Vysoká škola ekonomická v Prahe
Fakulta informatiky a štatistiky
Katedra informačných technológií

Študijný program: Aplikovaná informatika
Odbor: Informatika

Podporné materiály pre výučbu GUI v JavaFX

BAKALÁRSKA PRÁCA

Študent : Veronika Vallušová
Vedúci : Ing. Jarmila Pavlíčková, Ph.D.
Oponent : doc. Ing. Jan Pour, CSc.

2016

Prehlásenie:

Čestne prehlasujem, že som bakalársku prácu spracovala samostatne a že som uviedla všetky použité pramene a literatúru, z ktorých som čerpala.

V Prahe 01. decembra 2016 . . . . . .

Veronika Vallušová

Poďakovanie

Týmto by som sa veľmi rada poďakovala vedúcej mojej bakalárskej práce Ing. Jarmile Pavlíčkovej, Ph.D., za čas, ktorý mi venovala pri zodpovedaní mojich otázok, za cenné rady, odbornú pomoc, návrhy, pripomienky, trpezlivosť a ochotu, ktorú mi poskytla pri vypracovaní moje bakalárskej práce.

Abstrakt

Táto bakalárska práca sa zaoberá tvorbou grafického užívateľského rozhrania v jazyku JavaFX 8 a jeho využití pri výučbe na odbore Aplikovaná informatika na Vysokej škole ekonomickej. Na začiatku práce sa čitateľ dozvie bližšie informácie o charakteristike jazyka JavaFX 8 a jeho architektúre. Každá aplikácia má svoju logickú a grafickú časť. Grafická časť sa stará o vytvorenie vzhľadu celej aplikácie prostredníctvom využitia rôznych komponentov a panelov. Ich prehľad, možnosti použitia a praktické ukážky zdrojového kódu pochádzajúceho ako z vývojového prostredia NetBeans, tak aj z programu JavaFX Scene Builder, sú obsahom nasledujúcej kapitoly. Nakoľko sa predpokladá, že Scene Builder, nebude mať čitateľ stiahnutý, v práci je taktiež popísaný krátky popis inštalácie. V závere tejto bakalárskej práce je ukázaná jedna z možných variant ako by mohla vyzerať výsledná aplikácia pre dosiahnutie dostatočného počtu bodov na ukončenie prvej časti individuálnej semestrálne práce z nadväzujúceho kurzu 4IT115 Softvérové inžinierstvo.

Kľúčové slová

JavaFX 8, CSS, komponenta, panel, Adventúra, Výučba, grafické užívateľské rozhranie

Abstract

This bachelor’s thesis deals with the Graphical User Interface in JavaFX 8 and how it can be used for studying the field of Applied Informatics at the University of Economics. The first chapters provide details on the characteristics of the JavaFX 8 language and its architecture. Moreover, each application has its own logical and graphical part. The graphical part consists of the appearance of the application using a variety of panels and components and which should be used. In addition, a code sample is elaborated on how to program those using NetBeans and the JavaFX Scene Builder program. It is assumed that the reader will not have the Scene Builder downloaded yet, in this case, this bachelor’s thesis also describes a short installation description. Lastly, the current thesis will present the final applications as a product that is aimed at achieving a sufficient mark for the first part of the individual work of a follow-up course 4IT115 Software Engineering.

Keywords

JavaFX, CSS, component, panel, adventure, lecture, Graphical User Interface

Obsah

1. Úvod

V základnom kurze 4IT101 Programovanie v Jave na Vysokej škole ekonomickej v Prahe (ďalej VŠE) študenti získali základné informácie o programovaní v objektovo programovacom jazyku Java. Oboznámili sa so základnými programovacími technikami, základnou konštrukciou metód, spoznali jednoduché dátové typy a dátové štruktúry (List, Set, Map a pole). Bližšie bolo popísané čo sú výnimky, triedy, rozhrania, dedičnosť, polymorfizmus a jednotkové testy. Cieľom tohto kurzu bolo naprogramovať aplikáciu, ktorá bude spĺňať požiadavky uvedené na stránke www.java.vse.cz. Ako vzorová aplikácia bola vytvorená jednoduchú hru o Červenej Čiapočke. [BENÁČANOVÁ, 2015]

V nadväzujúcom kurze 4IT115 Softvérové inžinierstvo, ktorý sa tak isto vyučuje na VŠE, je pokračovaním aplikácie vytvorením grafického užívateľského rozhrania (ďalej GUI) na platforme JavaFX 8. V tejto verzii je možné využiť dva spôsoby tvorby GUI. Klasickým spôsobom, a to napísaním kódu jazykom JavaFX alebo využitím jazyka FXML pre vytvorenie GUI. Kurz 4IT115 Softvérové inžinierstvo sa zatiaľ vyučoval klasický spôsob programovania v programe NetBeans. Nakoľko sa od semestra 2016/2017 vyučuje aj pomocou JavaFX Scene Builder, táto práca je zameraná zároveň na programovanie v tomto programe.

1.1 Cieľ práce

Cieľom bakalárskej práce je vytvoriť príručku pre ďalších študentov VŠE študujúcich kurz 4IT415. Táto práca sa snaží obohatiť čitateľa zo základnými informáciami ohľadom jazyka JavaFX a urýchliť mu rozhodnutie, ktorý panel, komponent by mal použiť pri programovaní. Pripojený zdrojový kód obsahuje logickú aj grafickú časť aplikácie, teda ako by mohla vyzerať finálna verzia semestrálnej práce. Obidva programy majú rovnaký výstup, predstavujú rovnakú aplikáciu, avšak sú tam odlišnosti v zdrojovom kóde. Naprogramovanie aplikácie v oboch prípadoch sa považuje za zaujímavé, inšpiratívne ako sa dá naprogramovať tá istá aplikácia v inom programe. Pre jednoduchšie porovnanie bola naprogramovaná aj jednoduchá aplikácie s jedným tlačidlom, kde čitateľ na tomto príklade dokáže lepšie pochopiť aký je rozdiel a ako fungujú obidva programy. Predpokladom lepšieho porozumenia tejto práce sa očakáva, že každý čitateľ má základné vedomosti o programovaní v jazyku Java. Tieto vedomosti bolo možné získať v základnom kurze 4IT101 Programovanie v Jave.

1.2 Štruktúra práce

V úvode bakalárskej práce bude bližšie popísané, čo je JavaFX a architektúra platformy JavaFX 8. Počas tvorby grafickej aplikácie je možné použiť množstvo panelov a vkladať rôzne druhy komponentov. Všetky komponenty, ktoré budú použité vo finálne aplikácií budú bližšie popísané v úvodných kapitolách tejto práce. Ako vytvárať každý jeden komponent prvým spôsobom, teda využitím programu NetBeans, a zároveň použitím JavaFX Scene Builder bude objasnené následne. Preto je v závere práce stručné popísaný postup ako správne nainštalovať tento program. Na ukážkovom vzore môže čitateľ vidieť, ako by mohla vyzerať finálna aplikácia pre dosiahnutie dostatočného počtu bodov, pre úspešné ukončenie prvej časti individuálnej semestrálne práce z nadväzujúceho kurzu 4IT115 Softvérové inžinierstvo. V prílohách je možné nájsť pripojený zdrojový kód funkčnej grafickej verzie adventúry obidvomi spôsobmi. Pre jednoduchšie porovnanie bola naprogramovaná aj jednoduchá aplikácie s jedným tlačidlom, kde čitateľ na tomto príklade dokáže lepšie pochopiť aký je rozdiel a ako fungujú obidva programy.

1.3 Rešerš

JavaFX 8 patrí k najnovšej verzii jazyka Java, čo je možné vidieť aj na dostupnej odbornej literatúre. Všetky zdroje popisujú ako si užívateľ môže vytvoriť svoju prvú jednoduchú aplikáciu v jazyku JavaFX, ale aj zložitejšie aplikácie. Najviac dostupných informácií k tejto problematike vie čitateľ nájsť na stránkach www.oracle.com, teda na stránkach samotného vývojára programovacieho jazyka, z ktorých boli čerpané informácie aj v tejto bakalárskej práci. Stránka obsahuje popis všetkých komponent, s ukážkovými kódmi ako s komponentmi pracovať.

Ďalšie informácie môže čitateľ nájsť v anglickej knihe s názvom „Learn JavaFX8, Building User Experience and Interfaces with Java 8“ od autora knihy Kishori Sharan. Táto kniha je určená pre Java vývojárov od začiatočníckej až po pokročilejšiu úroveň. Popisuje ako vytvárať GUI aplikácie použitím práve JavaFX 8. Autor sa snaží priblížiť programovanie každej komponenty pomocou krátkej ukážky kódu.

Práca bývalého študenta školy VŠE, Jána Kopeckého bola dobrou inšpiráciou a taktiež dobrým zdrojom pri písaný tejto práce, nakoľko jeho práca s názvom: „JavaFX 8 a jej

využitie pri výučbe“, je veľmi podobná s témou tejto bakalárskej práce. Študent Michal Keder písal bakalársku prácu s názvom „Příručka tvorby databázové aplikace v Javě“. Pre nadobudnutie ďalších vedomosti ohľadom jazyka Java sa odporúča prečítanie aj tejto bakalárskej práce.

V prípade, že pri programovaní dôjde podľa nás k neriešiteľnej situácii, existuje stránka stackoverflow.com. Skúsenejší programátori odpovedajú na otázky, prečo nám tento problém vznikol, a ako by ho bolo možné jednoducho vyriešiť.

1.4 Obmedzenia práce

Predpokladom lepšieho porozumenia práce sa predpokladá, že čitateľ bude mať základné znalosti programovanie. Tieto znalosti bolo možné získať na základnom kurze 4IT101 “Programování v Javě” vyučujúcom na VŠE.

2. JavaFX

„JavaFX je súbor grafických a mediálnych balíčkov, ktoré umožňujú vývojárom designovať, vytvárať, debugovať a spúšťať bohaté užívateľské rozhrania, ktoré sú schopné fungovať na rôznych platformách [ORACLE, 2013a]. Pod pojmov bohaté sa rozumie vizuálne. Vďaka využitiu obrázkov, videí, hudby, grafov, CSS štýlov a ďalších technológií, môže aplikácia vyzerať atraktívnejšie pre užívateľov. Zároveň novšia verzia umožňuje uložiť našu aplikáciu ako desktopovú, webovú alebo mobilnú aplikáciu napísanú na platforme Java [ITNETWORK, 2016].

2.1 JavaFX 8

V programe JavaFX 8 je možné vytvárať grafické uživateľské rozhranie (GUI) dvomi spôsobmi:

- Programovaním rovnakým spôsobom ako v starších verziách pomocou Swingu. Celý kód grafickej časti aplikácie je napísaný v zdrojovom kóde v programe NetBeans,
- Druhým spôsobom je použitie jazyka FXML, odvodeného z XML. Samostatný program JavaFX Scene Builder prepojený s programom NetBeans dokáže samostatne prepísať kód do jazyka FXML. Grafický vzhľad si užívateľ vytvára samostatne pomocou programu Scene Builder ale kód aplikácie v jazyku FXML sa ukladá do programu NetBeans automaticky. Samozrejme je možné aj ručné písanie FXML kódu. Ide však o zložitejší a časovo náročnejší postup [ITNETWORK, 2016]. Podrobnejšie o tomto spôsobe bude popísané v celej kapitole 8 popis novo vytvoreného projektu.

2.2 Architektúra JavaFX 8

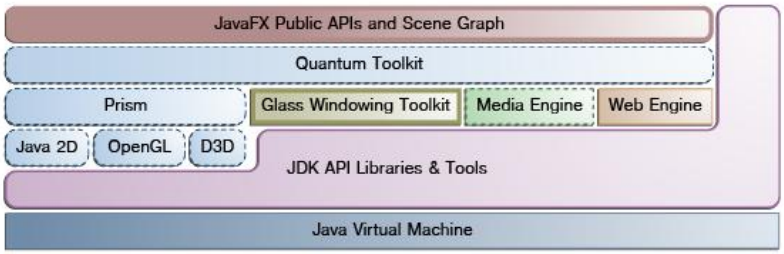
Obrázok 1 obsahuje architektúru komponent platformy JavaFX 8, ktorá rozdeľuje komponenty do vrstiev a ukazuje ako sú navzájom prepojené, ako spolupracujú.

Vrchná vrstva JavaFX sa skladá z dvoch častí: Public APIs[1] a Scene Graph, ktoré spolu predstavujú hlavné rozhranie pre programátora. Nasledujúca vrstva Quantum Toolkit je potrebná pre tvorbu GUI. Okrem tejto vrstvy sa na GUI podieľa aj knižnica Prism zložená z Java2D, OpenGL[2] a D3D[3] a ďalšie časti Glass Windowing Toolkit (Glass), Media Engine,

[1] Application Programming Interface

[2] Open Graphic Library

a WebEngine. Nato, aby bolo možné spustiť a vytvárať Java aplikácie, je veľmi potrebná JDK[4] API Libraries&Tools. Teda súbor knižníc a nástrojov dostupných v jazyku Java potrebných na spustenie JavaFX kódu. Všetky aplikácie vytvorené v tomto jazyku pracujú na Java Virtual Machine, na virtuálnom stroji Javy, ktorý zabezpečuje spustenie programu na počítači [ORACLE, 2013b].



Obrázok 1 Architektúra platformy JavaFX8. (Zdroj: [ORACLE, 2013b])

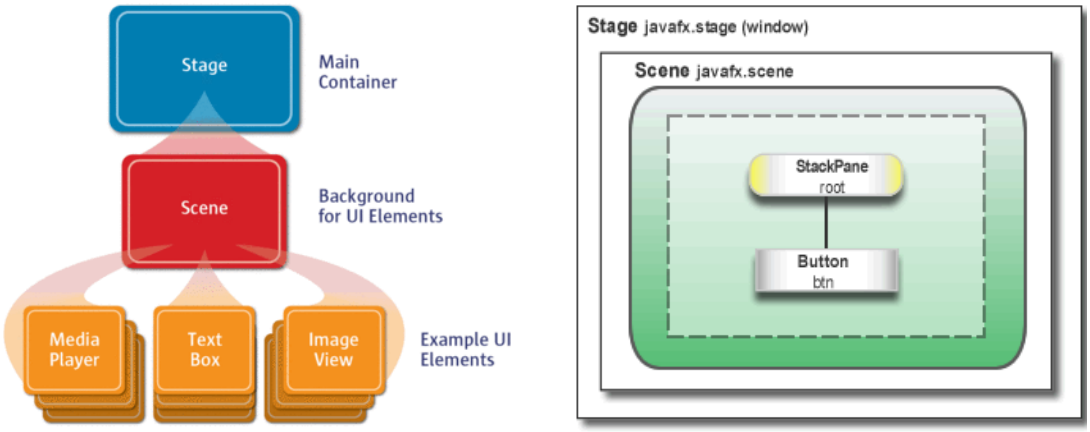
2.2.1 Scene Graph

Scene Graph je základným prvkom pre JavaFX aplikácie. Má štruktúru hierarchického koreňového stromu a predstavuje základný prvok pre tvorbu GUI, do ktorej sa následne vkladajú ďalšie komponenty. Komponent, do ktorého je možné vložiť ďalšie komponenty sa nazýva kontajner.

Hlavným kontajnerom pre spustenie JavaFX aplikácií je stage. Reprezentuje pozadie, na ktorom celá aplikácia behá, teda hlavné okno aplikácie. Pridávajú sa do neho scény, počet okien koľko má mať aplikácia. Napríklad úvodná scéna s možnosťou prihlásiť sa k aplikácií, druhou scénou bude potom samotná aplikácia. Scéne môžu byť následne pridelené jednotlivé grafické elementy, ako napríklad obrázok, video, text, tlačidlo a podobne.

[3] Direct3D

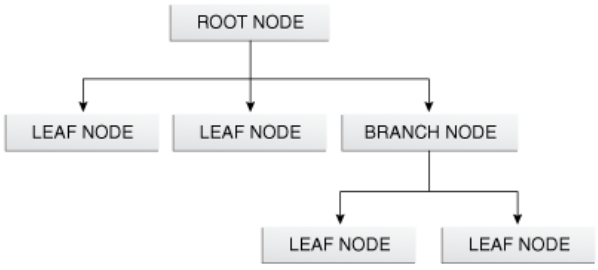
[4] Java Development Kit



Obrázok 2 Schéma scén v JavaFX aplikáciách. (Zdroj: [JAKOB, 2015])

Každý komponent je možné zaradiť do jednej z nasledujúcich častí, podľa toho či môže mať potomkov alebo predkov.

- Root Node(Node) – hlavná trieda, predstavujúca stage. Nemá predka ale môže mať niekoľko potomkov,
- Branch Node (Parent) – predstavuje scénu. Ako predka má stage a môže mať niekoľko potomkov,
- Leaf Node (Children) – grafické elementy. Posledné uzly, ktoré nemôžu mať potomkov. [ORACLE, 2013a]



Obrázok 3 Štruktúra Scene Graph. (Zdroj: [ORACLE, 2013a])

2.2.2 API

Do vrchnej vrstvy patrí ešte API pre podporu všetkého, čo dokáže urobiť JavaFX 8. Užívateľ si môže predstaviť súhrn všetkých knižníc, ktoré obsahujú hotové triedy a metódy, používané pri programovaní. Pretože JavaFX 8 je už časťou Java 8 SE, API tak získava všetky možnosti, ktoré umožňuje Java. Samotná JavaFX API pokračuje vo svojom modeli z verzie 1 avšak viac

sa spolieha na webové štandardy, akým sú napríklad CSS[5] na úpravu vzhľadu a ARIA pre uľahčenie prístupu [KOPECKY, 2015].

2.2.3 Grafický systém

Grafický systém je implementačný detail pod vrstvou JavaFX Scene Graph. Využíva možnosti hardvérového a softvérového vybavenia počítača. V prípade, že nie je hardwarové zázemie počítača dostačujúce pre vykreslenie, umožní grafický systém softwarové vykresľovanie. Obsahuje dve časti Prism a Quantum Toolkit, ktoré podporujú 2-D (dvojrozmerné) ale aj 3-D (trojrozmerné) prevedenie.

Prism

Prism využíva hardvérové aj softvérové vykresľovanie vrátane 3D grafiky. Používa hlavne hardwarové vykresľovanie, ale pokiaľ nie je dostupné, využíva Java2D, ktorá je dostupná na všetkých JREs[6]. Dokáže fungovať na všetkých operačných systémoch, ktoré podporujú platformu Java

- Windows XP a Windows Vista s využitím DirectX 9,
- Windows7 a Windows 10 s využitím DirectX11,
- Mac, Linux s využitím OpenGL.

Quantum Toolkit

Quantum Toolkit spája Prism a Glass Windowing Toolkit dohromady, a tým sprístupní JavaFX vrstvu, ktorá sa nachádza nad touto vrstvou. Zodpovedá za pravidlá pre prácu s vláknami, za spustenie celej grafiky aplikácie a spracovanie reakcií z vonkajších vstupov.

2.2.4 Glass Windowing Toolkit

Najnižšou vrstvou, ktorá spracováva grafickú časť aplikácie je Glass Windowing Toolkit. Jeho hlavnou vlastnosťou je zodpovedať za podporu služieb operačného systému ako je nastavenie okna aplikácie, časovač, a zároveň jeho prepojenie s platformou JavaFX. Zároveň zodpovedá za správu udalostí a beží na rovnakom vlákne ako celá aplikácia. Na rozdiel od

[5] Cascade Style Sheet

[6] Java Routine Enviroments

AWT[7], ktorá beží na vlastnom vlákne a JavaFX aplikácia na druhom vlákne. To spôsobovalo veľa problémov, ktoré sa vyriešili spustením aplikácie bežiacej na jednom vlákne. Celý systém beží na aspoň dvoch, poprípade na troch vláknach v rovnakom čase.

- JavaFX aplikačné vlákno – predstavuje primárne vlákno, ktoré využívajú programátori, vývojári. Ku každej živej scéne (scéna, ktorá je súčasťou okna) musíme pristupovať práve z tohto vlákna. Scénu je možné vytvárať a upravovať na inom vlákne v pozadí, ako náhle je koreňový uzol prepojený s nejakým živým objektom, musíme k nemu pristupovať cez JavaFX aplikačné vlákno. To umožňuje tvorbu komplexných scén pozadí spoločne bez spomalenia animácií, ktoré sa užívateľovi zobrazujú v popredí. JavaFX vlákna sú odlišné od Swingových, načo si treba dávať pozor pri písaní JavaFX kódu vo Swingových aplikáciách. Napríklad prenos dát by sa mal odohrávať na rovnakom vlákne ako užívateľské rozhranie [KOPECKY, 2015] [TOPLEY, 2010],
- Vykresľovacie vlákno Prism – Vlákno spracováva udalosti oddelene. Vykresľuje jeden objekt, zatiaľ čo sa spracováva ten druhý. To predstavuje veľkú výhodu pri moderných systémoch s viacerými procesormi,
- Media vlákno – Vlákno beží na pozadí a synchronizuje všetky objekty cez Scene Graph s použitím JavaFX aplikačného vlákna.

2.2.5 Media Engine

Media engine je dostupný cez `javafx.scene.media` a podporuje audio média (prehrávanie zvuku) vo formáte MP3, AIFF[8], WAV[9] a vizuálne média vo formáte FLV[10]. Funkcionalita funguje na troch komponentoch:

- Media Object – reprezentuje súbor, na ktorom je médium uložené,
- MediaPlayer – prehráva dané video,
- MediaView – vložením MediaView do scény, zobrazíme médium v okne aplikácie.

[7] Abstract Window Toolkit

[8] Audio Interchange File Format

[9] Waveform Audio File Format

[10] Flash Video

2.2.6 Web Engine

Web engine je založený na Webkit (open source vyhľadávač), ktorý podporuje HTML, CSS, JavaScript, DOM[11] a SVG formáty. Zabezpečuje plnú funkcionalitu vyhľadávanie cez API. Vyhľadávací komponent sa skladá z nasledujúcich tried:

- WebEngine – zabezpečuje vyhľadávanie webových stránkach,
- WebView – potrebné pre pridávanie metód, efektov. Predstavuje nadstavbu triedy Node [ORACLE, 2013a].

[11] Document Object Model

3. Základné grafické prvky

Podľa stromovej štruktúry z kapitoly 2.2.1 a obrázku 3 je možné základné grafické prvky popísané v tejto kapitole zoradiť do najnižšej vrstvy, teda do vrstvy „leaf node“. Jednotlivé prvky reprezentujú vlastnosti pre určitú komponentu ako napríklad vloženie obrázka alebo nastavenie štýlu písma pre text v komponente.

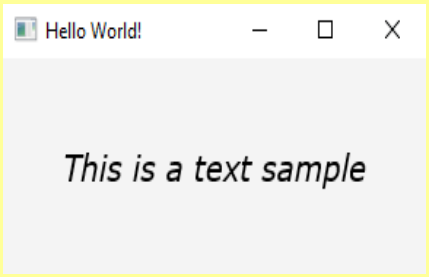
3.1 Font

Na nastavenie štýlu písma sa využíva inštancia triedy `javafx.scene.text.Font`. JavaFX umožňuje viacero možností ako priradiť štýl textu, a to buď výberom zo štýlov implementovaných vo vývojovom prostredí teda použiť implicitný štýl pre text, použitím kaskádových štýlov alebo vložením unikátneho štýlu, vo formáte TrueType font (.ttf) alebo OpenType (.otf), do zložky `resources/fonts` v projektovom adresári.

Pre font je možné nastaviť textu napríklad veľkosť a typ písma, teda normálny text, tučný a kurzíva.

Výpis 1 Font

```
import javafx.scene.text.Font;
import javafx.scene.text.FontPosture;
import javafx.scene.text.Text;

Text t = new Text();
t.setText("This is a text sample");
t.setFont(Font.font
      ("Verdana", FontPosture.ITALIC, 20));
```



3.2 Image

Pred začatím tvorby obrázku je nutné vytvoriť node ImageView. Vďaka nemu je možné zobraziť Image objekt, do ktorého bol najprv nahratý obrázok zo súboru, ku ktorému bola zadaná cesta ako String, URL alebo InputStream. JavaFX podporuje obrázky vo formáte BMP, PNG, JPEG alebo GIF. Pri vkladaní obrázku je možné nastaviť veľkosť obrázka. Pokiaľ veľkosť nenastavíme, vloží sa obrázok vo veľkosti akú má obrázok nastavená v súbore [SHARAN, 2015].

Výpis 2 Image

```
ImageView obrazok = new ImageView(new Image("/zdroje/planhry.png",
400, 250, false, false));
```

4. Základné komponenty

„Branch node“ predstavuje v stromovej štruktúre uzol s potomkami, teda komponenty, ktorým je možné priradiť ďalšie prvky spomínané v predchádzajúcej kapitole 3. O každom takomto komponente bude v tejto kapitole bližší popis s jeho vlastnosťami, základnými metódami, ktoré poskytuje a handlerom udalostí, ktorý urobí tento komponent aktívnym po užívateľovom kliknutí.

Pred popisom základných komponent sa čitateľ oboznámi s postupom tvorby menu rôznymi spôsobmi.

- MenuBar
- MenuItem
    - Menu
    - CheckMenuItem
    - RadioMenuItem
    - CustomMenuItem
        - SeperatorMenuItem
- ContextMenu

4.1 Menu

List položiek, ktoré tento komponent ponúka, sa užívateľovi nezobrazujú automaticky, ale sám si o nich musí požiadať kliknutím na ikonu. Po rozkliknutí sa zobrazia jednotlivé položky, z ktorých si je možné vybrať iba jednu. Menu je v menuBar zoskupované do nasledujúcich kategórií:

- Menu na tvorbu subpoložiek,
- MenuItems na tvorbu aktívneho tlačidla,
- RadioButtonItem na výber jednej z ponúkaných možností,
- CheckMenuItem na výber požiadaviek užívateľa pre danú aplikáciu.

Jednotlivé časti je možné od seba odlíšiť deliacou čiarou pomocou triedy SeperatorMenuItem. V aplikácií o Červenej čiapočke je použité dvojité zobrazenie menu. Na zobrazenie novej hry a ukončenie prebiehajúcej hry nielen kliknutím myšou na komponent ale aj použitím klávesových skratiek (viď obrázok 4 v nasledujúcej podkapitole.)

4.2 MenuBar

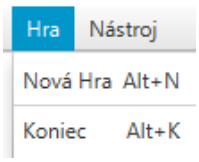
Komponent predstavuje lištu, na ktorej sa nachádza menu. Zvyčajne sa zakresľuje v hornej časti okna aplikácie a obsahuje tú časť menu, ktorú užívateľ uvidí hneď ako prvú pri spustení aplikácie. Jednotlivé menu položky obsahujú ďalšie menuItem, ktoré predstavujú aktívne tlačidla, po ich stlačení sa niečo udeje.

Výpis 3 MenuBar

```
MenuBar menuBar = new MenuBar();

Menu hraMenu = new Menu("Hra");
Menu nastrojMenu = new Menu("Nástroj");

MenuItem nastroj = new MenuItem("Nápoveda");
MenuItem novaHra = new MenuItem("Nová Hra");
MenuItem koniec = new MenuItem("Koniec");

hraMenu.getItems().addAll(novaHra,
                new SeparatorMenuItem(), koniec);
nastrojMenu.getItems().addAll(nastroj);
menuBar.getMenus().addAll(hraMenu, nastrojMenu);
```

4.3 SeparatorMenu

SeparatorMenu rozdelí komponenty vnútri menu oddeľovacou čiarou, pre jednoduchšiu vizualizáciu užívateľovi.



Obrázok 4 Menu

4.4 MenuItems

MenuItems obsahuje komponenty, na ktoré keď užívateľ klikne niečo sa udeje. Je zložený z tlačidla, takže rovnako ako komponent Button, popísaný v nasledujúcej kapitole, aj na tento komponent sa použije handler udalosti pomocou metódy `.setOnAction()`. Zároveň je možné použiť kombináciu tlačidiel na klávesnici pre zjednodušený prístup, vykonanie určitého úkonu.

Výpis 4 MenuItems

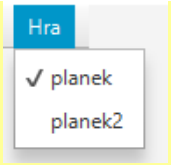
```
MenuItem koniec = new MenuItem("Koniec");
koniec.setAccelerator(KeyCombination.keyCombination("Alt+k"));
koniec.setOnAction(new EventHandler<ActionEvent>()
{
            @Override
            public void handle(ActionEvent t)
            {
                Platform.exit();
            }
});
```

4.5 CheckMenuItem

V menu si užívateľ môže taktiež vybrať z viacerých možností podľa jeho požiadaviek, čo chce mať zobrazené v okne aplikácie. Pomocou metódy `.setSelected(true)` sa ešte pred spustením aplikácie označí výber položky symbolom „fajka“. V prípade ďalšej práce s označenou položkou, sa využíva handler udalostí rovnakým spôsobom, ako v ostatných komponentoch.

Výpis 5 CheckMenuItems

```
CheckMenuItem planek = new CheckMenuItem("planek");
planek.setSelected(true);
CheckMenuItem planek2 = new CheckMenuItem("planek2");
```

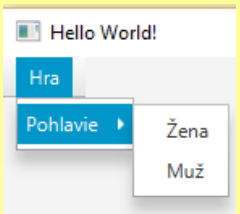


4.6 SubMenu

Pre vytvorené menu sa používa RadioMenuItem, teda užívateľovi je ponúkaný výber iba jednej možnosti zo všetkých ponúkaných. Aby jednotlivé možnosti boli schopné medzi sebou komunikovať, komponenty musia byť uložené v skupine ToggleGroup [ORACLE, 2014a].

Výpis 6 SubMenu

```
MenuBar menuBar = new MenuBar();
Menu hra = new Menu("Hra");
Menu pohlavie = new Menu("Pohlavie");
ToggleGroup group = new ToggleGroup();
RadioMenuItem muz = new RadioMenuItem("Muž");
muz.setToggleGroup(group);
RadioMenuItem zena = new RadioMenuItem("Žena");
zena.setToggleGroup(group);
pohlavie.getItems().addAll(zena, muz);
hra.getItems().addAll(pohlavie);
menuBar.getMenus().add(hra);
```



4.8 Label

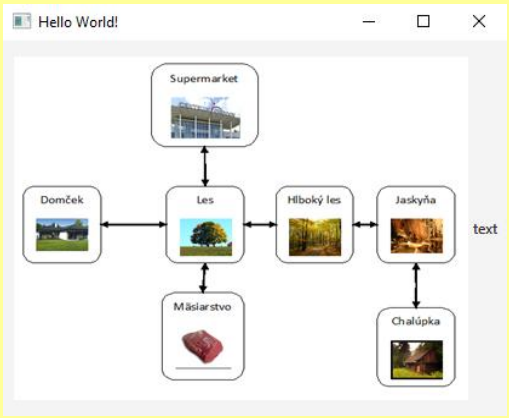
Komponent Label, predstavuje pasívny text, ktorý sa najčastejšie používa ako popis pre iný komponent. Pre neaktívny komponent sa handler udalostí nepridľuje. JavaFX nám ponúka tri možnosti zápisu:

- Label() – vytvorenie prázdneho komponentu. Pokiaľ chceme takto vyplniť aj text, je potrebné použiť metódu `.setText(String text)`. V prípade použitia obrázka metódu `.setGraphic(Node graphic)`,
- Label(String text) – vytvorenie komponentu s textom,
- Label(String text, Node graphic) – vytvorenie komponentu s textom a s obrázkom.

Label ma implicitne nastavený typ písma. Pomocou metódy `.setFont()` vie programátor zmeniť štýl aj veľkosť písma.

Výpis 7 Label

```
import javafx.scene.control.Label;
import javafx.scene.image.Image;
import javafx.scene.image.ImageView;

Label label1 = new Label("text");

ImageView image = new ImageView
(new Image("/zdroje/planek.jpg"));
Label label2 = new Label
                ("text", image);

Label label3 = new Label();
label3.setText("text");
label3.setGraphic(image);
label3.setFont(new Font("Arial", 20);
```



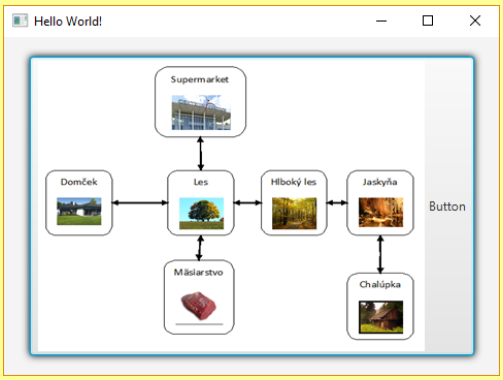
Samotný label nie je schopný rozlíšiť aký veľký text má byť v komponente. Celý text vkladá do jedného riadku. Pokiaľ vkladáme veľký objem textu a okno aplikácie je príliš malé, užívateľ uvidí iba jeho časť a nie celý text. Z toho dôvodu je vhodné použiť metódu `.setWrapText(true)`, ktorá zabezpečí rozdelenie textu na konci okna aplikácie [ORACLE, 2014b].

4.9 Button

Button predstavuje tlačidlo, ktorého funkciou je vykonať nejakú činnosť, ak na neho klikne užívateľ. Tlačidlo však poskytuje aj ďalšie možnosti ako napríklad po prejdení kurzorom myši po komponentu dôjde k jeho zafarbeniu. Vieme mu prideliť text, obrázok alebo oboje naraz. Vo výpise 8 sú využité obidve možnosti súčasne, pridelený text aj s fotkou. Základná činnosť nastavená na tlačidlo sa nastavuje po kliknutí pomocou metódy `.setOnAction()`. Samostatná činnosť sa vykonáva v metóde `handle(ActionEvent event)`. V nasledujúcom príklade sa vypíše v konzole slovo „text“. Metóda `MouseEvent.MOUSE_ENTERED` už z názvu hovorí, že nastavením kurzora na tlačidlo, sa tlačidlo ohraničí modrou farbou [ORACLE, 2014c].

Výpis 8 Button

```
import javafx.scene.effect.DropShadow;

Image image = new Image("/zdroje/fotka.jpg");
DropShadow shadow = new DropShadow();
Button button1  = new Button("Button");
button1.setGraphic(new ImageView(image));
button1.setOnAction(new EventHandler<ActionEvent>()
     {
         @Override
         public void handle(ActionEvent event)
         {
         System.out.println("text");
         }
     });

button1.addEventHandler
(MouseEvent.MOUSE_ENTERED,
(MouseEvent e) -> {
    button1.setEffect(shadow);
   });
```



Na základe priradenia rôznych štýlov, vieme jednoducho zmeniť funkciu tlačidla. JavaFX nám ponúka využitie CSS štýlov, ktorých popis tvorby je popísaný v kapitole 7.4.

4.10 CheckBox

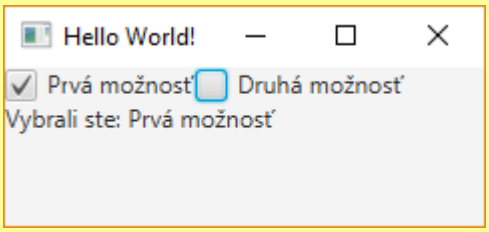
CheckBox je komponent, ktorý nám umožňuje výber z viacerých možností, kde je možné označiť všetky možnosti zároveň. Graficky je komponent znázornený ako štvorec s popisom. Existuje trojité označenie, buď je daná možnosť

- Vyberaná - v políčku sa objaví „fajka“,
- Nevybraná - políčko ostane prázdne,
- Nedefinovaná - v políčku sa objaví znak spojovník „-“.

Príkladom môže byť výber športu, ktorý má užívateľ rád. Človek má rád viacero druhov športu. Vo výpise 9 sú užívateľovi ponúknuté dve možnosti a jedno tlačidlo. V prvom riadku priradíme pre CheckBox popis „Prvá možnosť“, rovnako aj pre druhú možnosť. Druhému tlačidlu sme priradili nedefinovaný stav metódou `.setIndeterminate(true)`. Prvému tlačidlu je nastavené `ActionEvent()`. Po označení prvej možnosti, sa v komponente label zobrazí text „Prvá možnosť“, v druhom prípade sa vypíše „Druhá možnosť“. Pokiaľ vyberieme obidve možnosti, vypíšu sa obidva texty naraz [ORACLE, 2014d].

Výpis 9 CheckBox

```
Label label1 = new Label(" Prvá možnosť");
Label label2 = new Label ("Druhá možnosť");

CheckBox box = new CheckBox("Prvá možnosť");
CheckBox box2 = new CheckBox("Druhá možnosť");
Box2.setIndeterminate(true);

FlowPane root = new FlowPane();
root.getChildren().addAll(box, box2, label);

box.setOnAction((event) -> {

 if (box.isSelected()){
        root.getChildren().add(label1);
    }
    else {
        root.getChildren().remove(label1);
    }
});
box2.setOnAction((event) -> {
    if (box2.isSelected()){
       root.getChildren().addAll(label2);
    }
    else {
        root.getChildren().remove(label2);
    }
});
```



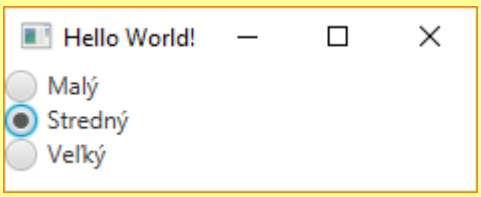
4.11 RadioButton

RadioButton je komponent, ktorý umožňuje výber jednej z viacerých možností v rovnakom čase. Existujú iba dve možnosti, buď je tlačidlo označené alebo nie je. Príkladom môže byť tlačidlo na výber pohlavia, teda muž alebo žena. Nemôže sa stať, že jedna osoba bude mužom aj ženou zároveň. V grafickej reprezentácií sa symbol líši od CheckBoxu. RadioButton je reprezentovaný krúžkom s popisom. Pri jeho označení, sa krúžok zafarbí. V checkBoxe označené tlačidlo, symbolizuje znak „fajka“.

Komponent je možné vytvárať pomocou konštruktoru bez parametru a následne pomocou metódy `.setText()` mu priradiť názvy jednotlivých ponúkaných možností alebo vytvoriť konštruktor s pridelenými parametrami. Už pri programovaní nastavíme, ktorá možnosť bude zaznačená na začiatku, a to metódou `.isSelected(true)`. Aby sme zabezpečili komunikáciu medzi komponentmi je nutné použitie ToggleGroup. Táto skupina nám zabezpečí, že iba jednu z ponúkaných možností v skupine je možné označiť, iba jedna možnosť bude označená [ORACLE, 2014e].

Výpis 10 RadioButton

```
ToggleGroup group = new ToggleGroup();
RadioButton rb1 = new RadioButton("Malý");
rb1.setToggleGroup(group);
rb1.setUserData("Malý");
rb1.setSelected(true);
RadioButton rb2 = new RadioButton("Stredný");
rb2.setToggleGroup(group);
rb2.setUserData("Stredný");
RadioButton rb3 = new RadioButton("Veľký");
rb3.setUserData("Veľký");
rb3.setToggleGroup(group);
}
});

group.selectedToggleProperty()
      .addListener(new ChangeListener<Toggle>()
{
@Override
public void changed(ObservableValue<? extends Toggle> observable,
      Toggle oldValue, Toggle newValue)
{
if (group.getSelectedToggle() != null)
      {
            System.out.println(group.getSelectedToggle()
                  .getUserDat().toString());
      }
}
});
```

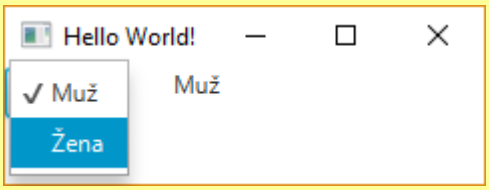


4.12 ChoiceBox

Komponent, umožňuje výber z viacerých možností. Označiť je však možné iba jednu. Tento komponent je vhodné použiť v prípade malého počtu možností, ako napríklad označenie pohlavie na muž alebo žena. Vytvoriť ho je možné použitím `observableArrayListu` ako je ukázané vo výpise 11 alebo vytvorením konštruktora bez parametru a pomocou metódy `.setItems()` mu priradiť hodnoty. Užívateľ, ktorý kurzor myši presunie na tento komponent, ukáže sa mu sprievodný obrázok s textom „Výber pohlavia“, vďaka použitia metódy `.setTooltop(new Tooltip())` [ORACLE, 2014f].

Výpis 11 ChoiceBox

```
final ChoiceBox<String> box = new ChoiceBox<String>
      (FXCollections.observableArrayList("Muž", "Žena"));
final String[] pohlavie = new String[] {"Muž", "Žena"};
box.setTooltip(new Tooltip("Výber pohlavia"));
box.getSelectionModel().selectedIndexProperty()
      .addListener(new ChangeListener<Number>()
{
public void changed(ObservableValue ov,
      Number value,Number new_value)
      {
            label.setText(pohlavie[new_value.intValue()]);
      }
});
```



4.13ComboBox

Komponent, ktorý umožní užívateľom výber jednej z viacerých možností. ComboBox a ChoiceBox, popísaný v predchádzajúcej podkapitole má takmer rovnaké vlastnosti. ComboBox je výhodné použiť vtedy, ak ponúkneme užívateľovi výber z väčšieho množstva možností. Automaticky sa pridá panel, v ktorom je možné posúvať sa medzi jednotlivými možnosťami. Túto schopnosť posúvania ale komponent ChoiceBox neponúka.

ComboBox je možné vytvoriť dvomi spôsobmi. Vytvorením inštancie triedy ComboBox a následným vpísaním komponent do observable listu, teda rovnakým spôsobom ako v predchádzajúcej podkapitole. Druhou možnosťou, ktorú JavaFX 8 ponúka, je použitím konštruktora bez parametru a zavolaním metódy `.setItems()`. Kedykoľvek pomocou `.getItems().addAll()` je možné pridať ďalšie možnosti do zoznamu. Metódou

`.setVisibleRowCount(3)` obmedzíme počet možností, ktoré sa majú zobraziť. Ďalšie možnosti sa zobrazia až po posunutí v panely. Počiatočnú možnosť, ktorá bude označená je možné nastaviť metódou `.getSelectionModel().select(0)`, kde číslovanie jednotlivých možností začína od čísla 0. Implicitne má komponent nastavené, že užívateľ nie je schopný meniť text jednotlivých možností. Toto obmedzenie umožňuje zmeniť metóda `.setEditable(true)` [ORACLE, 2014g].

Výpis 12 ComboBox

```
ObservableList<String> combodata =
FXCollections.observableArrayList("jdi", "seber", "vyhod");
combobox.getItems().addAll(combodata);
final String[] aa = new String[] {"Jdi", "seber", "vyhod"};
combobox.setVisibleRowCount(3);
combobox.getSelectionModel().select(0);
combobox.setEditable(true);
combobox.getSelectionModel().selectedIndexProperty()
      .addListener(new ChangeListener<Number>()
{
@Override
public void changed(ObservableValue<? extends Number> observable,
            Number oldValue, Number newValue)
      {
            label.setText(aa[newValue.intValue()]);
      }
});
```

JavaFX ponúka aj možnosť vložiť jednotlivé možnosti výberu naraz bez použitia observableListu, ako to bolo použité v predchádzajúcom výpise.

```
combobox.getItems().addAll("Jdi", "seber", "vyhod");
```

4.14 ListView

Komponent obsahujúci list položiek, z ktorých si môže užívateľ vybrať jednu z ponúkaných možností. Pokiaľ máme viac položiek automatický sa vygeneruje ScrollBar (podrobnejšie v kapitole 4.17). List položiek je možné zobrazovať horizontálne alebo vertikálne pomocou metódy `.setOrientation(Orientation.HORIZONTAL)` [ORACLE, 2014h].

Výpis 13 ListView

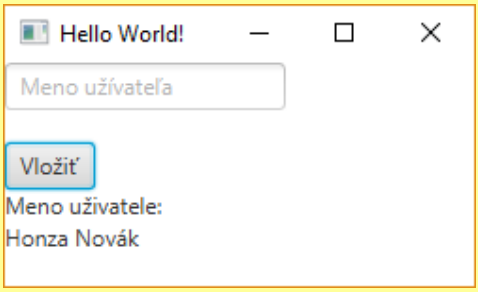
```
ListView<String> list = new ListView<>();
ObservableList<String> polozky =
FXCollections.observableArrayList("Polozka 1", "Polozka 2");
list.setItems(polozky);
```

4.15 TextField

Komponent, ktorý komunikuje so samotným užívateľom po zadaní textu užívateľa do komponentu. Aplikácia tento text dokáže prečítať a ďalej s textom pracovať. Ako pozadie komponentu sa implicitne nič nenastaví, iba prázdne políčko. Ideálne je vložiť text s bližším popisom čo sa od užívateľa očakáva, aká informácia, jeho meno, priezvisko, vek a podobne. Príkaz `.setPromptText()` ukazuje iba sprievodný text, po kliknutí užívateľa do políčka sa text automaticky sám vymaže. Užívateľ vložil text, príkazom `.getText()`, získa daný text a ďalej s ním vie pracovať. Avšak sprievodný text, promt text, nie je možné získať a použiť ho ďalej v aplikácií. Vo výpise 14, po stlačení tlačidla, získa text a vloží ho do label komponentu. Metódou `.clear()` sa vymaže obsah a nahradí sa prázdnym políčkom.

Výpis 14 TextField

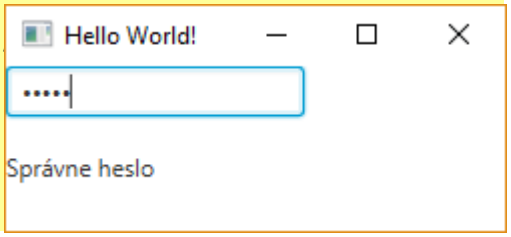
```
Label label = new Label();
TextField text = new TextField();
text.setPromptText("Meno užívateľa");
Button button = new Button("Vložiť");
button.setOnAction(new EventHandler<ActionEvent>()
{
@Override
public void handle(ActionEvent event)
      {
            label.setText(text.getText());
            text.clear();
      }
});
```



Pokiaľ použijeme tento komponent a užívateľ zadáva text, všetci ostatní pozorovatelia vidia zapísaný text. Pokiaľ však ide o súkromný text, akým môže byť napríklad heslo, nikto si nežiada aby tento text ostatní videli. V takomto prípade, je vhodné použiť komponent PasswordField. Znak vpísaný užívateľom sa ukáže ako „bodka“. Odporúča sa využiť metódu `.setText()`, ako informáciu pre užívateľa, či jeho vpísané heslo je rovnaké s heslom, ktoré uložil do aplikácie pri prvom prihlásení. Po potvrdení hesla tlačidlom „Enter“ na klávesnici sa užívateľovi ukáže, či jeho heslo bolo správne alebo nie, pomocou handleru udalosti [ORACLE, 2014i], [ORACLE, 2014j].

Výpis 15 PasswordField

```
Label label = new Label();
PasswordField password = new PasswordField();
password.setOnAction(new EventHandler<ActionEvent>()
{
@Override
public void handle(ActionEvent event)
      {
            if (!password.getText().equals("heslo")) {
                  label.setText("Nesprávne heslo");
            } else {
                  label.setText
                        ("Správne heslo");
            }
            password.clear();
      }
});
```



4.16Stage

Komponent Stage slúži na otvorenie nového okna v jednej aplikácií, teda aplikácia bude mať viac ako jedno výsledné okno. Príkladom môže byť okno nápovede, plán hry a úvodné okno pre prihlásenie užívateľa. Komponent nám iba vytvorí okno, bez žiadnej ďalšej súčasti. Následne je nutné vložiť scénu a panel, do ktorého vložíme ďalšie komponenty.

Výpis 16 Stage

```
Stage dialog = new Stage();
dialog.setTitle("Plan hry");
dialog.initStyle(StageStyle.UTILITY);
BorderPane border2 = new BorderPane();
Scene scene2 = new Scene(border2, 400, 250);
dialog.setScene(scene2);
dialog.show();
```

4.17 ScrollBar

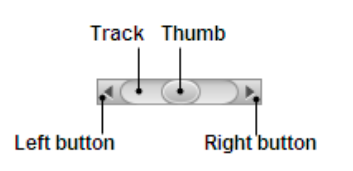
Pokiaľ aplikácia obsahuje veľké množstvo textu a ten sa nevojde do veľkosti okna, je potrebné použiť tento komponent. ScrollBar je tá časť panelu, vďaka ktorej sa vieme posúvať v okne vyššie, nižšie, vpravo, prípadne doľava. Obsahuje štyri časti:

- Pravé tlačidlo (Right button) – slúži na posúvanie doprava,
- Ľavé tlačidlo (Left button) – slúži na posúvanie doľava,
- Jazdec (Thumb) – tlačidlo, ktoré ukazuje aktuálnu polohu na stránke,
- Koľajnicu (track) – priestor, po ktorom sa palec pohybuje.

Metóda `.setMin()` a `.setMax()` slúži na definovanie minimálnej a maximálnej hodnoty, ktorá má byť použitá. Metóda `.setValue()`, určí na akej hodnote sa zobrazí komponent pri spustení aplikácie. Užívateľ si vie posúvať text kliknutím na krajné tlačidla, respektíve šípky, alebo kliknutím na jazdec v komponentu [ORACLE, 2014k], [ORACLE, 2014l].

Výpis 17 ScrollBar

```
ScrollBar bar = new ScrollBar();
 bar.setOrientation(Orientation.VERTICAL);
bar.setMin(0);
bar.setMax(100);
bar.setValue(50)
```

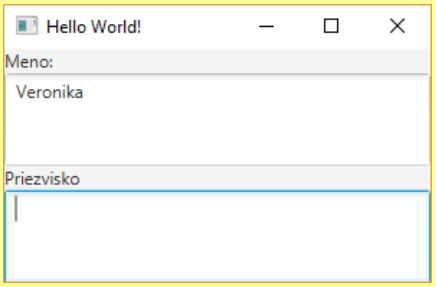


4.18 TextArea

Komponent, ktorý umožní užívateľovi vložiť text na obrazovku pri spustenej aplikácií. Aj tento komponent umožňuje použiť promt text, vysvetlený v predchádzajúcich podkapitolách. Slúži ako nápoveda pre užívateľa, aký text sa od neho očakáva napísať do komponentu [ORACLE, 2015].

Výpis 18 TextArea

```
Label label1 = new Label("Meno:");
TextArea text1 = new TextArea("");
Label label2 = new Label("Priezvisko");
TextArea text2 = new TextArea("");
```



5. Manažery pre rozmiestnenie komponent a práca s panelmi

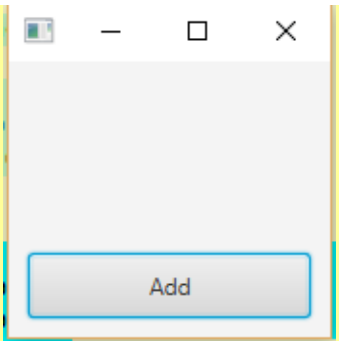
Layout panely, kontajnery, sú základom pre tvorbu GUI v jazyku JavaFX. Je možné použiť rôzne typy, druhy panelov: BorderPane, HBox, VBox, StackPane, GridPane, FlowPane, TilePane a AnchorPane s rôznymi možnosťami nastavenia pre každý element. Do každého panelu je následne možné vložiť ktorýkoľvek komponent z predchádzajúcej kapitoly.

5.1 AnchorPane

AnchorPane je kontajner, ktorý umožňuje ukotviť jednotlivé komponenty na určité miesto na obrazovke. Dokáže ju rozdeliť na viacero častí, pričom je možné aby sme určili, ktorý z komponent bude umiestnený navrchu ak dôjde k prekrývaniu. Pri úprave veľkosti okna, môže dôjsť k narušeniu usporiadanie komponent v kontajneri. Práve ukotvenie zabezpečí, že pri úprave veľkosti okna nedôjde k narušeniu, odsadeniu komponent. Pomocou príkazu `.setTopAnchor()` sa ukotví komponent v panely „hore“. Číslo v príkaze naznačuje v akej vzdialenosti od okraja sa má komponent umiestniť. Pomocou príkazu `.setMinSize(minSirka, minVyska)` sa nastaví veľkosť panela. V prípade, nenastavenia veľkosti, panel sa prispôsobí vloženým komponentom.

Výpis 19 AnchorPane

```
AnchorPane anchorPane = new AnchorPane();
Button button = new Button("Add");

AnchorPane.setRightAnchor(button, 10.0);
AnchorPane.setTopAnchor(button, 100.0);
AnchorPane.setLeftAnchor(button, 10.0);
AnchorPane.setBottomAnchor(button, 10.0);
anchorPane.getChildren().addAll(button);
```



Pri vytvorení nového projektu v Java FX Scene Builder sa tento panel zobrazí ako základný, primárny panel.

5.2 BorderPane

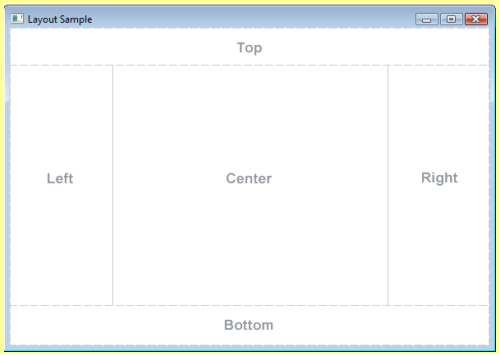
BorderPane je kontajner, ktorý rozdelí obrazovku na päť častí – na vrchnú (Top), spodnú (Bottom), pravú (Right), ľavú (Left) a strednú (Center) časť. Pokiaľ v našej aplikácií nie sú potrebné všetky časti, automaticky im bude pridelená nulová veľkosť a ich časť pripadne inej časti. Ak je okno aplikácie väčšie ako miesto nutné pre jednotlivé časti, extra miesto bude

pridelené strednej časti, t.j. do časti Center. Naopak, pokiaľ je okno aplikácie menšie ako miesto potrebné pre jednotlivé časti, spôsobí to prekrytie častí.

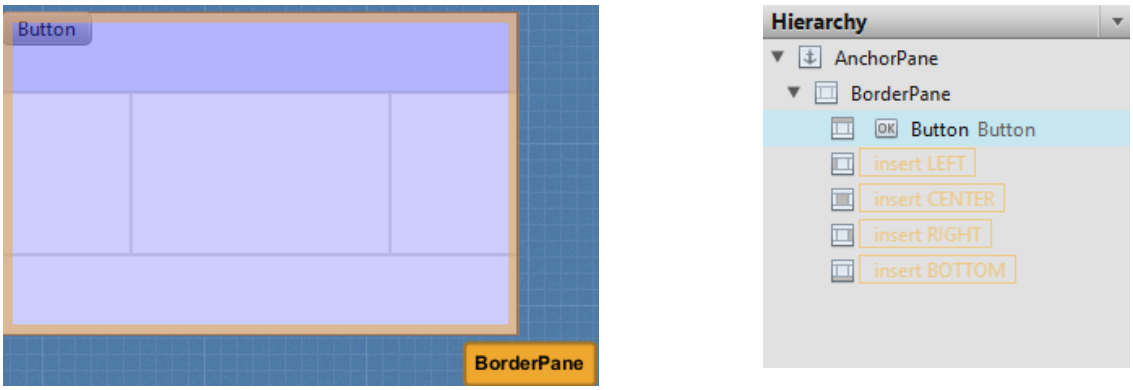
V nasledujúcom výpise je vytvorený jednoduchý BorderPane. Do každej časti je vložený `node` ako príklad. V aplikácií môže byť node nahradený elementom napríklad Button, Label alebo inými komponentmi popísanými v predchádzajúcej kapitole.

Výpis 20 BordePane rozloženie

```
BorderPane border = new BorderPane();

border.setCenter(node);
border.setBottom(node);
border.setLeft(node);
border.setRight(node);
border.setTop(node);
```



V JavaFX Scene Builder na ľavej strane programu, v časti Library, nájdeme rôzne druhy kontajnerov. Ako tretí v poradí je BorderPane. Kontajner presunieme za pomoci myši do vopred pripraveného panelu, AnchorPane. Program nám automaticky rozdelí panel do piatich častí, kde následne je možné vložiť ďalšie komponenty. Do hornej časti bolo vložené tlačidlo, Button. V časti „Hierarchy“ program aktuálne ukazuje ako máme poskladané elementy v kontajneri a v ktorej časti sa nachádza vložené tlačidlo.



Obrázok 5 BordePane a hierarchia vkladania panelov

5.3 HBox

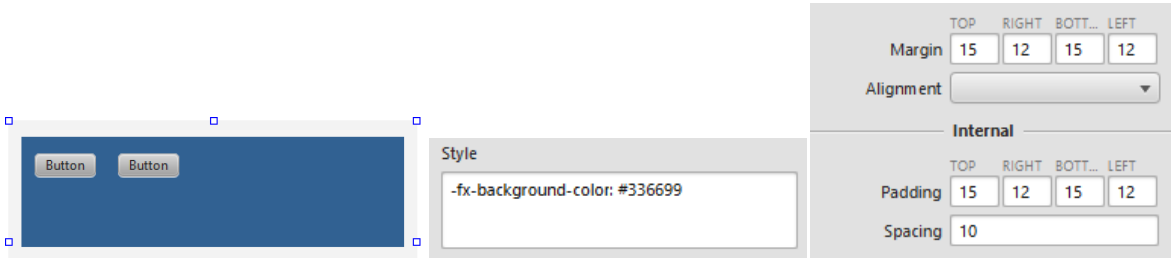
HBox, inak nazvaný HorizontalBox, je kontajner, ktorý dokáže uložiť elementy do jedného riadka horizontálne, nie však do viacerých riadkov. V prípade, že chceme mať výsledok vo

viacerých riadkoch, je potrebné vytvoriť nový hBox kontajner. Zároveň je možné nastaviť vzdialenosť medzi elementmi, nastaviť farbu pozadia.

Výpis 21 HBox

```
HBox hbox = HBox();
hbox.setPadding(new Insets(15, 12, 15, 12));
hbox.setSpacing(10);
hbox.setStyle("-fx-background-color:#336699");
hbox.setPrefSize(100, 100);
```

V JavaFXScene Builder vyberieme z časti Library Hbox a presunieme do hornej časti v BorderPane. V pravej časti Properties, časť Style ponúka nastavenie farby. V prípade obrázku je vložená modrá farba ako pozadie. *Margin* znamená odsadenie panelu od okraja okna. *Padding*, odsadenie komponenty vo vnútri panela z hornej, z pravej, z dolnej a z ľavej časti. Pomocou *Spacing* sa odsadí vzdialenosť jednotlivých komponent navzájom od seba.



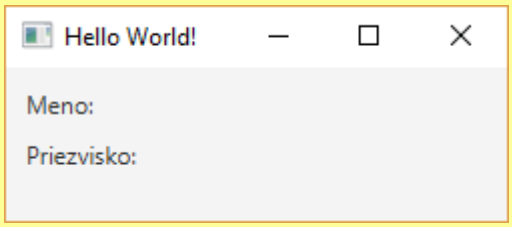
Obrázok 6 Hbox, nastavenie pozadia a preferencií panelu

5.4 VBox

VBox, inak nazvaný VerticalBox, je veľmi podobný HBoxu, ale VBox ukladá komponenty do jedného stĺpca, vertikálne. V prípade použitia viacerých stĺpcov, je znova potrebné vytvoriť nový kontajner VBox.

Výpis 22 VBox

```
VBox vbox = new VBox();
vbox.setPadding(new Insets(10));
vbox.setSpacing(8);
label = new Label("Meno:");
label0 = new Label("Priezvisko: ");

Scene scene = new Scene(vbox, 300, 300);
vbox.getChildren().addAll(label, label0);
```



5.5 StackPane

StackPane je kontajner, ktorý ukladá každý nový komponent na predchádzajúci vložený, tým dochádza k prekrytiu prvého komponentu. Príkladom môže byť orámovanie komponentu,

textu cez obrázok a podobne. Vo výpise 23 boli vytvorené dva komponenty, tlačidlo a text. Ako prvé je do panelu vložené tlačidlo, pretože má byť ako posledný komponent. Text, „otáznik“ má byť vrchný komponent, takže sa vkladá ako druhý v poradí. Pokiaľ by boli komponenty vložené v opačnom poradí, tlačidlo by prekrylo text.

Výpis 23 StackPane

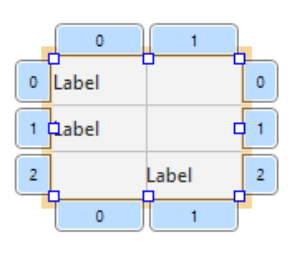
```
StackPane stack = new StackPane();
Button button = new Button();
TextArea text = new TextArea("?");
stack.getChildren().addAll(button, text);
```

5.6 GridPane

GridPane je kontajner, ktorý rozdelí okno na tabuľku, na viacero riadkov a stĺpcov. Čísluje sa od nuly, v poradí stĺpec, riadok. V tomto kontajneri je možné vložiť viac komponentov do jednej oblasti, ale budú sa prekrývať. Platí ten element, ktorý sa vloží do oblasti ako posledný. Umiestňované komponenty sú pridávané po riadkoch do jednotlivých stĺpcov a veľkosť stĺpca je prispôsobená najväčšiemu komponentu.

Výpis 24 GridPane

```
GridPane grid = new GridPane();
grid.setHgap(10);
grid.setVgap(10);
grid.setPadding(new Insets(0, 10, 0, 10));

Label label1 = new Label("Label:");
grid.add(label1, 0, 0); // 0 stlpec, 0 riadok
Label label2 = new Label("Label:");
grid.add(label2, 0, 1); //0 stlpec, 1 riadok
Label label3 = new Label("Label");
grid.add(label3, 1, 2); // 1 stlpec, 2 riadok
```

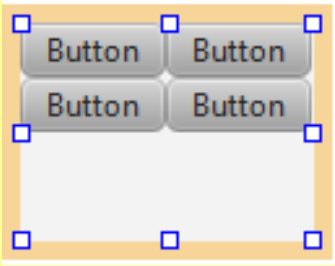


5.7 FlowPane

FlowPanel je kontajner, ktorý ukladá komponenty horizontálne do riadka alebo vertikálne do stĺpca. Samostatne je možné určiť použitie ukladania do stĺpcov alebo riadkov. Komponenty sa ukladajú za sebou, podľa poradia akým sú vložené do kontajnera. Pokiaľ sa ukladajú do riadka a veľkosť kontajnera je väčšia, ďalší element sa vloží do ďalšieho riadka.

Výpis 25 FlowPane

```
FlowPane flow = new FlowPane();

Label label1 = new Label("Label:");
Label label2 = new Label("Label:");
Label label3 = new Label("Label:");
Label label4 = new Label("Label:");

flow.getChildren().addAll(label1, label2, label3, label4);
```

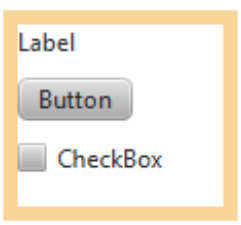


5.8 TilePane

TilePane je kontajner veľmi podobný panelu FlowPane. Všetky komponenty sa ukladajú do mriežky horizontálne alebo vertikálne a zároveň majú rovnakú veľkosť. Medzi jednotlivými komponentmi sa automaticky pridá malá medzera. Na nasledujúcom obrázku, sú v panely uložené komponenty vertikálne [ORACLE, 2013c].

Výpis 26 TilePane

```
TilePane tile = new TilePane();
Button button = new Button();
CheckBox box = new CheckBox();

TextArea text = new TextArea("?");
tile.getChildren().addAll(box, button);
```

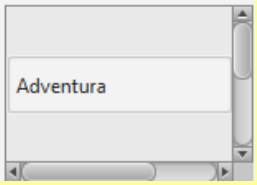


5.9 ScrollPane

Kontajner, ktorý sa využíva pokiaľ panel obsahuje veľké množstvo textu. Veľkú výhodu, ktorú ponúka samotný jazyk JavaFX, ako náhle je v panely viac textu, automaticky sa vloží ScrollBar (bližšie informácie v kapitole 4.17). Každý ScrollPane sa skladá z jedného vertikálneho a jedného horizontálneho komponentu Scrollbar.

Výpis 27 ScrollPane

```
ScrollPane pane = new ScrollPane();
TextField text = new TextField("Adventura");
pane.setContent(text);
```



6. Hlavné okno aplikácie

V tejto kapitole bude popísaný vzorový príklad ako by mohla vyzerať výsledná grafická reprezentácia hry o Červenej Čiapočke zo základného kurzu 4IT101 Programovanie v Jave pomocou jazyka JavaFX. Každý komponent a panel, ktoré boli alebo je možné použiť pri tvorbe GUI pre úspešné odovzdanie práce boli vysvetlené v predchádzajúcich kapitolách.

Okrem toho, že príklad adventúry z grafickým užívateľským rozhraním učí študenti používať prvky jazyka JavaFX, využíva aj návrhový vzor Observer. Realizáciu tohto návrhového vzoru čitateľ nájde v knihe „Návrhové vzory“ od Rudolfa Pecinovského na strane číslo 375. [PECINOVSKÝ, 2007].

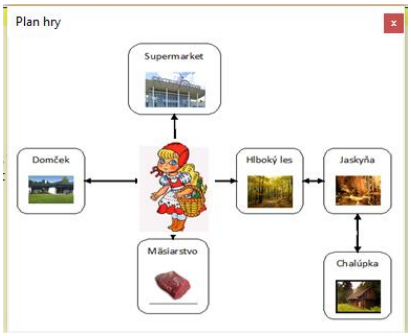
6.1 Výsledná aplikácia v JavaFX

Celá aplikácia sa skladá z dvoch častí, z dvoch okien. Do prvého okna zasahuje samotný užívateľ, hráč adventúry, zadávaním príkazov s cieľom vyhrať hru, teda dostať sa až k babičke do chalúpky. Do druhého okna aplikácie užívateľ nezasahuje. Toto okno slúži užívateľovi iba ako návod, ako sa dostať k babičke. Okno sa aktualizuje samostatne a ukazuje aktuálnu polohu Červenej čiapočky na ceste za svojim cieľom.

V prvom okne aplikácie je použitý BorderPane. Panel umožňuje rozdeliť obrazovku na päť častí, v našom prípade sa však využijú iba štyri časti. Samotný program automaticky upraví okno aplikácie podľa našich požiadaviek. V dolnej časti sú použité štyri komponenty: Label, ComboBox, TextArea a Button. Vďaka použitiu komponentu ComboBox, nie je nutné zadávať príkazy manuálne napísaním kódu do textového poľa, ale vybratím jedného z ponúkaných príkazov. V strednej časti je základ celej aplikácie, textová oblasť, v ktorej sa vypisujú jednotlivé texty hry. V ľavej časti sú umiestnené obrázky všetkých vecí, ktoré budú počas hry vložené do košíka. Do pravej časti je vložený list východov s aktuálnym zoznamom východov, kam sa je možné dostať z miestnosti, v ktorej sa hráč aktuálne nachádza.

Všetky komponenty sú navzájom prepojené. Po zadaní príkazu napríklad „jdi les“ sa hráč dostane do nasledujúcej miestnosti s názvom „les“. Zároveň v liste východov, v prvom okne aplikácie, dôjde k zmene, a to k prepísaniu východov, kam sa je možné pohybovať z aktuálnej miestnosti. Aktualizáciu jednotlivých komponent zabezpečujú triedy vydavateľ a predplatiteľ, bližšie popísané v kapitole 6.2.

V druhom okne aplikácie je vložený obrázok, vytvorený v inom grafickom editore. Po prejdení do inej miestnosti, sa obrázok Červenej Čiapočky presunie do inej miestnosti, na inú pozíciu, na základe predplatiteľa a vydavateľa zmeny aktuálneho priestoru.



Obrázok 7 Výsledná aplikácia: prvé okno

6.2 Predplatiteľ a Vydavateľ

V adventúre je nutné aktualizovať miestnosť, kde sa nachádza Červená Čiapočka a košík predmetov, ktoré nazbierala po ceste. Po prejdení do vedľajšej miestnosti sa musí aktualizovať list východov z danej miestnosti a následne na pláne hry sa Červená Čiapočka musí presunúť na inú pozíciu. Tým sa zabezpečí kompatibilita medzi oknami. V zvyšnej časti tejto kapitoly bude popísaný postup ako aktualizovať priestor kde sa postava nachádza. Zmena obsahu košíka sa implementuje rovnako ako zmena priestoru.

Na začiatku je potrebné vytvoriť triedu `PredplatitelZmenyAktualnihoProstoru`, ktorá obsahuje jednu metódu na aktualizáciu. Na túto triedu sa odkazujeme v grafickej časti aplikácie. Názov triedy, v ktorej sa nachádza panel s východmi, je potrebné doplniť o `implements PredplatitelZmenyAktualnihoProstoru`. NetBeans nám automaticky vytvorí v danej triede metódu `.aktualizujSe()` prebratú z tejto triedy.

Výpis 28 Predplatiteľ zmeny aktuálneho priestoru

```
public interface PredplatitelZmenyAktualnihoProstoru
{
    public void aktualizujSe();
}
```

Na začiatku triedy sa nachádza list typu String, do ktorého následne vložíme zoznam východov. Pomocou jednoduchého cyklu `for` sa načítajú všetky východy oddelené medzerou. Do listu sa vkladajú jednotlivé východy pod seba na základe medzery, ktorá ich od seba oddelí. Pokiaľ by sme takýmto spôsobom vkladali východy, veľkosť listu by sa zväčšovala a pridávali by sa nové východy k pôvodným. Preto je nutné na začiatku zavolať

metódu `data.clear()`, ktorá najprv vymaže pôvodné východy a následne načíta nový zoznam.

Výpis 29 Metóda aktualizujSe()

```
ObservableList<String>data;
data = FXCollections.observableArrayList();

@Override
    public void aktualizujSe()
    {
        String vychody =plan.getAktualniProstor().seznamVychodu();
        data.clear();
        String[] oddeleneVychody = vychody.split(" ");
        for (int i=1;i<oddeleneVychody.length; i++)
        {
            data.add(oddeleneVychody[i]);
        }
    }
```

Trieda `VydavatelZmenyAktualnihoProstoru` bude obsahovať tri metódy. Na túto triedu sa neodkazuje v grafickej časti ale v časti logiky. Triedu priestor a herný plán je potrebné implementovať s touto triedou. Na začiatku sa vytvorí list predplatiteľov ako zoznam nasledovne:

`List<PredplatitelZmenyAktualnihoPorostoru>seznamPredplatitelu`.

Metóda `zaregistruj(..)` slúži na zaregistrovanie predplatiteľa a jeho pridanie do zoznamu predplatiteľov pomocou jednoduchého príkazu `.add()`. Metóda `odregistruj(..)` predstavuje presný opak, teda vymazanie zo zoznamu pomocou príkazu `.remove()`.

Výpis 30 Vydávateľ zmeny aktuálneho priestoru

```
public interface VydavatelZmenyAktualnihoProstoru
{
    public void zaregistruj
           (PredplatitelZmenyAktualnihoPorostoru predplatitel);

    public void odregistruj
           (PredplatitelZmenyAktualnihoPorostoru predplatitel);

    public void upozorniPredplatitele();

}
```

Metóda `.upozorniPredplatitele()` je trošku zložitejšia. Už z názvu je jasné, že je potrebné upozorniť predplatiteľa ak sa niečo udeje. Pokiaľ dôjde k odobratiu alebo pridaniu veci do priestoru, je nutné upozorniť nato, že sa niečo udialo a aktualizovať panel. Jednoduchý výpis 31 vloží vec do priestoru a upozorni predplatiteľa. Keďže cieľom je aktualizovať panel, čo znamená, že ide o aktualizáciu grafickej komponenty, metódu `.aktualizujSe()` zavoláme v časti, kde sa vytvára samotný panel s východmi.

Výpis 31 Upozornenie predplatiteľa

```
public void vlozVec(Vec neco)
{
        veci.put(neco.getNazev(),neco);
        obsahProstoru.add(neco);
        upozorniPredplatitele();
}
```

7. JavaFX Scene Builder

JavaFX Scene Builder je jednoduchý dizajnový nástroj na tvorbu GUI jednoduchým presunutím komponenty na hlavné okno aplikácie za pomoci myši. Bez písania kódu je možné nastaviť farbu, veľkosť, štýl písma, vzdialenosť jednotlivých komponent v paneli a mnoho ďalších vlastností všetkých komponent. Samotný program nám automaticky vygeneruje FXML kód. Vďaka čomu sa rýchlo prepojí grafika s logikou celej aplikácie.

Aplikáciu Scene Builder je možné integrovať s vývojárskym prostredím NetBeans alebo Eclipse. Na VŠE sa zatiaľ vyučujú predmety v NetBeans. Z toho dôvodu je táto práca integrovaná iba s prostredím NetBeans.

7.1 Popis inštalácie nutných programov

Pred samotnou inštaláciou programu je potrebné skontrolovať základné požiadavky systému. Odporúča sa mať nainštalovanú minimálne JDK8, následne platformu Java SE[12] 8, ktorá obsahuje potrebné súčasti pre spustenie Java aplikácií a poslednú verziu NetBeans IDE[13] 8.0. Všetky spomínané inštalácie, by mali mať všetci študenti, ktorí absolvovali základný kurz programovania, už nainštalované na svojich počítačoch. V tejto podkapitole bude aj stručný návod pre prípad, že študenti tieto programy nainštalované nemajú. Nasledujúci popis je platný pre užívateľa vlastniaceho operačný systém Windows. Užívateľom iných operačných systémov je tento návod tiež užitočný, ale líši sa niekoľkými odlišnosťami. JDK8 je možné stiahnuť zo stránok firmy Oracle po akceptovaní licenčných podmienok pomocou nasledujúceho odkazu:

http://www.oracle.com/technetwork/java/javase/downloads/jdk8-downloads-2133151.html

Pomocou odkazu https://netbeans.org/downloads/ sa spustí inštalačný program vybratím technológie JavaSE a akceptovaním licenčných podmienok rovnakým spôsobom ako v predchádzajúcej inštalácií. Ešte pred samotným spustením inštalácie vyberieme možnost nainštalovať aj knihovnu JUnit.

[12] Standard Edition

[13] Integrated development environment

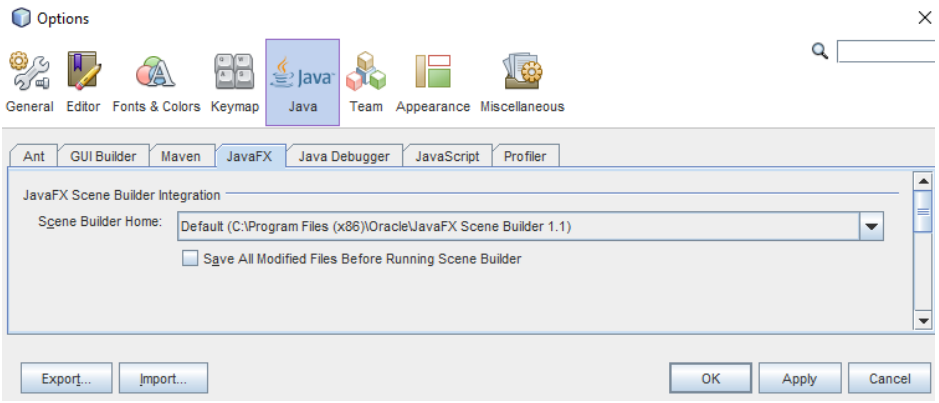
Po úspešnom ukončení všetkých predchádzajúcich inštalácií boli splnené potrebné požiadavky k samotnej inštalácii programu Scene Builder.

7.2 Popis inštalácie JavaFX Scene Builder

Nasledujúci webový odkaz poskytuje možnosť stiahnutia verzie 1.1 aj novšej verzie 2.0 po akceptovaní licenčných podmienok a vybratí správneho operačného systému, na ktorom pracuje počítač:

http://www.oracle.com/technetwork/java/javase/downloads/javafxscenebuilder-1x-archive-2199384.html

Po inštalácii JavaFX Scene Builder sa tento program nemusí sám do NetBeans priložiť. V samotnom programe NetBeans v lište menu a v časti Tools -> Options -> Java -> JavaFX je nutné skontrolovať toto prepojenie.



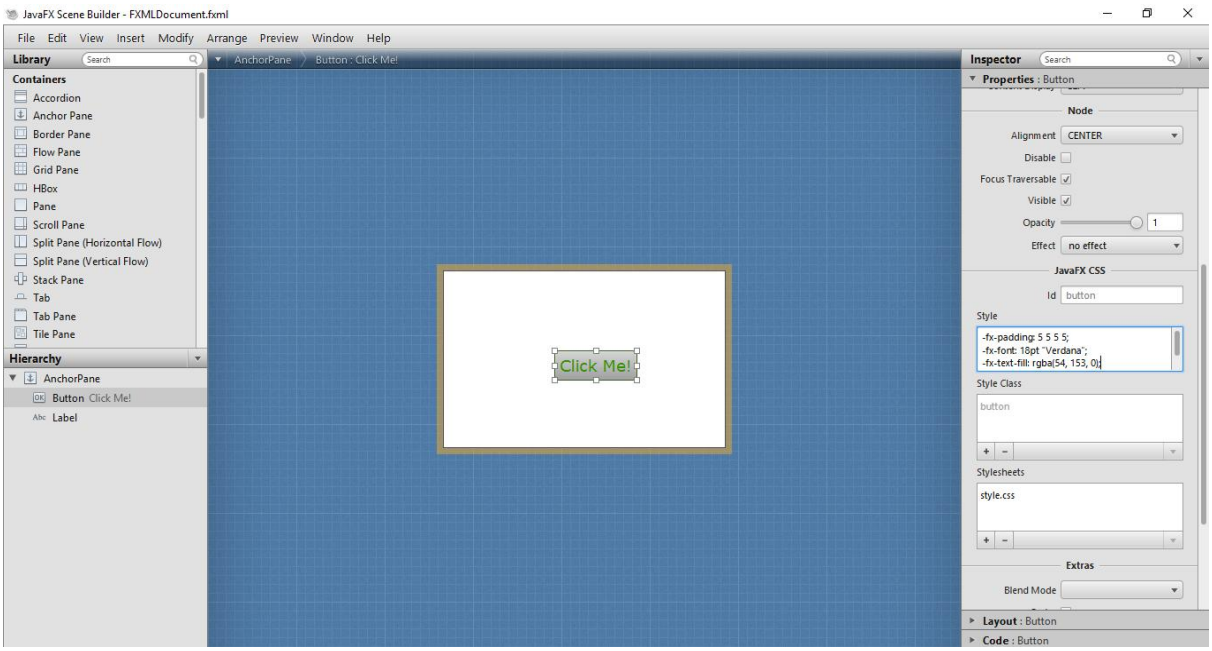
Obrázok 8 Pridanie Scene Builder do NetBeans

7.3 Popis nástroja

Po úspešnej inštalácií programu Scene Builder sa zobrazí základná obrazovka. Ľavá strana pod časťou Library, knihovňa, obsahuje jednotlivé grafické komponenty: kontajnery (containers), ovládače (controls), popus ovládače (popus controls), ovládače menu (menu controls), ostatné (miscellaneous), tvary (shapes) a grafy (charts). V druhej časti je zobrazená stromová štruktúra komponentov a hierarchia (Hierarchy) otvoreného dokumentu, ako sú jednotlivé komponenty usporiadané.

Pravá časť je rozdelená na tri menšie oblasti. V prvej časti, vlastnosti (properties), je možné nastaviť názov, popis, farbu, zarovnanie, štýl písma a využiť kaskádové štýly. V časti

rozvrstvenie (layout) sa nastavujú rozmery komponent, odsadenie od okraja a podobne. Posledná časť, kód (code) slúži na prepojenie dokumentu s ovládacou triedou. Tá by mala mať na začiatku FXML a na konci Controller. V tejto triede sa automaticky prepoja komponenty z Scene Builder s triedou za pomocou `@FXML`. Na základe tohto tagu, pridelíme kód, konkrétnemu komponentu. Podrobnejšie ako sa správne odkazovať je popísané v nasledujúcich kapitolách.



Obrázok 9 Popis nástroja Scene Builder

7.4 CSS

JavaFX 8 ponúka využitie kaskádových štýlov (CSS, Cascading Style Sheet) pre tvorbu farebného vzhľadu komponent. Umožňuje im priradzovať vlastnosti a tým urobiť aplikáciu lákavejšiu pre užívateľa. Základným štýlom pre JavaFX aplikácie je `modena.css` avšak každý užívateľ si vie vytvoriť vlastné štýly podľa svojich predstáv. Každý CSS súbor musí byť uložený s koncovkou .css a uložený v rovnakej zložke ako hlavná trieda celej aplikácie. Vzhľad pre všetky tlačidla sa začne aplikovať po priradení súboru do sceny aplikácie. JavaFX priradí štýl `.getStylesheets()` získaný zo súboru pomocou `.getResource()` a pridá do vopred pripravenej scény `.add(...)`.

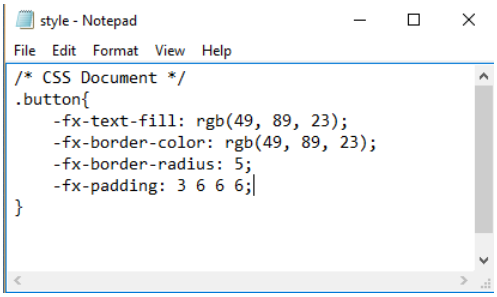
Výpis 32 Vloženie CSS súboru do scény

```
Scene scene = new Scene(Parent root);
scene.getStylesheets().add(getClass()
      .getResource("style.css")
      .toExternalForm());
```

Definícia kaskádových štýlov je tvorená z názvu štýlu, ktorý sa označuje aj ako selektor a z pravidiel, ktoré sú ohraničené zloženými zátvorkami „{}“. CSS vie taktiež rozlíšiť, či ide o nastavenie vzhľadu konkrétne pre jeden komponent alebo všeobecne pre všetky komponenty rovnakého typu. Pre konkrétny komponent je to selektor „`#`“ a pre všetky ostatné rovnakého typu selektor „bodka“ s popisom komponenty napríklad „`button`“.
Existuje dvojitý spôsob ako vytvárať kaskádové štýly. Na obrázku 9, v pravom dolnom rohu, sú tri malé „okienka“. Prvé s názvom Style, predstavuje prvý spôsob ako je možné vytvárať vzhľad. Definovanie štýlov, vlastností daného objektu týmto spôsobom začína znakom `-fx-`, ďalej pokračuje názvom a požadovanou hodnotou atribútu. Názov a hodnota sa oddelí dvojbodkou a každé nové pravidlo bodkočiarkou. Pomocou nasledujúceho príkazu pridelíme komponentu typ písma a jeho veľkosť `-fx-font: 18pt "Verdana"`.

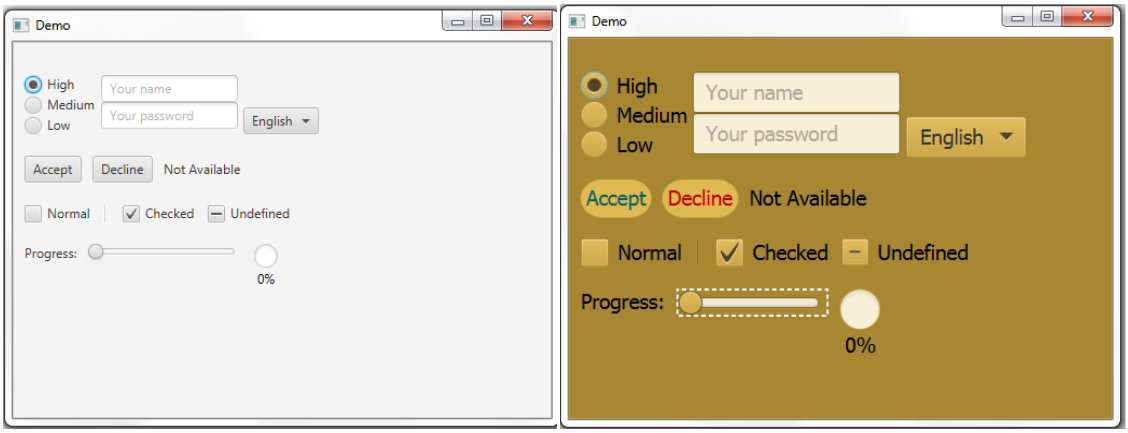
V druhom okne Style Class sa automaticky načíta komponent, ktorý je označený a pre ktorý sa vytvára vzhľad pomocou CSS. Pokiaľ sa do komponentu vkladajú nejaké iné informácie, text, zobrazia sa aj pridelené `id` všetkých jeho častí.

Posledná časť, Stylesheets, obsahuje druhý spôsob ako je možné vytvárať kaskádové štýly, a to pre všetky komponenty, ktoré patria do rovnakej rodiny, čo znamená, že patria do rovnakej scény (pre všetky tlačidla platí rovnaký štýl) načítaním .css súboru s vopred nastaveným štýlom vďaka tlačidlu „+“. Ak už je súbor priradený ale užívateľ požaduje zmeniť vzhľad komponenut, stačí kliknúť na malú šípku v pravom dolnom rohu okna, označiť „`open subor.css`“ a automaticky sa otvorí textový súbor, ktorý je možné meniť. Vytvorený textový súbor môže vyzerať nasledovne:



Obrázok 10 Style.css

Na nasledujúcich dvoch obrázkoch je zrejmé, že použitím CSS štýlov je vzhľad výslednej aplikácie pre užívateľa farebnejší a tým prížažlivejším. Na prvom obrázku je vzhľad základného štýlu, modena.css, druhý obrázok s použitím vlastných CSS štýlov.



Obrázok 11 Modena.css a použitie vlastných CSS štýlov. (Zdroj: [ORACLE, 2014m])

Príklad kódu niektorých komponent.

Výpis 33 CSS súbor

```
.root
{
	-fx-font-size: 14pt;
	-fx-font-family: "Tahoma";
	-fx-base: #DFB951;
	-fx-background: #A78732;
	-fx-focus-color:#B6A678;
	}
	.button1{
	-fx-text-fill: #006464;
	-fx-background-color: #DFB951;
	-fx-border-radius: 20;
	-fx-background-radius: 20;
	-fx-padding: 5;
	}
```

8. Hlavné okno aplikácie

V tejto kapitole bude popísaný vzorový príklad ako by mohla vyzerať výsledná grafická reprezentácia hry o Červenej Čiapočke zo základného kurzu 4IT101 Programovanie v Jave pomocou programu JavaFX Scene Builder. Všetky komponenty a panely, ktoré boli alebo je možné použiť pri tvorbe GUI pre úspešné odovzdanie práce boli vysvetlené v predchádzajúcich kapitolách.

8.1 Výsledná aplikácia použitím JavaFX Scene Builder

Aplikácia pomocou programu JavaFX Scene Builder vyzerá vzhľadovo rovnako. Boli použité presne tie isté komponenty ako v predchádzajúcom príklade, avšak grafické komponenty neboli programované napísaním kódu v programe NetBeans ale jednoduchým kliknutím myši v programe Scene Builder. Automaticky to naprogramuje kód pre grafiku priamo v NetBeans v triede FXMLDocument.

Výpis 34 Aplikácie v Scene Builder

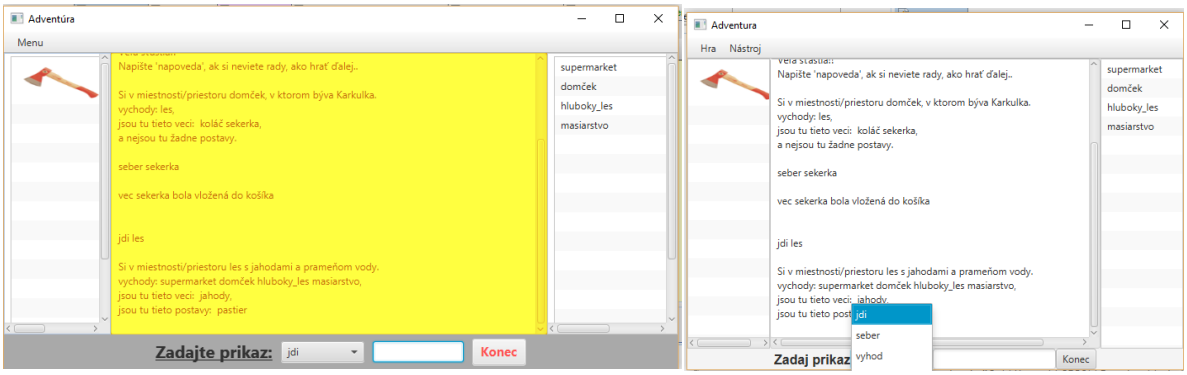
```
<BorderPane fx:id="borderPane" maxHeight="-Infinity" maxWidth="-
Infinity" minWidth="-1.0" prefHeight="400.0" prefWidth="800.0"
style="-fx-background-color: #daa520"xmlns:fx=
"http://javafx.com/fxml/1" xmlns="http://javafx.com/javafx/2.2"
fx:controller="thesisapplication.FXMLDocumentController">

<top>
    <MenuBar>
      <menus>
        <Menu mnemonicParsing="false" text="Menu">
          <items>
            <MenuItem mnemonicParsing="false"
           onAction="#menuNovaHra" text="Nová hra"
           fx:id="menuNovaHra" />
            <MenuItem mnemonicParsing="false"
            onAction="#menuNapoveda" text="Nápoveda"
            fx:id="menuNapoveda" />
            <MenuItem mnemonicParsing="false"
            onAction="#menuKoniecA" text="Koniec"
            fx:id="menuKoniec" />
           </items>
         </Menu>
       </menus>
     </MenuBar>
```

Každé okno aplikácie sa vytvára ako nový súbor, pre ktorý sa vytvorí ďalší FXMLDocument. V našom prípade sme museli vytvoriť dva FXML dokumenty, jeden ako aplikáciu pre užívateľa a druhý pre obrázok s plánom hry. V nasledujúcom kóde bol načítaný FXML súbor so všetkými komponentmi, ktoré boli do neho vložené.

Výpis 35 Odkaz na vytvorené okno v Scene Builder

```
private void planekLayout()
{
      try {
            FXMLLoader loader = new FXMLLoader((getClass()
                        .getResource("FXMLPlanek.fxml")));
            anchorPane = (AnchorPane) loader.load();

            Stage dialog = new Stage();
            Scene scene2 = new Scene(anchorPane);

            dialog.setScene(scene2);
            dialog.setTitle("Plán hry");
            dialog.show();

          }
      catch (IOException e) { //jednoduche odchytenie vynimky
            e.printStackTrace();
         }
}
```

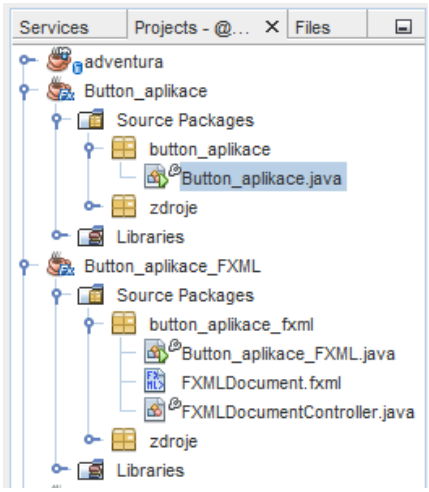
Porovnanie výsledného okna aplikácie s použitím kaskádových štýlov a bez nich.



Obrázok 12 Porovnanie výsledných aplikácií

9. Jednoduchá aplikace Button

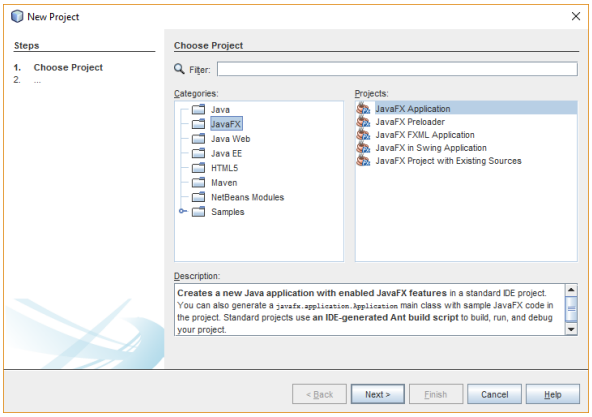
V tejto kapitole vytvoríme jednoduchú aplikáciu pomocou programu Scene Builder a s použitím nástroja FXML. Už z obrázka je zrejmé, že stromová štruktúra jednotlivých súborov je odlišná. Pokiaľ pri vytvorení funkčnej aplikácie pomocou programu NetBeans je potrebná iba jedna trieda, pomocou FXML sa vytvoria rovno tri triedy. Jedna pomocou ktorej sa aplikácia spúšťa, druhá pre zobrazenie automaticky vytvoreného kódu alebo úpravu FXML kódu a tretia pre prepojenie programu Scene Builder s programom NetBeans. V nasledujúcich podkapitolách čitateľ nájde celý popis kódu pre vytvorenie rovnakej aplikácie dvomi rôznymi spôsobmi.



Obrázok 13 Porovnanie stromovej štruktúry programu JavaFX a JavaFX Scene Builder

9.1 Pomocou programu Netbeans

Po otvorení nástroja NetBeans kliknutím na hornej lište `File`, a následne `New Project` sa na obrazovke objaví okno s piatimi možnými možnosťami výberu. Pri zakladaní nového projektu vyberáme defaultne prednastavenú možnosť tvorby projektu a teda kategóriu JavaFX a projekt JavaFX Application.



Obrázok 14 Založenie nového projektu JavaFX

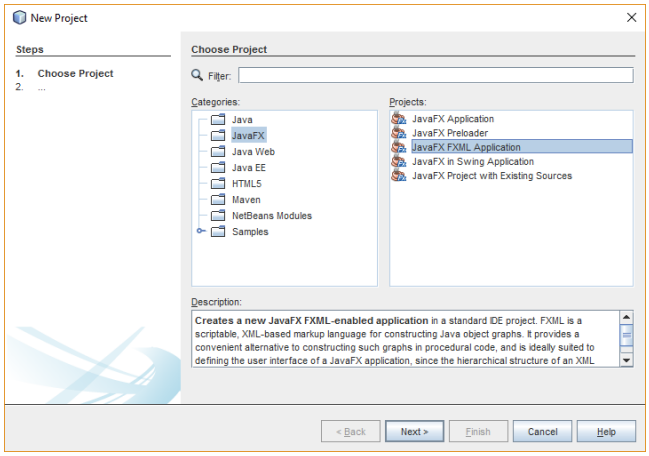
Po potvrdení výberu tlačidlom `Next`, v nasledujúcom okne je potrebné vyplniť `Project Name`, názov projektu, `Project Location` miesto kam sa bude projekt ukladať a skontrolovať či JavaFX Platform je nastavená na verziu JDK 1.8. Po stlačení tlačidla `Finish` sa vytvorí aplikácia.

Po spustení je možné vidieť jednu triedu s implementovaným kódom, ktorý obsahuje jedno tlačidlo, label, obrázok a panel do ktorého sa následne vložia spomínané komponenty. Po stisnutí tlačidla s obrázkom sa zobrazí v komponente Label text. Predpokladom na spustenie aplikácie je využitie potomka triedy `javafx.application.Application` a `javafx.stage.Stage`. Prvá trieda obsahuje metódu `.start()`, ktorá je potomkom triedy `Application` a obsahuje všetky grafické komponenty s ich handlery udalostí. Na vytvorenie okna sa využíva príkaz `Scene` do ktorého sme v nasledujúcom príklade vložili panel s potomkami, teda grafickými komponentmi. Metóda `main(String [] args)` zabezpečí spustenie celej aplikácie pomocou metódy `.launch()`, ktorá predstavuje externý spúšťač pre JavaFX aplikácie.

```
package button_aplikace;

import javafx.application.Application;
import javafx.event.ActionEvent;
import javafx.event.EventHandler;
import javafx.geometry.Pos;
import javafx.scene.Scene;
import javafx.scene.control.Button;
import javafx.scene.control.Label;
import javafx.scene.image.Image;
import javafx.scene.image.ImageView;
import javafx.scene.layout.AnchorPane;
import javafx.stage.Stage;

/**
 * @author Veronika Vallušová
 */
public class Button_aplikace extends Application
{
    Label label;

    @Override
    public void start(Stage primaryStage)
    {
        Anchor anchor = new AnchorPane();
        label = new Label(" ");
        label.setLayoutX(60);
        label.setLayoutY(134);
        label.setAlignment(Pos.CENTER);
        Button button1 = new Button();
        button1.setLayoutX(93);
        button1.setLayoutY(23);
        Image image = new Image("/zdroje/fotka.png");
        button1.setGraphic(new ImageView(image));
        button1.setOnAction(new EventHandler<ActionEvent>()
     {
         @Override
         public void handle(ActionEvent event)
         {
             label.setText
                     ("Klikol si na tlačitlo s obrázokom jahôd");
         }
     });
        anchor.getChildren().addAll(button1, label);
        Scene scene = new Scene(anchor, 320, 200);
        primaryStage.setScene(scene);
        primaryStage.show();
    }

    /**
     * @param args the command line arguments
     */
    public static void main(String[] args)
    {
        launch(args);
    }
```

9.2 Pomocou programu Scene Builder a využitím FXML

Pri zakladaní nového projektu pomocou Scene Builder vyberáme kategóriu JavaFX a projekt JavaFX FXML Application.



Obrázok 15 Založenie nového projektu JavaFX FXML Application

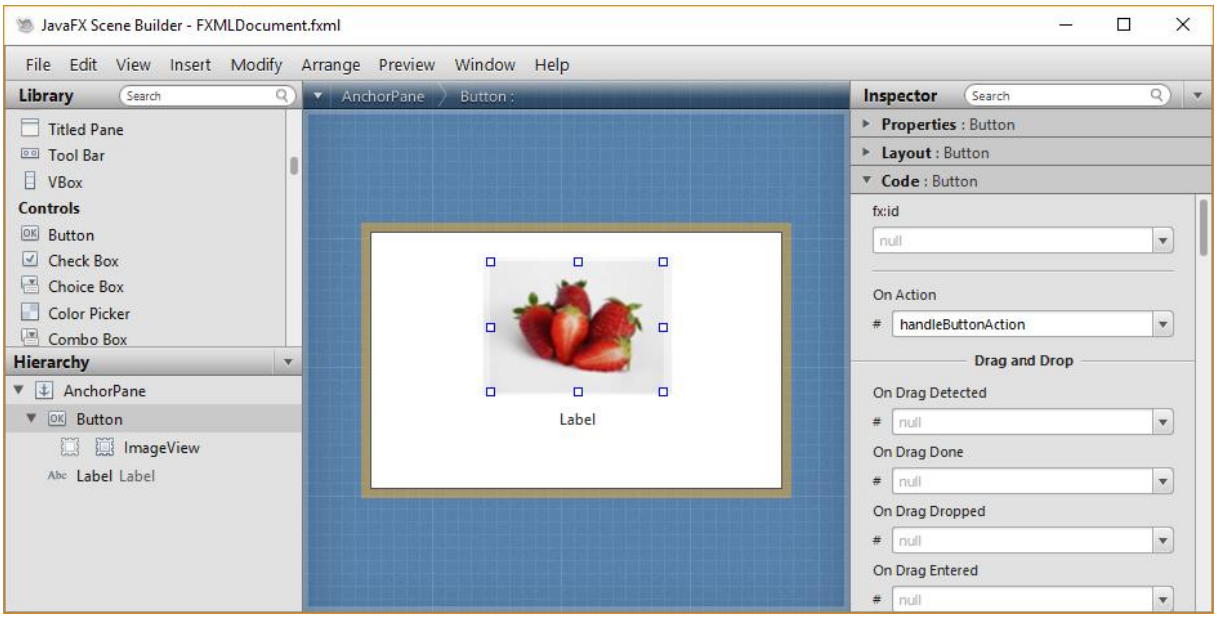
Po potvrdení vybraných možností sa vytvoria tri triedy. Jedna trieda pomocou ktorej sa aplikácia spúšťa, druhá pre zobrazenie automaticky vytvoreného kódu alebo pre úpravu FXML kódu a tretia pre prepojenie programu Scene Builder s programom NetBeans.

Prvú triedu užívateľ neupravuje. Ide o potomka abstraktnej triedy `Application`, ktorá implementuje metódu `start`. Dôležitá metóda, ktorej by mal užívateľ venovať pozornosť. Program NetBeans ju vygeneruje a pre užívateľa napíše automaticky. Pomocou FXMLLoader-u načíta FXML súbor a na jeho základe vytvorí formulár, scénu. Odkazuje sa na názov, meno zdroja a kde je uložená grafická časť aplikácie pomocou druhého programu Scene Builder. Ďalej sa v tejto triede nachádza metóda `main`, ktorá je nutná pre spustenie každej Java aplikácie. V jej tele je verejná statická metóda `launch`, ktorej definícia je už obsiahnutá v abstraktnej triede Application.

Výpis 36 Trieda button_apliukace_fxml

```
package button_aplikace_fxml;

import javafx.application.Application;
import javafx.fxml.FXMLLoader;
import javafx.scene.Parent;
import javafx.scene.Scene;
import javafx.stage.Stage;

/**
 * @author Veronika Vallušová
 */
public class Button_aplikace_FXML extends Application
{

    @Override
    public void start(Stage stage) throws Exception
    {
        Parent root = FXMLLoader.load(getClass()
                            .getResource("FXMLDocument.fxml"));

        Scene scene = new Scene(root);

        stage.setScene(scene);
        stage.show();
    }

    /**
     * @param args the command line arguments
     */
    public static void main(String[] args)
    {
        launch(args);
    }
}
```

Jazyk FXML je založený na XML a slúži k navrhovaniu GUI. Logická časť aplikácie je oddelená od grafickej časti, a tým sa samotný kód stáva prehľadnejším. Druhú triedu predstavuje práve grafická časť aplikácie, ktorá má koncovku `.fxml`. Dvojklikom na túto triedu v stromovej štruktúre sa otvorí okno v nástroji JavaFX Scene Builder. V tomto prípade je hlavným panelom AnchorPane, do ktorého sú vložené dve komponenty tlačidlo (Button) s obrázkom a Label. Zložka kód (Code) v pravej časti okna je veľmi dôležitá. Nato, aby sa na komponent dalo odkazovať v Java kóde, je nutné mu prideliť `id`. Nakoľko s našom prípade nepotrebujeme na tlačidlo odkazovať, `fx:id` mu prideľovať nebudeme. Po kliknutí na tlačidlo, chceme aby sa vykonala nejaká činnosť preto v políčku „`On Action`" mu pridelíme handler udalostí, ktorý následne použijeme v poslednej, teda tretej triede, ktorú nám program NetBeans automaticky vygeneroval.



Obrázok 16 Trieda FXMLDocument.fxml

Pokiaľ namiesto dvojkliku klikneme na triedu pravým tlačidlom myši a vyberieme položku `Edit`, otvorí sa vygenerovaný XML popis komponent, ktoré boli použité. Do tohto kódu užívateľ nemusí zasahovať. Za XML hlavičkou sa nachádzajú importy, načítané balíčky, z ktorých sa vytvorili komponenty vložené do AnchorPane, hlavného okna aplikácie. Tomuto panelu bol pridelený kontroler, popísaný v inej kapitole tejto práce. Ako bolo zmienené v kapitole 2, panel obsahuje elementy children, deti, v našom prípade komponenty Button s obrázkom a Label. Komponentom sú pridelené vlastnosti ako je napríklad veľkosť, poloha, farba písma a podobne. Každá jedna zmena vykonaná v Scene Builder sa automaticky doplní do nasledujúceho kódu. To platí aj naopak. Pokiaľ užívateľ prepíše xml kód v programe NetBeans, program Scene Builder to prekreslí podľa parametrov v programe NetBeans.

Výpis 37Popis komponent pomocou FXML kódu

```
<?xml version="1.0" encoding="UTF-8"?>

<?import java.lang.*?>
<?import java.util.*?>
<?import javafx.scene.*?>
<?import javafx.scene.control.*?>
<?import javafx.scene.image.*?>
<?import javafx.scene.layout.*?>

<AnchorPane id="anchor" prefHeight="200.0" prefWidth="320.0"
xmlns:fx="http://javafx.com/fxml/1"
xmlns="http://javafx.com/javafx/2.2"
fx:controller="button_aplikace_fxml.FXMLDocumentController">
  <children>
    <Button contentDisplay="CENTER" layoutX="93.0" layoutY="23.0"
minHeight="102.0" minWidth="135.0" mnemonicParsing="false"
onAction="#handleButtonAction" prefHeight="102.0"
prefWidth="135.0" style="" text="" textFill="BLACK">
      <graphic>
        <ImageView fitHeight="114.0" fitWidth="152.0"
mouseTransparent="true" pickOnBounds="true" preserveRatio="true">
          <image>
            <Image url="@../zdroje/fotka.png" />
          </image>
        </ImageView>
      </graphic>
    </Button>
    <Label fx:id="label" alignment="CENTER"
contentDisplay="CENTER" layoutX="60.0" layoutY="134.0"
prefHeight="21.0" prefWidth="201.0" text="Label"
textAlignment="CENTER" textOverrun="WORD_ELLIPSIS" />
  </children>
</AnchorPane>
```

Kontrolér predstavuje kód v jazyku JavaFX, ktorý sprostredkuje prepojenie medzi grafickou a logickou vrstvou a obsahuje definíciu všetkých komponent, ku ktorým je nutná nejaká ďalšia aktivita spolu s ich handlermi udalosťami. Implementuje rozhranie `Initalizable` s jednou metódou `initialize (URL url, ResourceBundle rb)`, ktorá pridá základné údaje a priradí ich do príslušnej komponenty [KOPECKY, 2015]. U každého komponentu z grafickej vrstvy, na ktorú chceme odkazovať, je potrebné použiť notáciu `@FXML`. Názov premennej musí byť rovnaký ako `fx:id` v FXML kóde, inak nedôjde k prepojeniu.

V nasledujúcom výpise je nutné sa odkazovať na Label a handler udalostí. Na samotné tlačidlo sa odkazovať netreba, odkazuje sa iba na metódu, ktorá je mu pridelená nakoľko so

samotným tlačidlom sa nič nevykonáva. Metóda po kliknutí na tlačidlo priradí text do Label. JavaFX vykoná všetky operácie a spustí aplikáciu.

Výpis 38 Trieda button_aplikace_fxml

```
package button_aplikace_fxml;

import java.net.URL;
import java.util.ResourceBundle;
import javafx.event.ActionEvent;
import javafx.fxml.FXML;
import javafx.fxml.Initializable;
import javafx.scene.control.Label;

/**
 *
 * @author Veronika Vallušová
 */
public class FXMLDocumentController implements Initializable
{

    @FXML
    private Label label;

    @FXML
    private void handleButtonAction(ActionEvent event)
    {
        label.setText("Klikol si na tlačitlo s obrázkom jahôd");
    }

    @Override
    public void initialize(URL url, ResourceBundle rb)
    {
        // TODO
    }

}
```

9.3 Využitie CSS štýlu

Na nasledujúcich dvoch výpisoch môže čitateľ vidieť názornú ukážku ako vytvoriť CSS štýl dvomi spôsobmi. Bolo zmenené pozadie, štýl a farba písma. Názov súboru s kódom je pomenovaný style.css. Tento súbor je nutné vložiť do aplikácie v Scene Builder. V pravej časti programu Properties->Stylesheet priradíme súbor pomocou tlačidla „+“. Vyhľadáme .css súbor, uložený v počítači a uložíme. Týmto spôsobom je možné jednoducho priradiť farbu a iné štýly do aplikácie.

Výpis 39 Stylesheet1

style.css - Poznámkový blok

Súbor Úpravy Formát Zobraziť Pomocník

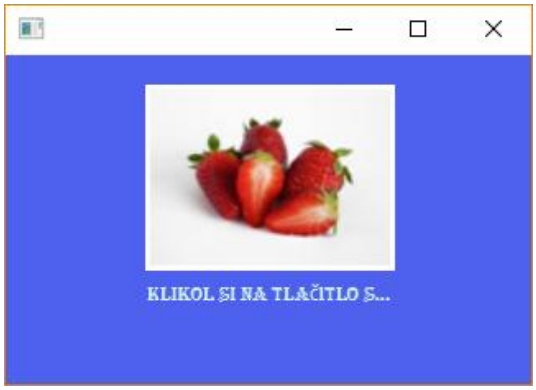
```
/* CSS Document */
.label{
        -fx-text-fill: rgb(255,0,0);
  -fx-font-family: "Courier New";
}
.root{
  -fx-background-color: rgb(253, 211, 80);
}
```

Výpis 40 Stylesheet2

style2 - Poznámkový blok

Súbor Úpravy Formát Zobraziť Pomocník

```
/* CSS Document */
.root{
  -fx-background-color: rgb(77, 97, 238);
}
.label{
  -fx-text-fill: rgb(215, 245, 247);
  -fx-font-family: "Algerian";
}
```



10. Záver

JavaFX ponúka veľké množstvo komponent, ktoré môže programátor použiť. Bohužiaľ, rozsah bakalárskej práce nedovoľuje popísať všetky komponenty, ktoré jazyk JavaFX ponúka. Z toho dôvodu sú v tejto práci obsiahnuté všetky tie komponenty, ktoré by mohol študent kurzu 4IT115 Softvérové inžinierstvo, pri programovaní svojej finálnej aplikácie využiť v jazyku JavaFX alebo pomocou JavaFX Scene Builder.

Cieľom bakalárskej práce bolo vytvoriť príručku pre ďalších študentov VŠE študujúcich tento kurz. Po prečítaní tejto práce by sa čitateľ mal vedieť rozhodnúť, ktorý komponent, panel by sa mal použiť. Vďaka prepojeniu teoretickej časti s praktickou, dokáže jednoduchšie pochopiť ako komponenty či panely naprogramovať a ako budú nakoniec vo finálnej aplikácii po spustení vyzerať. Obidve naprogramované aplikácie majú rovnaký výstup, predstavujú rovnakú aplikáciu, avšak sú tam malé odlišnosti v zdrojovom kóde. Práve, kvôli týmto odlišnostiam, je tá istá aplikácie napísaná dvomi spôsobmi, pomocou jazyku JavaFX a s pomocou Scene Builder.

Pevne verím, že všetky uvedené informácie v tejto bakalárskej práci pomôžu všetkým študentom pri programovaní GUI pre adventúru Červenej čiapočky a zároveň aj ostatným čitateľom tejto práce.

11. Terminologicky slovník

Termín	Skratka	Význam
Application Programming Interface	API	Rozhranie pre programovanie aplikácie.
Audio Interchange File Format	AIFF	Audio format pre uloženie zvukových dát na osobnom počítači.
Abstract Window Toolkit	AWT	Komponent, ktorý sa používa pre podporu grafiky.
Cascade Style Sheet	CSS	Kaskádové štýly pre úpravu vzhľadu aplikácie.
Direct3D	D3D	Grafické rozhranie pre programovanie aplikácií.
Document Object Model	DOM	Objektovo orientovaná reprezentácia dokumentu XML alebo HTML.
Flash Video	FLV	Formát súboru obsahujúci video pre prehrávanie videí na internete.
Integrated development environment	IDE	Softvérová aplikácia poskytujúca rozsiahle vybavenie vývoja softvéru pre programátorov.
Java Standard Edition	Java SE	Slúži pre použite Javy na osobných počítačoch.
Java Development Kit	JDK	Podpora pre vývoj Java aplikácií.
Java Routine Environments	JRE	Virtuálny stroj pre spustenie Java aplikácií.
Open Graphic Library	OpenGL	Štandard pre viacplatformové rozhranie.
Waveform Audio File Format	WAV	Bezdrôtový zvukový formát.

12. Zoznam použitej literatúry

[BENÁČANOVÁ, 2015] BENÁČANOVÁ, H. *Sylabus predmetu 4IT101 - Programování v Javě (FIS - ZS 2015/2016). In: Integrovaný študijný informačný systém* [online]. Praha: Vysoká škola ekonomická v Praze, 2015 [cit. 2016-02-07]. Dostupné z: <https://isis.vse.cz/auth/katalog/syllabus.pl?predmet=110971;aktualni=1>

[ITNETWORK, 2016] ITNETWORK. *Úvod do JavaFX* [online]. 2016 [cit. 2016-03-03]. Dostupné z: <http://www.itnetwork.cz/java/javafx/java-tutorial-uvod-do-javafx>

[JAKOB, 2015] JAKOB, M. *JavaFX 8 Tutorial - Part 1: Scene Builder* [online]. 2015 [cit. 2016-03-11]. Dostupné z: http://code.makery.ch/library/javafx-8-tutorial/part1/

[KOPECKY, 2015] KOPECKY, J. *JavaFX 8 a její použití při výuce,* 2015 [cit. 2016-03-04].

[ORACLE, 2013a] ORACLE. *Working with the JavaFX Scene Graph* [online]. 2013 [cit. 2016-03-02]. Dostupné z: <http://docs.oracle.com/javafx/2/scenegraph/jfxpub-scenegraph.htm>

[ORACLE, 2013b] ORACLE. *JavaFX Architecture* [online]. 2013 [cit. 2016-03-03]. Dostupné z: http://docs.oracle.com/javafx/2/architecture/jfxpub-architecture.htm

[ORACLE, 2013c] ORACLE. *Using Built-in Layout Panes* [online]. 2013. 2016-03-03]. Dostupné z:

<http://docs.oracle.com/javafx/2/layout/builtin_layouts.htm#CHDGHCDG>

[ORACLE, 2014a] ORACLE. *JavaFX: Working with JavaFX UI Components. 24 Menu* [online]. 2014 [cit. 2016-04-01]. Dostupné z: <https://docs.oracle.com/javase/8/javafx/user-interface-tutorial/menu_controls.htm>

[ORACLE, 2014b] ORACLE. *JavaFX: Working with JavaFX UI Components. 2 Label* [online]. 2014 [cit. 2016-04-01]. Dostupné z: <https://docs.oracle.com/javase/8/javafx/user-interface-tutorial/label.htm>

[ORACLE, 2014c] ORACLE. *JavaFX: Working with JavaFX UI Components. 3 Button* [online]. 2014 [cit. 2016-04-01]. Dostupné z: <https://docs.oracle.com/javase/8/javafx/user-interface-tutorial/button.htm>

[ORACLE, 2014d] ORACLE. *JavaFX: Working with JavaFX UI Components. 6 Checkbox* [online]. 2014 [cit. 2016-04-01]. Dostupné z: <https://docs.oracle.com/javase/8/javafx/user-interface-tutorial/checkbox.htm>

[ORACLE, 2014e] ORACLE. *JavaFX: Working with JavaFX UI Components. 4 Radio Button* [online]. 2014 [cit. 2016-04-01]. Dostupné z: https://docs.oracle.com/javase/8/javafx/user-interface-tutorial/radio-button.htm

[ORACLE, 2014f] ORACLE. *JavaFX: Working with JavaFX UI Components. 7 Choice Box* [online]. 2014 [cit. 2016-04-01]. Dostupné z: <https://docs.oracle.com/javase/8/javafx/user-interface-tutorial/choice-box.htm>

[ORACLE, 2014g] ORACLE. J*avaFX: Working with JavaFX UI Components. 16 Combo Box* [online]. 2014 [cit. 2016-04-01]. Dostupné z: <https://docs.oracle.com/javase/8/javafx/user-interface-tutorial/combo-box.htm>

[ORACLE, 2014h] ORACLE. *JavaFX: Working with JavaFX UI Components. 12 List View* [online]. 2014 [cit. 2016-04-01]. Dostupné z: <https://docs.oracle.com/javase/8/javafx/user-interface-tutorial/list-view.htm>

[ORACLE, 2014i] ORACLE. *JavaFX: Working with JavaFX UI Components. 8 Text*

Field [online]. 2014 [cit. 2016-04-01]. Dostupné z: <https://docs.oracle.com/javase/8/javafx/user-interface-tutorial/text-field.htm>

[ORACLE, 2014j] ORACLE. *JavaFX: Working with JavaFX UI Components. 9 Password Field* [online]. 2014 [cit. 2016-04-01]. Dostupné z: https://docs.oracle.com/javase/8/javafx/user-interface-tutorial/password-field.htm

[ORACLE, 2014k] ORACLE. *JavaFX: Working with JavaFX UI Components. 10 Scroll Bar* [online]. 2014 [cit. 2016-04-01]. Dostupné z: <https://docs.oracle.com/javase/8/javafx/user-interface-tutorial/scrollbar.htm>

[ORACLE, 2014l] ORACLE. *JavaFX: Working with JavaFX UI Components. 11 Scroll Pane* [online]. 2014 [cit. 2016-04-01]. Dostupné z: <https://docs.oracle.com/javase/8/javafx/user-interface-tutorial/scrollpane.htm>

[ORACLE, 2014m] ORACLE. *37 Styling UI Controls with CSS* [online]. 2014 [cit. 2016-03-11]. Dostupné z: <http://docs.oracle.com/javase/8/javafx/user-interface-tutorial/apply-css.htm#CHDGHCDG>

[ORACLE, 2015] ORACLE. *Class TextArea.* [online]. 2015 [cit. 2016-11-26]. Dostupné z: <https://docs.oracle.com/javase/8/javafx/api/javafx/scene/control/TextArea.html>

[PECINOVSKÝ, 2007] PECINOVSKÝ, R. *Návrhové vzory: [33 vzorových postupů pro objektové programování]*. Brno:Computer Press, 2007 ISBN 978-80-251-1582-4.

[SHARAN, 2015] SHARAN, K. *Learn JavaFX 8: Building User Experience and Interfaces with Java 8*. Apress, 2015 ISBN 978-1-4842-1142-7.

[TOPLEY, 2010] TOPLEY, K. *JavaFX Developer's guide.* Prvé vydanie. Ann Arbor: Addison-Wesley, 2010. 1109 s. ISBN 978-0-321-60165-0.

13. Zoznam obrázkov

14. Zoznam výpisov