



**Gymnázium, Praha 6, Arabská 14**

**Obor Programování**



**Ročníkový Projekt**

Petr Soukop, 1.E

**Piškvorky**

Rád bych poděkoval panu Mgr. Janu Lánovi a panu Ing. Danielu Kahounovi, kteří mě učili tomuto předmětu.

Prohlašuji, že jsem jediným autorem tohoto projektu, všechny citace jsou řádně označené a všechna použitá literatura a další zdroje jsou v práci uvedené. Tímto dle zákona 121/2000 Sb. (tzv. Autorský zákon) ve znění pozdějších předpisů uděluji bezúplatně škole Gymnázium, Praha 6, Arabská 14 oprávnění k výkonu práva na rozmnožování díla (§ 13) a práva na sdělování díla veřejnosti (§ 18) na dobu časově neomezenou a bez omezení územního rozsahu.

V Praze dne 3. května 2021

**Název práce:** Piškvorky

**Autoři:** Petr Soukop

**Anotace:** Cílem projektu je vytvořit známou hru piškvorky (v poli 3x3) ve virtuální podobě. Součástí práce je vytvořit dvě základní prostředí, jedno pro hru o dvou lidských hráčích, a druhé pro hru o jednom lidském hráči, tedy “člověk proti počítači”. Práce by se tedy měla skládat celkem ze 3 částí. První částí je prostředí, kde by si hráč zvolil, zda chce hrát s dalším lidským hráčem, či “s počítačem”. Druhou částí je samotné prostředí pro piškvorky, které by u obou módu fungovalo stejnou či alespoň podobnou formou. A třetí částí je samotný “nelidský” hráč, který by měl na základě předepsané strategie vyhodnotit tah “lidského” hráče a učinit svůj vlastní tah. A samozřejmě by program vypsál, zda a kdo vyhrál, případně prohrál.

**Zadání:** Cílem práce je vytvořit automatického hráče piškvorek v poli 3x3. Automatický hráč by měl vždy vyhodnotit tah lidského hráče a na základě předepsané strategie zahrát vlastní tah. Program by také měl vypsát, zda lidský hráč vyhrál či prohrál.

# Obsah

1) Úvod .....	6
2) Historie hry.....	6
3) Jak program funguje? .....	7
3.1) Použité technologie.....	7
3.2) Pole.....	7
3.3) Nastavení hráče.....	8
3.4) Kliknutí.....	8
3.5) Výhra či prohra.....	9
3.6) Remíza .....	9
3.7) Zobrazení stavu .....	10
3.8) Druhá šance či konec?.....	10
3.9) Automatické reakce .....	11
4) Závěr.....	12
Bibliografie .....	13
Seznam použitých obrázků.....	14

# 1) Úvod

Projekt umožňuje si zahrát světoznámou hru piškvorky ve třech různých módech. Toto téma jsem si vybral, protože piškvorky zná úplně každý, tedy alespoň podle mě, a také jsem u této hry strávil hodně času. Přesto byl, ale projekt těžší, než jsem předpokládal.

Když jsem se rozhodl pro různé módy, tak mě napadlo, že by to také mohla být možnost k tomu vyzkoušet si vcelku podobný program různými cestami, proto je většina metod a proměnných specifikována k danému módu a není centralizována pro všechny. Osobně si myslím, že mód pro multiplayer je povedenější a efektivnější než zbylé dva módy. Hlavně protože automatické módy jsou plné překombinovaných a neefektivních podmínek, aby program fungoval, jak má. Bohužel mě, ale při vytváření nenapadl žádný jiný koncept pro tyto metody.

# 2) Historie hry

První zmínky piškvorek se tradují až zhruba do roku 2000 před Kristem v Číně v oblasti okolo Žluté řeky, ale byly objevené hry podobné piškvorkám i v období antického Řecka a také v předkolumbovské Americe.

Prvně se, ale hraje oficiálně zhruba v 7. století v Japonsku a má název Gomoku. Go v japonštině znamená pět a moku znamená průsečík. První mistrovství světa v piškvorkách se konalo roku 1989 v Japonsku. V České republice proběhlo mistrovství světa v roce 2009.

## 3) Jak program funguje?

### 3.1) Použité technologie

Použitými technologiemi jsou Java a JavaFX.

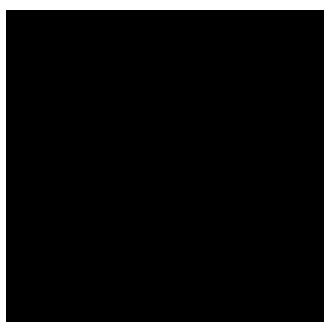
Java je objektově orientovaný programovací jazyk, známý pro svou univerzálnost a možnost běžet na několika platformách. Jazyk byl vyvinutý firmou Sun Microsystems roku 1995.

Hlavně díky tomu, že kompilovaný kód může běžet na všech platformách je Java v současnosti jedním z nejvíce používaných a nejpobulárnějších programovacích jazyků na světě. V roce 2020 byla Java dokonce na žebříčku nejpobulárnějších programovacích jazyků před jazyky C a Python.

JavaFX je moderní framework pro tvorbu grafických aplikací, postavený na bázi Java. JavaFX je velice pobulární pro své široké funkce (jako třeba podpora CSS stylů či podpora grafů a podobně), ale zároveň silným důrazem na jednoduchost. Je používána, jak pro tvorbu desktopových aplikací, tak i pro tvorbu mobilních aplikací či webových appletů.

### 3.2) Pole

Základním prvkem celého projektu bylo vytvoření pole.



Obrázek 1

Pole je druh proměnné, který jsem vytvořil jako jedno políčko hrací plochy. Funguje podobně jako třída Pane, respektive z ní dědí. Třída pole je deklarována jako pane o rozměrech 200x200 a s černým pozadím, na jejíž kliknutí se zobrazí znak momentálního hráče.

Ve třídě jsou též deklarovány metody `getHrac()`, `setHrac()`, `kliknuti()`.



### 3.3) Nastavení hráče

Metody `getHrac()` a `setHrac()` fungují pro nastavení hráčů u jednotlivých polí. Výrazně jsem je využil při vypracovávání automatického hráče. V metodě `setHrac()` je též deklarován vzhled pro jednotlivé hráče. Obě metody využívají proměnnou `hrac` typu `char`.

### 3.4) Kliknutí

K nastavení toho, co se stane v multiplayer módu, když je na jedno z polí kliknuto myší je popsáno v metodě `kliknuti()`. Ta funguje za pomoci proměnných `hrac` a `nynejsiHrac`, které se díky ní mění.

Obrázek 2

```
private void kliknuti() {  
    if (hrac == ' ' && nynejsiHrac != ' ') {  
        setHrac(nynejsiHrac);  
  
        if (vyhra(nynejsiHrac)) {  
            nynejsiHrac = ' ';  
        } else if (jePlno()) {  
            nynejsiHrac = ' ';  
        } else {  
            nynejsiHrac = (nynejsiHrac == 'X') ? 'O' : 'X';  
        }  
    }  
}
```

Tento problém jsem nejprve řešil pouze s jednou proměnnou, ale po pár pokusech jsem došel k tomu, že bude jednodušší vytvořit proměnné dvě. Metoda funguje tak, že pokud není `nynejsiHrac` prázdný, tak nastaví jeho hodnotu do `hrac`. A za pomoci dalších podmínek určí hodnotu `nynejsiHrac` po odehraném tahu.

Pokud jeden z hráčů vyhrál nebo nastala remíza nastaví `nynejsiHrac` prázdnou. Pokud, ale žádná z možností nenastane pouze předá `nynejsiHrac` novou hodnotu a to tu, kterou momentálně nemá. Překvapivě tato část byla pro mě jednou z nejtěžších, protože nejsem zvyklý používat ternární operátor, a když jsem se snažil napsat to bez něj bylo to složité.

V módu automatického hráče jsou jednotlivá kliknutí na jednotlivá pole deklarována v metodách `CompEasy()` a `CompHard()`.

## 3.5) Výhra či prohra

Zda a kdo vyhrál je v projektu vyhodnocováno zase dvěma různými způsoby. První, v multiplayer módu, funguje za pomoci metody `vyhra()`, pro tuto metodu jsem původně měl dvě verze pro X a pro O, ale nakonec jsem ji zefektivnil a nyní se dá jedna metoda užít pro obě varianty zadáním hodnoty proměnné `nynejsiHrac`, kterou jsem u původního návrhu ještě nevytvořil. Metoda pomocí jednoduchých podmínek a cyklů vyhodnotí, zda je zde nějaká výherní trojice, pokud ano vrátí hodnotu `true`, pokud ne vrátí `false`. V třídě `pole` je využita k zobrazování stavu.

Druhý způsob je využit v automatických módech. Ten je komplikovaně napsaný, ale vcelku primitivní, v zásadě jako všechno u těchto módů. Používá u každého z polí cyklus s podmínkami a nefunguje jako samostatná metoda, nýbrž jako součást metod `CompEasy()` a `CompHard()`.

## 3.6) Remíza

Remíza v multiplayer módu je znovu užita jako samostatná metoda, která užívá cykly a podmínky. Metoda vyhodnotí, zda je hrací plocha prázdná, pokud ano vrátí hodnotu `false`, pokud ne vrátí hodnotu `true`.


V automatických módech je remíza zapsána jako součást metod `CompEasy()` a `CompHard()` pomocí cyklů a podmínek. Původně jsem tuto část chtěl zapsat jako vnořené cykly a jednoduchou podmínku, ale docházelo k chybám, a proto jsem se rozhodl ji zapsat jako komplikovanou podmínku.

### 3.7) Zobrazení stavu

Pod název stav v projektu spadají texty, které ve hře indikují výhru, prohru a ostatní podobné věci. Znovu je tato část provedena ve více variantách pro různé módy.

V multiplayer módu je zapsána přímo ve třídě Pole, v metodě kliknutí(). Metoda určuje, kdy v multiplayer módu některý Pro každý ze stavů mění text pod hrací plochou:

z hráčů vyhrál, a také zařizuje výměnu hodnot proměnných nynejsiHrac a hrac.

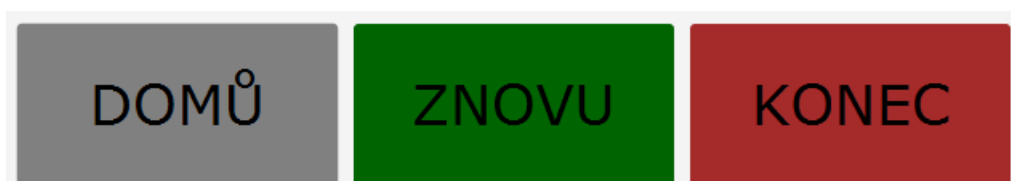


Obrázek 3

V automatických módech je stav vyřešen v rámci metod CompEasy() a CompHard() a to konkrétně v části na indikaci výhry, prohry či remízy. A též pro každý ze stavů nastavuje jiný text.

### 3.8) Druhá šance či konec?

Pod hrací plochou se nachází tři tlačítka:



Obrázek 4

První z nich, domů, slouží k návratu na úvodní stranu. Jednoduše nastaví scénu pro úvodní stranu, ale také má v sobě funkce druhého tlačítka, aby se restartovaly všechny hry. Mým původním nápadem bylo, že by se po dohrání hry objevilo nové okno s dotazem “Hrát znovu?” a dvě tlačítka ano a ne.

Tento koncept jsem, ale vymazal, protože by hráč pokaždé hru musel dokončit a neměl by šanci si změnit mód. Proto jsem přistoupil ke stálým třem tlačítkům, kdy můžete po jakémkoliv neúspěšném tahu hru restartovat a dělat, že se nic nestalo.

Druhé tlačítko, znovu, vychází z původního konceptu nového okna. Kromě vymazání všech polí, také vymaže text stavu. To je věc, u které mi došlo, že ji musím dopracovat až, když jsem hru párkrát ozkoušel.

A třetí a poslední tlačítko je konec. To funguje tak, že prostě ukončí program. V původním návrhu bylo toto tlačítko na úvodní straně, ale nejenom, že nevypadalo hezky, ale také mi přišlo zbytečné.

### **3.9) Automatické reakce**

Automatické reakce v metodách `CompEasy()` a `CompHard()` jsou provedeny jako soubor podmínek pro jednotlivá pole. Původně jsem měl v plánu napsat celou tuto část zcela jinak, ale když jsem zapsal metodu pomocí cyklů, tak nastala spousta chyb, a tak jsem se rozhodl zapsat pro každé pole trochu rozdílnou strategii.

Ve strategii piškvorek je klíčový první tah, tudíž jsem si zjistil nejlepší možné první tahy druhého hráče a ty jsem zapsal jako jedny z posledních podmínek. Celý soubor podmínek je strukturovaný jinak u lehkého a těžkého módu.

U lehkého módu má program vypsané pouze možné reakce na blokování soupeře a poté nejlepší možné tahy pro případ, že není, co blokovat. U těžkého módu má program nejprve vypsané všechny možné případy k vytvoření vlastní výherní trojice poté tahy k blokování soupeře, jako u lehkého módu, následně první tah k vytvoření trojice, tedy tahy na druhého pole v řadě, a konečně naposled nejlepší možné tahy, když není, co blokovat či spojovat.

## 4) Závěr

K závěru bych řekl, že jsem celkem spokojený s grafickou stránkou a funkcí. Ale u kódu by jistě šlo u dělat spousta vylepšení. Kupříkladu samotné reakce by šly zapsat mnohem jednodušeji a efektivněji. Také bych rád v budoucnu zefektivnil i zobrazení stavu, a také více prohloubil rozdíly obtížnosti mezi automatickými módy.

# Bibliografie

1. Wikipedie. Načteno z Piškvorky: <https://cs.wikipedia.org/wiki/Pi%C5%A1kvorky>
2. Wikipedie. Načteno z Tic Tac Toe: <https://en.wikipedia.org/wiki/Tic-tac-toe#Strategy>
3. Wikipedie. Načteno z Java: [https://cs.wikipedia.org/wiki/Java\\_\(programovac%C3%AD\\_jazyk\)](https://cs.wikipedia.org/wiki/Java_(programovac%C3%AD_jazyk))
4. Wikipedie. Načteno z JavaFX: <https://cs.wikipedia.org/wiki/JavaFX>
5. Lána, J. Načteno z Testovadlo: <https://gyarab.ddns.net/>
6. Oracle. Načteno z Java SE 8 Documentation: <https://docs.oracle.com/javase/8/index.html>
7. Stack Overflow. Načteno z <https://stackoverflow.com/questions/37111582/simplest-way-to-add-an-image-in-javafx/47102173>
8. W3Schools. Načteno z <https://www.w3schools.com/css/default.asp>
9. Fontspace. Načteno z <https://www.fontspace.com/aquire-font-f43735>

# Seznam použitých obrázků

Obrázek 1: Prázdné pole piškvorek .....	7
Obrázek 2: Ukazka kódu z metody kliknutí().....	8
Obrázek 3: Příklad ukazatele stav z multiplayer módu .....	10
Obrázek 4: Ovládací panel hry s tlačítky.....	10