP a přiblíží její strukturu (nikoli výčtem kapitol, spíše v obecné rovině).

### Jednotlivé kapitoly

#### Použité technologie

Pracovali jsme ve vývojovém prostředí InteliJ, které jsme už předtím používali na jiné projekty. Je to jednodušší než k takové rozsáhle práci používat například BlueJ. Protože je zde lépe přehledná struktura kódu, je lépe optimalizovaný a dokáže napovídat s kódem. Je zde i mnohem víc možností. Pro lépe zpracované grafické zobrazení, například tlačítek, jsme použili stránku na vytváření tlačítek buttonoptimazer a také Photoshop, ve kterém jsme je dále upravovali.

#### Timescheet

Obsah obrázku stůl

Popis byl vytvořen automaticky

#### Vytvoření Menu

Sice jsme začali s hracím polem, ale menu je přeci to, co je vidět jako první a tím i začneme. Skoro celé menu vytvářel Simon. Používáme třídu Piskvorky, která je umístěna v balíčku Tvoreni\_menu. Převážná část kódu je napsaná v konstruktoru, zbytek třídy tvoří 2 metody. Tato třída dědí z třídy MyButtons a implementujeme zde rozhraní Music.

Základem je JFrame, na kterým je JLabel a na něm nastavený BorderLayout. Po spuštění aplikace, se spustí menu vždycky v prostředku obrazovky o rozměrech 800x600. Není možné měnit jeho velikost. Celou aplikaci jsme udělali v modrobílém provedení. Takže JLabel je nastaven na modrou barvu a text “Piškvorky” je na něm nastaven na pozici CENTER a TOP bílou barvou. V pravo od něj je přidán JLabel, na kterém je pomocí ImageIcon přidán obrázek “Tic tac”, jako reference na anglické pojmenování této hry „Tic tac toe“. Pod textem “Piškvorky” jsou 2 tlačítka: Hra proti AI a Hra pro více hráčů. Na obě tlačítka jsou přidány ImageIcony, aby lépe vypadala. V pravém dolním rohu je tlačítko pro vypnutí a zapnutí hudby. I na něm je přidána ikona reproduktoru. Na všechna tlačítka jsou přidány actionListenery. Pod tlačítky je výběr kol. Bude se hrát buď na 1, 3, anebo 5 kol.

Po stisknutí na tlačítko Hra proti AI, se objeví nový JFrame třidy MenuAI, na kterém zůstává text Piškvorky, a přidají se 3 nová tlačítka. Předchozí JFrame se disposne. 2 tlačítka jsou na zvolení úrovně AI a poslední tlačítko na vrácení se zpět na předchozí JFrame. Po stisknutí tlačítka Hra pro více hráčů, se změní obsah daného JFramu. Tlačítka Hra proti AI a Hra pro více hráčů se nastaví na setVisible(false) a tlačítka jako Server a Client se nastaví na setVisible(true), dále se zviditelní i tlačítko na krok zpět.

Při stisknutí tlačítka Server vyskočí okno s informací o IP adrese počítače, na kterém je server spuštěn, a pod ním radiobuttony pro výběr kol, a také 2 tlačítka, první pro spuštění serveru a druhé pro vrácení se do menu.

Při stisknutí tlačítka Client vyskočí okno s textfieldem, do kterého musí hráč zadat ip adresu serveru, aby se k němu připojil. Potvrdí správnost IP adresy tlačítkem pod ním. Pokud zadal špatnou adresu nebo není zaplý server, tak se ukáže error, že musí zadat správnou IP adresu, nebo počkat, až se zapne server. Také tam bude možnost vrátit se zpět do menu.

To celé se odehrává v konstruktoru třídy Piškvorky. Pod ním jsou ještě 2 metody. PocetKol a getAdress.

Metoda pocetKol je public a její návratová hodnota typu int. Vstupují do ní 3 parametry, a to 3 JRadioButtony: r1, r2 a r3. V ní je poté vnořeno větvení if. Podmínka v ifu nám vrací číslo 1,3,5 podle zvoleného JRadioButtonu. Pokud zvolen není ani jeden vrátí nám 0, ale to není možné, protože je v základu nastaven r1 s číslem 1.

Metoda getAdress je také public a její návratová hodnota je typu String. Nemá žádné vstupní parametry. K tomu, abychom mohli zjistit IP adresu jsme importovali knihovnu java.net.InetAddress. V této metodě používáme její proměnnou ip, kterou nejprve nastavíme na null. Poté ohraničíme kód v try a catch. V try nastavujeme adresu do proměnné ip: ip = InetAddress.getLocalHost(). A nakonec vracíme hodnotu přetypovanou na řetězec: return ip.toString().

Obsah obrázku text

Popis byl vytvořen automaticky

#### Vytvoření hracího pole

Po menu tedy je další důležitá a vlastně ta nejdůležitější část hrací pole. K vytvoření hracího pole jsme použili 2 třídy: Start a Vykresleni z našeho balíčku Spousteni\_hry. Dále na úpravu tlačítek na hracím poli používáme třídu MyButtons z balíčku Uprava\_tlacitka. Začneme tedy třídou Start, kterou jsme vytvářeli společně. Tato třída obsahuje konstruktor a 4 metody. Implementujeme zde rozhraní FirstTurn. Konstruktor této třídy získává šířku a výšku obrazovky a to pomocí speciální metody Toolkit.getDefaultToolkit().getScreenSize(); height = screenSize.getHeight() – 35; Následně získá výšku, odečte od ní 35px za pomocí metody getHeight() a uloží ji do proměnné height.Tohle zaručí pokaždé stejnou velikost okna na jakékoliv obrazovce. Dále metoda public void start(), se vstupními parametry objekt třídy Piskvorky piskvorky a String ai. Objekt třídy Piskvorky získá tento string a díky tomu co ten string obsahuje, buď “Lehká” nebo “Těžší” zvolí úroveň naší AI. Poté se vytvoří nový JFrame, který se nastaví velikostně podle proměnný height a díky (screenSize.getWidth()-frame.getSize().width)/2; se kdykoliv zobrazí na středu jakékoliv obrazovky. Poté vykreslíme pomocí cyklu for 225 tlačítek a přiřadíme je do ArrayListu.

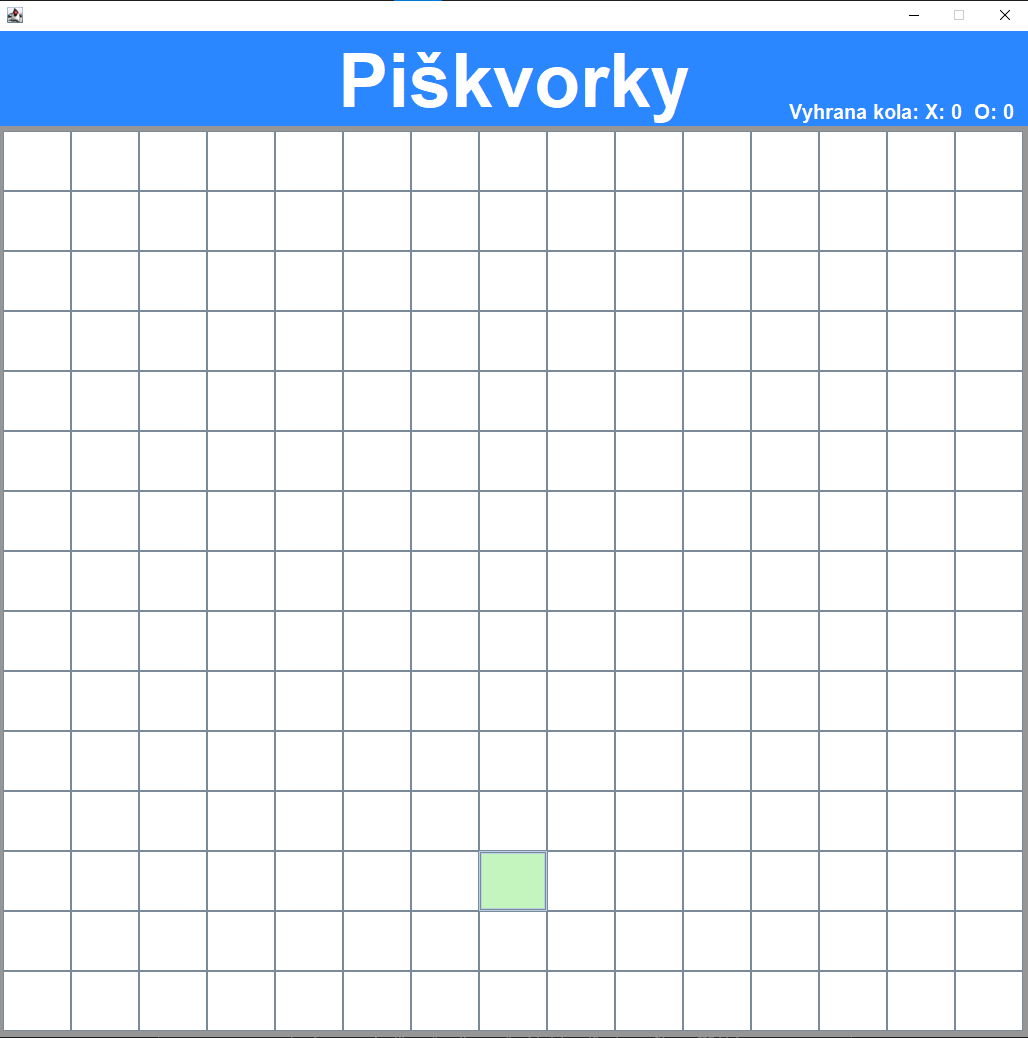
Podle zvolené AI se použijí metody třídy Vykresleni: vykresleniEz() nebo vykresleniHard(). Tyto metody nastavují, kdo má v daný moment hrát na textfield nad hrací plochou a po stisknutí tlačítka na něm nastaví znak X nebo O.

Kontrolují se zde hranice hrací plochy, aby nedošlo k vyjímce a znaky které jsou sice za sebou v ArrayListu, ale už nejsou za sebou v hrací ploše, se nespojily a díky tomu nedošlo k výhře. To díky rozhraní Podminky. Také jsou zde metody, které kontrolují, zda se může hrát do určitých směrů, od naposledy stisknutého tlačítka. Pokud dojde k výhře, vyskočí okno s informací o tom, který znak vyhrál a JLabel, který počítá skóre, připíše bod danému znaku.

Metoda startLan() dělá stejnou funkci jako metoda start(), až na to, že tam nepoužíváme AI, a po výhře musíme ukončovat komunikaci mezi serverem a clientem.

Metoda getServer předává instanci třídy Server, abychom ji mohli používat v této třídě.

Metoda getIndexTlaco() vrací hodnotu jako řetězec z tlačítka.



#### Rozhraní

Rozhraní jsou velkou součástí našeho kódu. O jejich vytvoření se postaral Vojtěch. Začneme tedy rozhraním Music. Toto rozhraní nám spouští muziku v našem programu. Implementujeme ho do třídy Piskvorky. Obsahuje pouze jednu metodu a to nacteni().

Metoda nacteni() načítá soubor typu .wav do instance rozhraní Clip. Museli jsme importovat speciální knihovny, abychom zde mohli pracovat se zvukovými soubory. K náš audioclip načteme do instance třídy InputStream, dále tuto instanci načteme pomocí speciální metody do instance třídy AudioInputStream, a tu poté nahrajeme do instance Clip. Díky tomu můžeme spustit námi vybranou skladbu.

Dále tedy rozhraní FirstTurn. Toto rozhraní kontroluje a nastavuje, kdo začne hrát jako první, respektive, který znak. Implementujeme toto rozhraní do tříd: Start, Client, Piskvorky, Vykresleni a MyButtons. Obsahuje 3 metody: firstTurn(), firstTurnLan() a firstturnAI().

Metoda firstTurn() nastavuje na textfield, který je nad hracím polem, kdo má začít hrát. Je v ní větvení if, které podle náhodného čísla, buď 1 nebo 2, vybere znak, který začne hrát. X má přiřazené číslo 1 a O zase 2.

Metoda firstTurnLan() dělá úplně to samé jako předchozí metoda, akorát zvolí, jestli začne hrát hráč, který zapl server, nebo hráč, který zapl clienta.

Metoda firstTurnAI() dělá rovněž to samé co předchozí metody, ale zvolí zda začne hrát hráč nebo umělá inteligence.

Třetí a zároveň poslední rozhraní je rozhraní Podminky. Toto rozhraní zajišťuje, že se nenapojí tlačítka, která jsou indexovaná v poli za sebou, ale nejsou ve stejné řadě. Tím je myšleno na okrajová tlačítka. Obsahuje 2 metody, akorát je jedna přetížená, takže jsou tam metody 3.

Metoda naplneni() naplňuje ArrayList typu Integer jménem podmínky. Má 3 vstupní parametry: int b, int pocatecni a int konecna. Int b udává o kolik indexů se budou hraniční tlačítka posouvat v daném ArrayListu. Int pocatecni určuje, jaké hraniční tlačítko je první. Int konecna určuje, které tlačítko bude jako poslední. Pomocí cyklu for přidá všechny hodnoty z daného okraje herní plochy do ArrayListu. Tímto taky pomáhá ošetřovat jiné metody.

Přetížená metoda naplneni() má stejný účel, akorát je udělaná i pro ošetření křížových směrů. Jsou zde tady 2 for cykly. Jeden je stejný jako u předchozí metody a ten druhý má stejnou funkci, akorát pro jiné hraniční tlačítka. Je zde o 3 vstupní parametry víc, které určují to samé, akorát se využívají v druhém cyklu.

Metoda vyhodnocení() vyhodnocuje zda je, nebo není index právě hraniční hodnotou. Má 2 vstupní parametry. Int poradi určuje index tlačítka, u kterého chceme zkontrolovat, zda je hraniční hodnotou. ArrayList podmínky je ArrayList, který jsme naplnili předchozími hodnotami. Je zde větvení if, které když podmínky obsahuje poradi: if (podminky.contains(poradi)), tak vrátí false, ale když ho neobsahuje, tak je zde další if, který kontroluje zdato poradi je větší než 0, a pokud je tak nastaví proměnnou pravda na true a tu poté vrátí.

#### Vytvoření LAN

Pro navázání komunikace LAN jsou zapotřebí dvě věci a to server, který je vytvořený na nějaké adrese a naslouchající na definovaném portu a client, který navazuje komunikaci s adresou, na které je založený server a portem, na kterém server naslouchá.

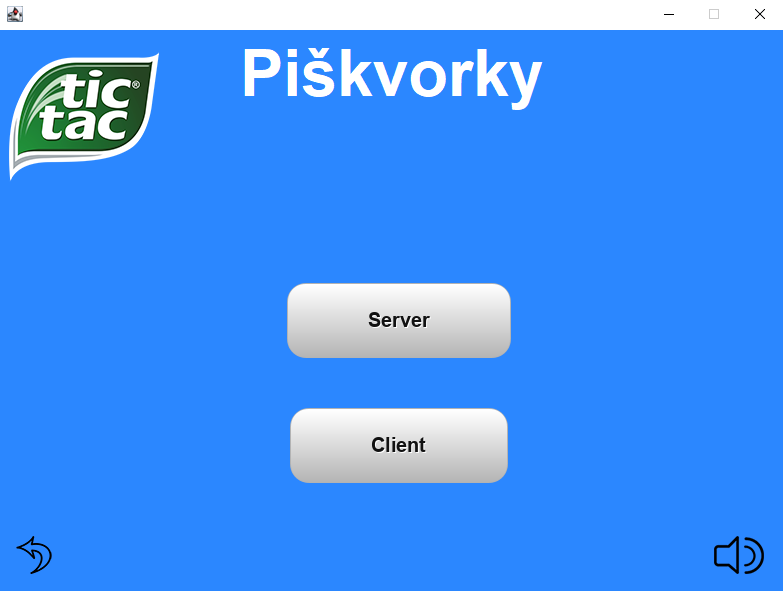
V naší aplikaci je zapotřebí nejdříve spustit server, ten se spustí na ip adrese, pod kterou je počítač v sítí připojený.

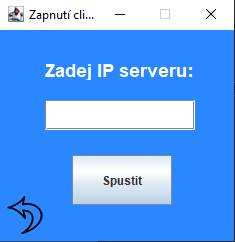
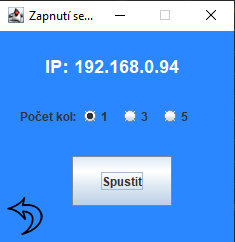
K tomu využíváme třídu SettingUpServer, zde se vytváří JFrame dialogy, na který je přidán JLabel barva, ten slouží jako layout pro tři JRadioButtony, jenž slouží jako volba pro hraných kol, JLabel ipaddress, který ukazuje na jaké adrese se server spustí, JButton zpatky, pro přenesení na předchozí obrazovku a JButton send, které nám zajistí, za pomocí metody firstTurn() a firstTurnLan() v rozhraní FirstTrun, kdo za jaký obrazec bude hrát, dále kolik kol budou hráči hrát, také skrývá JFrame dialogy, nakonec založí objekt třídy Server, jenž spustí její konstruktor. Třída také obsahuje metodu getAdress(), ta získává IP adresu počítače.

Třída server slouží jako spouštěč serveru, obsahuje konstruktor, který se o to stará. Nejdříve se vytvoří ServerSocket na daném portu, který je uložen do objektu třídy Server, poté se čeká, než se nějaký klient připojí k tomuto serveru. Následuje metoda startLan(), getServer(), jenž předává objekt třídy. Nadále je zapotřebí vytvořit DataInputStream, pro přijímání dat od clienta, DataOutputStream, pro odesílání dat clientovi a nakonec BufferedReader, který slouží ke čtení příchozích informací. Taktéž jsou zde vytvořeny dvě vlákna. Vlákno thread odesílá data clientovi, pokud je stlačené tlačítko jiné než minule a zároveň to není tlačítko, které stiskl client, tak se informace odešle a pro clienta se všechna tlačítka vypnou, dokud neodehraje protihráč. Vlákno thread2 slouží ke čtení příchozích informací od clienta, za podmínky, že příchozí informace se neshoduje s odchozí informací, se příchozí informace přetypuje na Integer, který představuje pořadí tlačítka v hracím poli. Nejdříve je ovšem zapotřebí zase tyto tlačítka zapnout, o což se stará metoda buttonOn, poté se na tlačítko s přišlým pořadím vyvolá umělé stisknutí díky metodě doClick(). Pokud přijde od clienta informace “Over“, ve vlákně se spustí metoda end(), jenž se stará o správné ukončení komunikace s clientem. Vlákna používáme, aby Server a Client mohli informace posílat a zároveň je přijímat. Třída ještě obsahuje metodu sendEnd(), která posílá klientovi potřebné informace k ukončení komunikace.

Při zapínání clienta využíváme třídu SettingUpClient, na rozdíl od třídy SettingUpServer obsahuje v JLabelu barva JLabel nazev, a JTextField ipaddress, který se stará o získání adresy, na kterou se client má připojovat, naopak neobsahuje JRadioButtony na volbu počtu kol, jinak zůstává vše stejné. Tlačítko send dělá také to samé, až na to, že vytváří objekt třídy Client, při kterém se spustí její konstruktor.

V konstruktoru se nejdříve tvoří dvě vlákna. První vláno thread je totožné s vláknem thread u serveru. V druhém vlákně thread2 je pouze přidána podmínka pro nastavení proměnné player1\_turn v rozhraní FirstTurn, aby client rozpoznal, s jakým znakem bude hrát. Taktéž obsahuje metodu end(), ta se stará o ukončení komunikace správnou cestou.





#### Vytváření AI

Pro spouštění naší umělé inteligence používáme třídu MenuAI, ve které je vytvořen JFrame frame, na ten je přidán JLabel vzhled, jenž slouží jako plátno pro další komponenty. Na JLabelu vzhled nastavený text Piškvorky a přidány tlačítka JButton b1, JButton b2 a JButton zpatky. Tlačítko zpatky se stará o navrácení na předchozí obrazovku a vypnutí aktuální obrazovky. Na b1 a b2 je přidaný ActionListener, který třída implemetuje, ten po kliknutí na jedno z tlačítek ničí aktuální frame, zakládá instanci třídy Start, následně pomocí podmínky, která rozhoduje o tom, jaké tlačítko bylo stlačeno, se ukládá do proměnné st daný text. Poté se spustí metoda start pomocí intence třídy Start, díky tomu se nám zobrazí hrací pole.

Následuje třída AI, která obsahuje metody getZnak, naplneni(), evaluationEz(), evaluationHa(), zapis(), find(), generateNumber(), getList(), move().

Metoda getZnak(), získává ArrayList tlačítek, z herní plochy a text, který budeme na tlačítkách hledat. Podle toho jaký to bude text, se rozhoduje, s jakým ArrayListem budeme metodu naplneni() volat v parametru.

Metoda naplneni() nastavuje globální proměnnou found na false a následovně projde ArrayList buttons, který obsahuje tlačítka z hrací plochy. Pokud najde tlačítko s hledaným textem, vytvoří se 8 objektů třídy CheckAI, následně se tyto objekty uloží do ArrayListu, který je poslaný do metody jako parametr.

Po té je ze třídy Vykresleni zavolána buď metoda evaluationEz() nebo evaluationHa().

Metoda evaluationEz() nastavuje proměnnou max na nulu a zakládá si objekt třídy CheckAI. Následovně projde celý ArrayList obranaList a vybere z něj objekt, který má největší číslo v proměnné row. Tento objekt uloží ho do dříve vytvořeného objektu. Pokud bude tato hodnota větší než nula a zároveň proměnná ai bude nastavena na true, zavolá se metoda zapis(). Jestli že bude hodnota rovna nule a zároveň proměnná ai bude nastavena na true, do lokální proměnné number se uloží hodnota ze zavolané metody find(), když metoda přepíše proměnnou found na true, zavolá se i metoda move(). V jiném ostatních případech zůstane proměnná found false a provede se podmínka, kde se zavolá metoda find(), s odlišným parametrem než předtím, uvnitř metody move().

Metoda evaluationHa() dělá to samé, jako metoda evaluationEz(), akorát po vytvoření objektu se nejdříve kontroluje ArrayList utokList a pokud se v ArrayListu obranaList nenajde vetší proměnná row, objekt se bude se uloží z ArrayListu utokList.

Metoda zapis() kontroluje, zda je index tlačítka na které chce umělá inteligence kliknou v rozsahu ArrayListu buttons a zdali na tomto tlačítku již není nějaký text. Pokud podmínka projde, pomocí metody doClick() uměle stiskneme tlačítko, poté se vyčistí ArrayListy utokList a obranaList, globální proměnná found se nastaví na true a metoda vrátí true. Jinak vrací false.

Metoda find() nastavuje lokální proměnnou index na nulu, následovně projde ArrayList buttons, který obsahuje tlačítka z herní plochy, pokud na tlačítku najde text u parametru metody, globální proměnná found se nastaví na true, do indexu se uloží index tlačítka v ArrayListu a cyklus se ukončí. Metoda následovně vrátí proměnnou index.

Metoda generateNumber() generuje náhodně číslo v rozsahu, který je určen vstupním parametrem number.

Metoda getList() ukládá ArrayList ze vstupního parametru do globálního ArrayListu list.

Metoda move() naplňuje ArrayList l 8 čísly, které jsou odvozeny od vstupního parametru index, který představuje index tlačítka na herní ploše. Následovně se zavolá metoda generateNumber(), jejíž výsledek se uloží do lokální proměnné number. Poté se do lokální proměnné written uloží výsledek zavolané metody zapis(), pokud tento výsledek bude false zavolá se metoda move() od znova, jinak se vyčistí ArrayList l.

Třídá CheckAI se stará o správné nastavení hodnoty row pro každý index, na který by umělá inteligence mohla zapsat.

Obsahuje metody check(), coundIndex(), getIndex() a getRow().

Metoda check() se nejdřív rozhoduje na základě vstupních parametrů, jaké hodnoty se uloží do proměnné znak1 a znak2, následně si uloží do proměnné index vstupní hodnotu z parametru index a zavolá metodu coundIndex(), jejíž výstup se uloží do proměnné index. Následně se vyhodnotí podmínka, která řeší, jestli je index v požadovaném rozsahu a jestli není roven vstupnímu parametru. V případě, že podmínka neprojde, vyhodnotí se podmínka, která zjišťuje, zdali je index v požadovaném rozsahu a jestli je se index shoduje s indexem ze vstupního parametru, pokud ano nastaví se proměnná row na nulu. V případě že předchozí podmínka projde spustí se podmínka, která vyhodnocuje, jestli na uloženém indexu obsahuje ArrayList buttons tlačítko, jehož text se shoduje s hodnotou v proměnné znak1, pokud ano, proměnná row se navýší o jedna a zavolá metoda se znovu zavolá, jinak se spustí další podmínka. Ta vyhodnocuje, jestli na uloženém indexu obsahuje ArrayList buttons tlačítko, jehož text se shoduje s hodnotou v proměnné znak2, pokud ano nastaví proměnnou třídy row na nulu.

Další podmínka zkoumá, jestli v ArrayListu button je na daném indexu tlačítko bez textu, pokud ano zavolá se metoda countIndex(), jejíž výstup se uloží do proměnné prom. Jestli bude prom v požadovaném rozsahu a zároveň v ArrayListu na indexu z proměnné prom bude tlačítko obsahující text jako znak1 navýší se row o jedna, toto se opakuje ještě jednou, ale s odlišnými hodnotami v parametru metody countIndex().

Metoda check() následovně vrací instanci třídy.

Metoda countIndex() na základě vyhodnocení switche vypočítá hodnotu odvozenou od parametru index a tu následovně i vrací.

Metoda getIndex() je přístupová metoda pro privátní a globální proměnné index této třídy.

Metoda getRow() je přístupová metoda pro privátní a globální proměnné row této třídy.

### Závěr

Myslíme si, že jsme zadání splnili. Menu je plně funkční a celkem i úhledné, hlavně přehledné a jednoduché. Mohlo tedy být hezčí, ale takhle bohatě stačí.

S hracím polem jsme spokojeni, taktéž je provedené v jednoduchém designu a hráči se v něm nemůžou ztratit. Text nad hracím polem upozorňuje, který znak má v daný okamžik hrát, kdyby náhodou hráč nevěděl. Když se hraje na více kol, tak je dobré vidět stav skóre, které je v pravém horním rohu.

Rozhraní jsou velmi důležitá v našem kódu. Sice nejsou abstraktní, jak to u nich má být, ale tím víc jsou nám k užitku.

LAN komunikaci zařídil Vojta skvěle, vše funguje a je to jednoduché i na použití. Možná to chtělo zlepšit připojování, tím i vytvořit lobby, kde by si client mohl vybírat hráče, s kterými by chtěl hrát. Ale tímto našim způsobem je zaručeno, že se kdykoliv připojíte právě vy k vašemu kamarádovi.

AI je rozdělena na 2 úrovně, jak bylo požadováno. Nikdo z nás nečekal, že to bude fungovat tak dobře.

Samozřejmě mohlo vše být ještě lepší, než je, ale i tak si myslíme, že jsme odvedli skvělou práci.

Když teď budu psát za sebe (Simona), tak jsem strašně rád, že jsem mohl mít za svého kolegu Vojtěcha, protože bez něj bych to nikdy nezvládl. Zatímco já jsem spíš přes ten design, tak on udělal neskutečně moc programátorské práce.