# Gymnázium, Praha 6, Arabská 14

Obor Programování

## ROČNÍKOVÝ PROJEKT

## Hra Lodě

Petr Chalupa I.E

Duben 2020

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, Arabská14 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 …… dne …… Petr Chalupa ……

Anotace:
Cílem projektu je vytvoření uživatelsky jednoduché aplikace pro hraní hry Lodě digitálně na jednom zařízení. Aplikace dovoluje hru dvěma hráčům proti sobě, kdy si každý hráč může sám vybrat pozice pro své lodě. Hru si mohou hráči také ozvláštnit výběrem svého jména. Hra je velmi intuitivní a navíc vždy oznámí, co se má v dané chvíli dít. Grafika hry je velmi jednoduchá a tím pádem i jednoznačná.

# Obsah

# 1. Úvod

V tomto dokumentu je popsána funkce aplikace a řešení problémů spojených s vytvářením této aplikace. Téma projektu vzniklo ze zájmu vytvořit nějakou hru, která zabaví a zároveň není příliš složitá. Inspirací se stala klasická hra Lodě hraná na papíru.
Hraní hry je velmi jednoduché a probíhá na jednom zařízení, takže aspoň ve stávající fázi vývoje se musí oba hráči fyzicky sejít.

# 2. Zadání

Téma: Hra Lodě

Autor: Petr Chalupa

Popis:
Hráči si postupně vyberou pozice pro své lodě a pokud chtějí, tak i své jméno. Poté se střídají a hrají podle pravidel klasických papírových lodí. Hra pokračuje, dokud jeden z hráčů nezničí všechny protivníkovi lodě.

Platforma: JavaFX

# 3. Technologie a architektura

## 3.1. Použité nástroje a jazyky

Jako vývojové prostředí jsem používal Apache Netbeans IDE 12.2 společně s JavaFX Scene Builder 2.0. Projekt je napsaný v jazyce Java, specifičtěji na jeho platformě JavaFX a částečně stylovaný pomocí CSS.
Vývoj GUI nebyl příliš složitý, vzhledem k jeho jednoduchosti a veliké pomoci použitého Scene Builderu. Vývoj aplikační vrstvy zabral poměrně více času, hlavně kvůli řešení nastalých problémů, ale podařilo se ji také dokončit.

## 3.2. Architektura programu

Program je tvořen dvěma částmi.

a) GUI – Jednoduché uživatelské rozhraní pro hraní hry. Je tvořeno třemi obrazovkami, přičemž jedna z nich slouží jako dva v jednom. Je tvořeno tak, aby nic příliš nevyčnívalo a hráč nemusel jednotlivé prvky hledat.
b) Aplikační vrstva – Zde s nachází výpočty a veškerá logika. Děje se zde vše co hráč nevidí, ale zato ovlivňuje.

# 4. Algoritmus pro tvorbu polí

```
private void generateButtons() {
    for (int i = 1; i < 11; i++) {
        for (int j = 1; j < 11; j++) {
            ObservableList<Node> childrens = gridPane.getChildren();
            for (Node node : childrens) {
                if (node instanceof Button && gridPane.getColumnIndex(node) == j && gridPane.getRowIndex(node) == i) {
                    gridPane.getChildren().remove(node);
                    break;
                }
            }

            Button btn = new Button();
            btn.setId(j + ";" + i); // x;y coordinates
            gridPane.add(btn, j, i);

            btn.setOnAction(e -> {
                int selectedCount = Integer.parseInt(label_selected.getText().split("/")[0]);
                int selectedCountMax = Integer.parseInt(label_selected.getText().split("/")[1]);
                if (btn.getText().isBlank()) {
                    if (selectedCount < selectedCountMax) {
                        selectedCount++;
                        label_selected.setText(selectedCount + "/" + selectedCountMax);
                        btn.setText("X");
                    }
                } else {
                    selectedCount--;
                    label_selected.setText(selectedCount + "/" + selectedCountMax);
                    btn.setText("");
                }
            });
        }
    }
}
```

*Obrázek 1 - Algoritmus pro tvorbu polí*

Velmi důležitou součástí programu je algoritmus, který vytváří herní políčka 10 x 10. Je „recyklovatelný“, protože se používá při každé příležitosti, kdy jsou políčka potřeba, což zahrnuje výběrovou i herní fázi. On sám zaručuje vytvoření všech políček (se smazáním těch stávajících), jejich označování a odznačování, včetně kontroly maximálního počtu označených políček najednou.
Vývoj algoritmu nebyl dlouhý, ale zato poměrně převratný. Velkých předělávek se dočkala hlavně část, která maže políčka již vygenerovaná a také část kontrolující kolik políček je označených. Nyní je celkem jednoduchý a maximální počet políček není fixní.

# 5. Herní fáze a algoritmus pro herní kola

```
private void startPlay() {
    btn_submitSelection.setOnMouseClicked(null);
    btn_submitSelection.setOnMouseClicked((el) -> {
        if (label_selected.getText().equals("1/1")) {
            calculateShot("P2", p2Ships);
            if (p2Ships.length > 0) { //P2 can still play
                label_header.setText("Hraje " + p2Name);
                label_selected.setText("0/1");
                generateButtons();

                btn_submitSelection.setOnMouseClicked(null);
                btn_submitSelection.setOnMouseClicked((e2) -> {
                    if (label_selected.getText().equals("1/1")) {
                        calculateShot("P1", p1Ships);
                        if (p1Ships.length > 0) { //P1 can still play -> end of round
                            label_header.setText("Hraje " + p1Name);
                            label_selected.setText("0/1");
                            generateButtons();
                            startPlay();
                        } else {
                            try {
                                App.setRoot("results", p2Name); //P2 wins -> end of game
                            } catch (IOException ex) {
                                System.out.println(ex);
                            }
                        }
                    } else {
                        showAlert("Musíte vybrat pole!");
                    }
                });
            } else {
                try {
                    App.setRoot("results", p1Name); //P1 wins -> end of game
                } catch (IOException ex) {
                    System.out.println(ex);
                }
            }
        } else {
            showAlert("Musíte vybrat pole!");
        }
    });
}
```

*Obrázek 2 - Algoritmus pro herní kola*

Po výběru svých polí začíná herní fáze, při které se hráči postupně střídají ve vybírání cílů, tedy políček, kde by mohly být protihráčovi lodě. Aby mohli hráči hrát, musí existovat algoritmus, který zajistí průběh hracího kola a případně toto opakuje, dokud je to nutné. Je tedy potřeba, aby kromě samotného střídání hráčů hlídal i zásahy do lodí protihráče a v konečném důsledku i ukončení herní fáze ve chvíli, kdy jednomu z hráčů dojdou lodě.

Algoritmus využívá jednoduché logiky a opakovaného volání metody, ve které se nachází. V této fázi vývoje je fixně dáno, že vždy začíná hráč 1. Nejdříve je zkontrolováno, jestli hráč 1 nějaké pole vybral a poté jestli strefil nějakou loď hráče 2. Pokud ještě hráč 2 může hrát (má aspoň jednu loď), je tento postup zopakován pro hráče 2. Jakmile jeden z hráčů hrát nemůže, druhý hráč vítězí a končí herní fáze.

# 6. Ošetřování zvláštních situací

Vzhledem k tomu, že hra je offline, hraje se jen na jednom zařízení a nemusí pracovat s žádnými externími soubory, není potřeba řešit tolik výjimek. Je ale potřeba řešit několik zvláštních herních situací a ty musí být řešeny tak, aby tomu hráč rozuměl.

## 6.1. Výjimky

Pokud může nastat nějaká výjimka, je zachycena v try-catch bloku nebo throws metody.

### Seznam možných výjimek:

a) IOException

## 6.2. Zvláštní herní situace

Ve hře může nastat několik různých situací, na které je potřeba hráče upozornit. O to se stará jednoduché zobrazování alertů, tedy speciálně upravených dialogových oken, které situaci jednou větou popíší. Pokud hráč sám alert do několika vteřin nezavře, sám se odstraní.

### Seznam možných situací:

b) Při výběru polí si hráč nevybere všech deset
c) Hráč zkusí zadat prázdné políčko se jménem
d) Jména hráčů se shodují
e) Při hře hráč nevybere pole, na které chce střílet

# 7. Závěr

Odevzdaná práce splňuje původní zadání, ale je zde spousta prostoru pro zlepšení a další vývoj. Například lze hru obohatit o losování o začínajícího hráče, lepší grafickou podobu, ale i další herní režimy. Jako jednu z možností dalšího režimu vidím režim hráč proti počítači nebo režim online. Možných pokračování ve vývoji je spousta, ale nejdůležitější je, že se přes všechny problémy podařilo dostat aplikaci do funkční fáze.

# 8. Bibliografie a zdroje

Data, R. (2021). *W3Schools Online Web Tutorials*. Načteno z W3Schools: https://www.w3schools.com/

FAVPNG. (2021). *FAVPNG*. Načteno z FAVPNG.com - Free Transparent PNG Images: https://favpng.com/png_view/boat-origami-paper-origami-paper-origami-step-by-step-boat-png/FJ04BRXg

Jakob, M. (2021). *code.makery.ch | code.makery.ch*. Načteno z code.makery: https://code.makery.ch/

JavaTpoint. (2021). *JavaFX Tutorial - javatpoint*. Načteno z JavaTpoint: https://www.javatpoint.com/javafx-tutorial

Oracle Corporation. (2021). *Java Documentation - Get Started*. Načteno z Java Documentation: https://docs.oracle.com/en/java/

Stack Overflow. (2021). *Stackoveflow - Where Developers Learn, Share & Build Careers*. Načteno z Stackoveflow: https://stackoverflow.com/

# 9. Seznam obrázků