

UNIVERZITA PARDUBICE

Fakulta elektrotechniky a informatiky

Herní platforma NetBeans

Jaroslav Janíček

Bakalářská práce

2015

Univerzita Pardubice  
Fakulta elektrotechniky a informatiky  
Akademický rok: 2014/2015

## **ZADÁNÍ BAKALÁŘSKÉ PRÁCE**

(PROJEKTU, UMĚLECKÉHO DÍLA, UMĚLECKÉHO VÝKONU)

Jméno a příjmení: **Jaroslav Janíček**  
Osobní číslo: **I12141**  
Studijní program: **B2646 Informační technologie**  
Studijní obor: **Informační technologie**  
Název tématu: **Herní platforma NetBeans**  
Zadávající katedra: **Katedra informačních technologií**

### **Z á s a d y   p r o   v y p r a c o v á n í :**

Cílem práce je vytvořit vlastní vývojovou platformu Netbeans pro tvorbu 2D her.

V teoretické části bude popsána tvorba vlastních modulů a jejich propojení do NetBeans platformy. Dále budou popsány potřebné tovární API moduly a jejich implementace do NetBeans. Stručně budou popsány technologie XML, Java8 a JavaFX.

V praktické části práce bude vytvořen vlastní modul do vlastní NetBeans platformy a realizováno GUI pro tvorbu 2D her.

Rozsah grafických prací:

Rozsah pracovní zprávy:

Forma zpracování bakalářské práce: **tištěná/elektronická**

Seznam odborné literatury:

**DEA, Carl, Mark HECKLER, Gerrit GRUNWLD, Jose PEREDA a Sean M PHILLIPS. Pro JavaFX 8: a definitive guide to building rich java clients. 2. vyd. Berkeley: Apress, 2014. ISBN 978-143-0265-740.**

**SHARAN, Kishori. Beginning Java 8 language features: interfaces, inner classes, threads, i/o and collections. Berkeley: Apress, 2014. ISBN 978-143-0266-587.**

**WEXBRIDGE, Jason a Walter NYLAND. NetBeans Platform for Beginners: Modular Application Development for the Java Desktop [online]. 1. vyd. Leanpub, 2014 [cit. 2014-10-20]. Dostupné z: <https://leanpub.com/nbp4beginners>**

Vedoucí bakalářské práce:

**Ing. Zdeněk Šilar, Ph.D.**

Katedra informačních technologií

Datum zadání bakalářské práce:

**20. prosince 2014**

Termín odevzdání bakalářské práce:

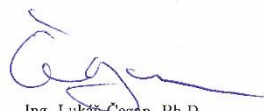
**11. května 2015**



prof. Ing. Simeon Karamazov, Dr.  
děkan



L.S.



Ing. Lukáš Čegan, Ph.D.  
vedoucí katedry

V Pardubicích dne 31. března 2015

### **Prohlášení autora**

Prohlašuji, že jsem tuto práci vypracoval samostatně. Veškeré literární prameny a informace, které jsem v práci využil, jsou uvedeny v seznamu použité literatury.

Byl jsem seznámen s tím, že se na moji práci vztahují práva a povinnosti vyplývající ze zákona č. 121/2000 Sb., autorský zákon, zejména se skutečností, že Univerzita Pardubice má právo na uzavření licenční smlouvy o užití této práce jako školního díla podle § 60 odst. 1 autorského zákona, a s tím, že pokud dojde k užití této práce mnou nebo bude poskytnuta licence o užití jinému subjektu, je Univerzita Pardubice oprávněna ode mne požadovat přiměřený příspěvek na úhradu nákladů, které na vytvoření díla vynaložila, a to podle okolností až do jejich skutečné výše.

Souhlasím s prezenčním zpřístupněním své práce v Univerzitní knihovně.

V Pardubicích dne:

Jaroslav Janíček

## **Poděkování**

Děkuji svému vedoucímu práce, že mi umožnil vypracovat tuto semestrální práci, jejíž obsah mě velice zaujala. Dále také za zkušenosti, které jsem obdržel v jeho předmětu Programovací techniky v jazyce Java.

## **Anotace**

V práci je věnována pozornost vývoji vlastních platforem NetBeans za pomoci platforem Java a JavaFX. Výsledkem je desktopová aplikace, kterou lze nainstalovat za pomoci instalačního balíku. Aplikace umí vytvářet 2D mapy obsahující dvě vrstvy a následné uložení do XML. Součástí aplikace je i vytvořená vzorová hra využívající prostředky aplikace jakožto vývojové prostředí.

## **Klíčová slova**

NetBeans, Java, JavaFX, NetBeans Platform Application, XML

## **Title**

Game NetBeans platform.

## **Annotation**

The attention is paid to develop custom NetBeans platform with the help of Java and JavaFX. The result is a desktop application that can be installed using the installer package. Applications can create 2D map containing two layers and subsequent storage in XML. Part of the application is a game created a sample application using the funds as a development environment.

## **Keywords**

NetBeans, Java, JavaFX, NetBeans Platform Application, XML

## Obsah

<b>Seznam zkratk.....</b>	<b>8</b>
<b>Seznam obrázků.....</b>	<b>9</b>
<b>Seznam tabulek .....</b>	<b>9</b>
<b>Úvod .....</b>	<b>10</b>
<b>1 Pojem hra .....</b>	<b>11</b>
1.1 Elektronické hry .....	11
1.2 Rozdělení her .....	11
1.2.1 RPG hry .....	11
1.2.2 Deskové a karetní hry .....	11
1.2.3 Plošinové hry neboli Platform game .....	12
1.2.4 Tower defense .....	12
<b>2 Vývojové prostředí NetBeans .....</b>	<b>13</b>
2.1 Historie .....	13
2.2 Aktuální verze .....	13
2.3 Licence .....	13
<b>3 Java .....</b>	<b>14</b>
3.1 Java 8 .....	14
3.2 JavaFX .....	15
3.2.1 JavaFX Script, JavaFX2 a JavaFX8 .....	15
3.3 Swing .....	16
3.3.1 JComboBox .....	16
3.3.2 JColorChooser .....	17
3.3.3 JFileChooser .....	17
<b>4 Extensible Markup Language (XML) .....</b>	<b>18</b>
4.1 Syntaxe XML .....	18
4.2 Zpracování XML v Javě .....	18
<b>5 Platforma NetBeans.....</b>	<b>19</b>
5.1 Vytvoření vlastní NetBeans Platform Application .....	19
5.2 Základní tovární moduly a uzly .....	23
5.3 Vytvoření vlastních modulů .....	24
5.3.1 Window TopComonent .....	24

5.3.2	Vytvoření Action tlačítek .....	25
5.4	Vytvoření instalačního souboru a značkování (Branding) .....	26
<b>6</b>	<b>Realizace vývojového prostředí Game Platform .....</b>	<b>28</b>
6.1	Struktura prostředí .....	28
6.2	Action AddPlatformLevelListener & OpenPlatformMapListener .....	28
6.3	Level Editor Window .....	29
6.3.1	Vytvoření nové mapy .....	29
6.3.2	Funkcionalita .....	30
6.4	ToolsEditor Window .....	31
6.5	Vytvoření vlastní hry s názvem Platform .....	32
6.5.1	Specifikace úrovně pro hru Platform .....	33
6.5.2	Typy nepřátel .....	33
6.5.3	Kolize .....	34
6.5.4	Vrstvy .....	35
6.5.5	Přidání nové úrovně .....	35
6.5.6	Ovládání hry .....	35
6.6	Distribuce .....	35
	<b>Závěr .....</b>	<b>36</b>
	<b>Literatura .....</b>	<b>37</b>
	<b>Příloha A – Ukázka zpracování XML .....</b>	<b>39</b>
	<b>Příloha B – Zdrojový kód pro vytvoření nové mapy .....</b>	<b>41</b>
	<b>Příloha C – Základní obrázky pro tvorbu mapy .....</b>	<b>42</b>
	<b>Příloha D – Kolize hráče s plošinou .....</b>	<b>43</b>

## Seznam zkratek

GNU GPL	General Public License
SDK	Software Development Kit.
JDK	Java Development Kit
JVM	Java Virtual achine
RIA	Rich Internet Applications
XML	Extensible Markup Language
HTML	HyperText Markup Language
DOM	Document Object Model
SAX	Simple API for XML
GUI	Graphic User Interface
IDE	Integrated Development Enviroment
RPG	Role-Playing Game
MMORPG	Massively Multiplayer Online Role-Playing Game

## Seznam obrázků

Obrázek 1 NetBeans logo .....	13
Obrázek 2 Zadní štítek Blue-Ray přehrávače od firmy Samsung .....	14
Obrázek 3 JComboBox .....	16
Obrázek 4 JColorChooser .....	17
Obrázek 5 JFileChooser .....	17
Obrázek 6 Gephi.org .....	19
Obrázek 7 Přehled aktivních modulů v NetBeans .....	20
Obrázek 8 Nový projekt NetBeans Platform Application .....	21
Obrázek 9 Zadání názvu a lokace nového projektu NetBeans Platform Application .....	21
Obrázek 10 Výchozí nastavení Netbean Platform Application .....	22
Obrázek 11 Přehled továrních uzlů .....	23
Obrázek 12 Přidání modulu do existujícího projektu .....	24
Obrázek 13 Propojení modulů .....	24
Obrázek 14 Pozice oken v Platformě NetBeans .....	25
Obrázek 15 Action v Global Toolbar .....	26
Obrázek 16 Nastavení vytváření instalačních souborů .....	27
Obrázek 17 Okno pro nastavení značkování (Branding) .....	27
Obrázek 18 Tlačítka v global Toolbaru pro vytvoření a otevření mapy .....	29
Obrázek 19 LevelEditor Window .....	30
Obrázek 20 Ukázka vrstev .....	31
Obrázek 21 ToolsEditor Window .....	32
Obrázek 22 Hra Platforms .....	33
Obrázek 23 Postup v úrovni default a přehled životů .....	34
Obrázek 24 Přidání nové úrovně do hry .....	35

## Seznam tabulek

Tabulka 1 Základní soubory při vytvoření nové platformy NetBeans .....	22
--	----

## Úvod

Cílem této práce je vytvoření vlastního vývojového prostředí založený na platformě NetBeans pro tvorbu jednoduchých her. Prostředí bude umožňovat tvorbu vlastních dvouvrstvých map ve 2D, s možností uložení map do XML a její následné načtení a upravování. Díky uložení map do souborů XML je možné její importování do jakékoliv hry, kterou si konečný uživatel navrhne a je zcela na něm, jak mapu využije.

Prostředí bude obsahovat možnost vytvoření vlastního projektu JavaFX a projektu Java převzaté z vývojového prostředí NetBeans pro tvorbu vlastní hry, která využije mapu vytvořenou v témže prostředí.

Tteoretické část je zaměřena na použitou technologii, která byla zvolena pro realizaci vlastního vývojového prostředí.

Praktická část se zabývá implementací technologií pro vytvoření vlastního prostředí za pomoci platformy NetBeans pro tvorbu her. Součástí praktické části je ukázková hra využívající platformy JavaFX a vytvořeného vývojového prostředí.

# 1 Pojem hra

Hra je činnost jednoho či více lidí, která nemusí mít konkrétní smysl, ale přitom má za cíl radost či relaxaci. Hry se hrají především pro zábavu, ale mohou také sloužit například ke vzdělávání. Rolí hry ve společnosti se zabývá věda známá jako ludologie [1].

## 1.1 Elektronické hry

Elektronické hry neboli počítačové, konzolové, mobilní či webové hry jsou v současné době velmi populární. Největší oblibě se těší hry pro více hráčů či hry umožňující co-operace<sup>1</sup>, nejlépe založené na herním modelu free-to play<sup>2</sup>. Jednou z nejpokulárnějších her free-to play je například Dota 2 či Counter-Strike Global Offensive.

Nově se opět těší oblibě plošinové hry neboli 2D hry, dnes už s retro grafikou neboli pixelovou grafikou. Opětovnou oblibu plošinových her má zřejmě na svědomí rozvoj chytrých telefonů, které umožnily reinkarnaci starých her.

## 1.2 Rozdělení her

Základní rozdělení her je na hry textové nebo grafické (2D a 3D hry). Po tomto základním rozdělení se hry mohou dále dělit na hry on-line, které lze hrát pouze po připojení k internetu a k hernímu serveru nebo na hry off-line (bez připojení k internetu).

Způsob rozdělení her je opravdu mnoho a dost zřídka se k dnešním hrám dá přiřadit jasný styl a kategorie. Poněvadž většina moderních velkých her obsahuje tolik žánrů, miniher a možností, jak danou hru hrát, například sám v co-op módu nebo online s neznámými hráči.

### 1.2.1 RPG hry

RPG<sup>3</sup> hry jsou asi nejvíce populární hry ze všech. Jedná se o hry, kdy hráč postupně vylepšuje svoji postavu (hrdinu). Plní různé úkoly a získává za jejich splnění zkušenosti a zlato či jinou odměnu. Zkušenosti umožňují v růstu postavy pro získání nových schopností či zvýšení atributů hrdiny, například síly, výdrže nebo zdraví.

Jednou z nejznámějších a nejpokulárnějších RPG 2D her je Diablo a Diablo 2. Nebo ze světa 3D her opět hra Diablo 3 nebo Zaklínač, nemluvě o králi mezi MMORPG<sup>4</sup> hrami Word of Warcraft.

### 1.2.2 Deskové a karetní hry

Deskové hry patří také k velmi oblíbeným hrám. Především karetní hry, které se dají dnes hrát online za skutečné peníze a tím dostaly nový rozměr.

---

<sup>1</sup> Co-operace neboli také co-op vychází z anglického slova cooperation neboli spolupráce.

<sup>2</sup> Free-To Play označení pro hry zdarma, nejčastěji s přídavným obsahem za peníze.

<sup>3</sup> RPG Role-playing game neboli hry na hrdiny.

<sup>4</sup> MMORPG Massively Multiplayer Online Role-Playing Game neboli masivní multiplayerové RPG.

### **1.2.3 Plošinové hry neboli Platform game**

Plošinové hry neboli lidově „plošinovky“ jsou také označovány jako „skákačky“. Jsou nejpopulárnější 2D hry. Jedná se o hry založené na principu překonání překážek za pomoci přeskakování z jedné plošiny na druhou. Cestu většinou znepříjemňují nepřátelé, kteří se pohybují dle předem daných pravidel a nemají vlastní umělou inteligenci.

Světově známá a dnes už legendární plošinová hra je Super Mario nebo méně známý Prehistoric. Samozřejmě existují i 3D plošinové hry, kde opět kraluje Mario nebo méně známý Trine 2.

### **1.2.4 Tower defense**

Tower defense hry jsou ideální pro dvourozměrný prostor. Princip her Tower defense je velmi prostý. Na předem stanovené mapě je určena základna, kterou se budou snažit nepřátelé zničit. Nepřátelé se pohybují po předem definované cestě nebo cestu tvoří hráč pomocí rozmístění svých obranných věží.

Tower defense hry jsou velmi vhodné pro mobilní zařízení díky svému jednoduchému ovládání. Svým netradičním pojetím dobyla hra Plants vs. Zombie celý svět jak na desktopových tak na mobilních zařízeních.

## 2 Vývojové prostředí NetBeans

Jedná se o moderní vývojové prostředí s komplexními nástroji určené k vývoji aplikací. Toto prostředí usnadňuje práci programátorům zvýrazňováním syntaxe, detekováním chyb či možností debuggování neboli krokování programu. Zároveň umožňuje jednodušší tvorbu dokumentace ke zdrojovému kódu nebo možnost vytvoření vlastních testů pro vyvíjenou aplikaci [2].

### 2.1 Historie

Vývojové prostředí NetBeans vznikl jako studentský projekt v České Republice v roce 1996, původně s názvem Xelfi. Cílem práce bylo vytvořit Delphi<sup>5</sup> jako Java IDE<sup>6</sup> (Integrated Development Environment). Xelfi byl prvním IDE napsán v programovacím jazyce Java.

Po absolvování studentů se ze studentského projektu stal projekt komerční. Vývojem událostí byl Xelfi přejmenován na NetBeans. Roku 1999 vyšel NetBeans DeveloperX2, jenž podporoval grafickou knihovnu Swing. V témže roce byla změněna architektura NetBeans na modulární architekturu a stal se více modulární platformou.

Po odkoupení NetBeans společností Sun Microsystems, byl ukončen vývoj NetBeans jakož to komerční projekt. Sun Microsystems zveřejnila zdrojové kódy a jakožto open source je vyvíjen dodnes [3].



Obrázek 1 NetBeans logo

### 2.2 Aktuální verze

Aktuální stabilní verzí je NetBeans IDE 8.0.2 s plnou podporou JDK<sup>7</sup>8. V této verzi je v základním vybavení možné vyvíjet aplikace na platformě Java, JavaFX, PHP, HTML5 & JavaScript, C++ a NetBeans Platform.[2]

### 2.3 Licence

Šíření NetBeans je pod licencí GNU GPL, jenž je licencí pro svobodný software. Licence GNU GPL vyžaduje, aby odvozená vytvořená díla pod touto licencí, byla šířena pod stejnou licencí.[4]

---

<sup>5</sup> Delphi je integrované grafické vývojové prostředí.

<sup>6</sup> IDE je vývojové prostředí integrující všechny nástroje nutné k vývoji pro platformy, pro které je určeno.

<sup>7</sup> JDK nebo také Java SDK je sada nástrojů k vývoji softwaru na platformě Java.

### 3 Java

Java je objektově orientovaný jazyk, který vyvinula firma Sun Microsystems v roce 1995. Je jedním z nejoblíbenějších programovacích jazyků na světě. V oblibě je především pro svoji přenositelnost mezi platformami. Je velmi jednoduchá a umožňuje automatické uvolnění operační paměti, obsazenou již nepotřebnými objekty.

Především jde použít jak pro vývoj desktopových aplikací, mobilních aplikací nebo čipových karet. Například Blue-Ray přehrávače od firmy Samsung fungují na platformě Java, jak lze vyčíst ze štítku ze zadní strany přehrávače na Obrázku 2.



Obrázek 2 Zadní štítek Blue-Ray přehrávače od firmy Samsung

Vývoj Javy probíhá jako Open Source neboli knihovny Javy jsou volně šiřitelné, s výjimkou JVM<sup>8</sup>. Z toho důvodu operační systém Android pro mobilní zařízení, využívající platformu Java, má vlastní JVM. I kdyby byl původní JVM volně šiřitelný, nebyl by pro mobilní zařízení vhodný.

Pro vývoj Java aplikací je zapotřebí JDK, který obsahuje sadu základních nástrojů pro vývoj aplikací na platformě Java.[5]

#### 3.1 Java 8

Java 8 je označení pro poslední nejnovější verzi Javy fungující pod JDK 1.8. Největší novinkou v této verzi jsou lambda výrazy. Lambda výrazy v Javě umožňují jiný způsob zápisu anonymních vnitřních tříd.

```
//pomocí vnitřních anonymních tříd
Button btn = new Button();
    btn.setText("Say 'Hello World'");
    btn.setOnAction(new EventHandler<ActionEvent>() {
        @Override
        public void handle(ActionEvent event) {
            System.out.println("Hello World!");
        }
    });
```

---

<sup>8</sup> JVM slouží ke zpracování Java bytecode na různých platformách.

```
//pomocí lambda výrazu
Button btn = new Button();
    btn.setText("Say 'Hello World'");
    btn.setOnAction(event -> {
        System.out.println("Hello World!");
    });
```

Jak jde na první pohled vidět, zápis pomocí lambda výrazů je daleko snazší a kratší. Díky úspornějšímu zápisu by mělo být provádění lambda výrazů rychlejší a také by mohly přilákat vývojáře z platformy C#, kteří jsou na zápis lambda výrazů již zvyklí [15].

## 3.2 JavaFX

JavaFX je založena na platformě Java od společnosti Sun Microsystems. Vznikla jako reakce na platformu Adobe Flash a Microsoft Silverlight. JavaFX je zaměřena především na tvorbu RIA<sup>9</sup> aplikací neboli bohaté webové aplikace. Snahou je vytvořit interaktivní aplikace, které s uživatelem nějakým způsobem komunikují, ať už animacemi či jinými efekty.

Jelikož vývoj JavaFX aplikací je pro tvorbu RIA aplikace, zaměřené na web, je možné spustit aplikaci formou plugin<sup>10</sup> přímo ve webovém prohlížeči. Jednou z velkých výhod je možnost aplikaci stáhnout do svého počítače a poté aplikaci spustit jako klasickou desktopovou aplikaci.

V březnu roku 2014 JavaFX oficiálně nahradila knihovnu Swing pro tvorbu GUI v Javě. NetBeans v aktuální verzi funguje celý na Swingu, proto při vytváření vlastní platformy NetBeans je nutno použít Swing.

Pro inovátory zde ale existuje způsob, jak propojit JavaFX s prvky ze Swingu. Tato problematika bude popsána při vytváření vlastního prostředí **Game Platform** využívající platformu NetBeans.

Jednou z mnoha novinek v JavěFX je obsluha dotykových událostí a gest. Otevírá tím možnost vývoje aplikací pro dotykové displeje. Aplikace JavaFX jsou spustitelné na počítači i na mobilních zařízeních s operačním systémem **Android** a **Windows Mobile**.

### 3.2.1 JavaFX Script, JavaFX2 a JavaFX8

JavaFX Script, JavaFX2 a JavaFX8 - jaký je mezi těmito názvy rozdíl? Ano, jedná se především o označení verze, ale také o různou implementaci této technologie.

Původní JavaFX Script byl skriptovací jazyk s vlastním souborem, zdrojovým souborem s koncovkou **.fx** a byl kompilován jako skriptovací jazyk. Měl vlastní syntaxi a jednalo se prakticky o úplně nový jazyk.

<sup>9</sup> RIA je označení pro takzvané bohaté aplikace umožňující snadný přístup k multimédiím a tvorbu interakce s uživatelem například za pomoci animace.

<sup>10</sup> Plugin či plug-in je přídavný obsah, který nepracuje samostatně, ale jako doplněk jiné aplikace.

JavaFX2 byl implementován přímo do platformy Javy a tím pádem mnohem přívětivější pro Java vývojáře. JavaFX Script byl definitivně zrušen. Jediný pozůstatek z JavaFX Script je takzvaný builder pro vytvoření primitivních tvarů.

```
Rectangle rect2 = RectangleBuilder.create()
    .arcWidth(30)
    .arcHeight(30)
    .fill(Color.WHITESMOKE)
    .x(10)
    .y(160)
    .strokeWidth(3)
    .stroke(Color.BLACK)
    .build();
rect2.setWidth(280);
rect2.setHeight(130);
```

JavaFX8 oficiálně nahrazuje Swing a sjednocuje se s verzí JDK 1.8 (Java8) a také s NetBeans 8. Přibyla možnost stylování grafiky pomocí CSS. Definitivně byla odstraněna jakákoliv spojitost s JavaFX Script včetně builder .

```
Rectangle rect2 = new Rectangle(60, 60, 200, 200);
rect2.setFill(Color.TRANSPARENT);
rect2.setStroke(Color.RED);
rect2.setStrokeWidth(10);
```

Je možné očekávat velké změny a to jak ze strany NetBeans, tak i u vývoje GUI v Javě. Jedná se ale pouze o domněnky a tak nezbývá než čekat, co nového z vývoje JavaFX (JavaFX9) vzejde.[6]

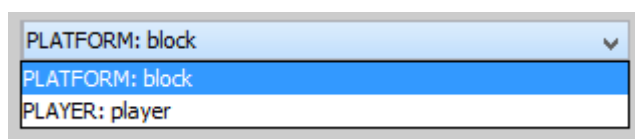
### 3.3 Swing

Swing byl donedávna primární sadou knihoven obsažených v Javě pro grafické rozhraní aplikací neboli GUI rozhraní. Oproti AWT poskytuje emulaci nativního vzhledu komponent pro nejrozšířenější platformy.[7]

#### 3.3.1 JComboBox

JComboBox je swingová komponenta, určená pro výběr z předem nastavených možností. Zobrazená data jsou reprezentována datovým modelem, určující způsob zobrazení. Typickým modelem pro JComboBox je DefaultComboBoxModel.

Pro výběr hodnot z JComboBox jsou tři možné způsoby. První `getSelectedIndex` vrací index neboli pořadí v daném JComboBoxu. Druhý, přímý výběr objektu pomocí `getSelectedItem` nebo pole objektů pomocí `getSelectedObjects`.

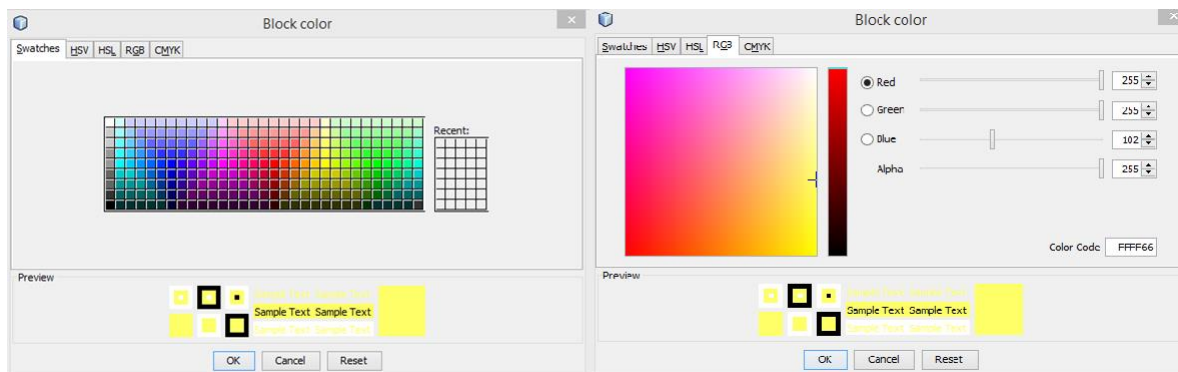


Obrázek 3 JComboBox

### 3.3.2 JColorChooser

JColorChooser je swingová komponenta vyvolávající dialogové okno pro výběr barvy. Výběr barvy je možný buď z předem vytvořené palety nebo za pomoci barevného prostoru RGB, HSV, HSL nebo CMYK. Návratovou hodnotou je objekt Color z knihovny awt.

```
JColorChooser.showDialog(this, "Block color", Color.YELLOW);
```

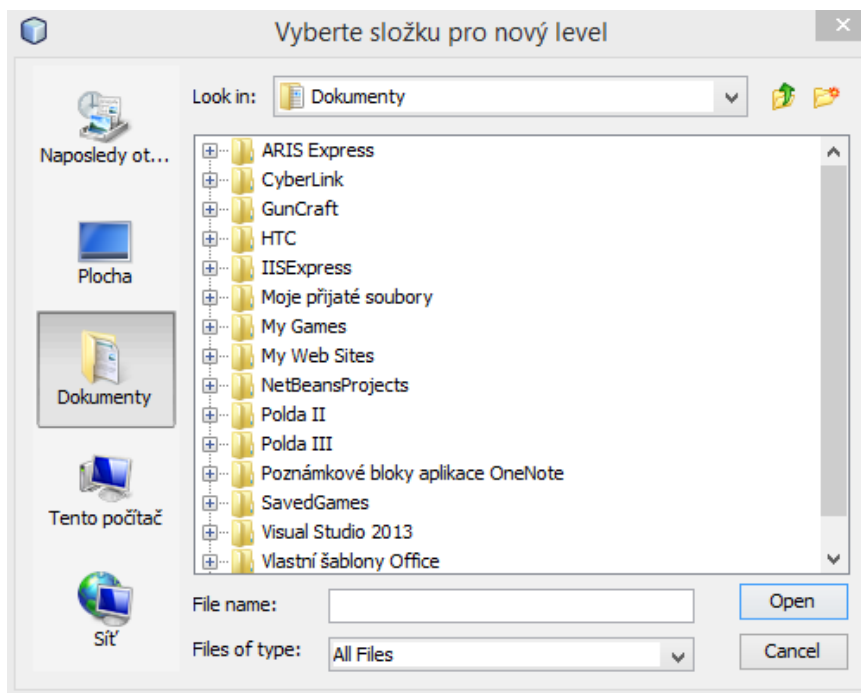


Obrázek 4 JColorChooser

### 3.3.3 JFileChooser

JFileChooser je swingová komponenta vyvolávající dialogové okno pro výběr souborů či složek. Lze použít FileFilter pro filtraci souborů určitého typu, například výběr pouze obrázků nebo textové dokumenty a podobně.

Dále je možné omezit výběr pouze na složku nebo pouze na soubor. Zde je možno, zadat zda se má vybrat pouze jeden objekt nebo zda lze vybrat pouze jednu nebo více položek.



Obrázek 5 JFileChooser

## 4 Extensible Markup Language (XML)

Jedná se o značkovací jazyk, který na rozdíl od **HTML** nemá předem určené značky neboli elementy (tag). Díky tomu umožňuje vytvářet vlastní značkovací jazyk. XML se zabývá výměnou dat především mezi aplikacemi pro publikování nebo komunikaci mezi různými systémy.

### 4.1 Syntaxe XML

Dokument musí obsahovat jeden kořenový element (root). Všechny elementy musí být uzavřené. Atributy elementu musí být uzavřeny v jednoduchých nebo dvojitéch uvozovkách. Elementy mohou být vnořeny do jiného elementu, ale nesmí se překrývat.

```
<?xml version="1.0" encoding="UTF-8"?>
<root>
  <element atribut="vlastnost atributu">
    obsahElementu
  </element>
  <element>
    <vnitrniElement>obsah vnitřního elementu</vnitrniElement>
  </element>
</root>
```

### 4.2 Zpracování XML v Javě

Ke zpracování XML se nejčastěji používají dva způsoby. DOM parser, kdy se celý obsah načte do paměti a vytvoří se stromová struktura. Poté se ze stromu čtou potřebné informace. Nevýhodou zpracování pomocí DOM je v případě nutnosti přečíst pouze jednu informaci. I tak je za potřebí načíst celý dokument do paměti. Avšak v případě velkých dokumentů to není příliš výhodné.

Druhým nejčastějším způsobem je SAX parser. Soubor XML se pouze čte. Celé zpracování dokumentu zajišťuje programátor. Ukázka jak využít SAX parser v Javě je v příloze A [8, 9].

## 5 Platforma NetBeans

NetBeans platforma je framework pro vytvoření desktopových nástrojů v Javě (např. pro tvorbu her). Nabízí spoustu funkcí jako je spojování akcí do položek v menu, klávesové zkratky, správu oken, textové editory či kompilátor Javy. To vše již NetBeans umí a není zapotřebí vytvářet vlastní prostředí, pouze stačí za pomoci již vytvořených modulů s kombinací vlastních modulů vytvořit svoji vlastní platformu.

Díky ušetřenému času je možné se zaměřit pouze na vlastní produkt, například galerie obrázků, hudební přehrávač nebo UML editor. Můžeme využít již hotový textový editor a upravit si ho k obrazu svému [10].

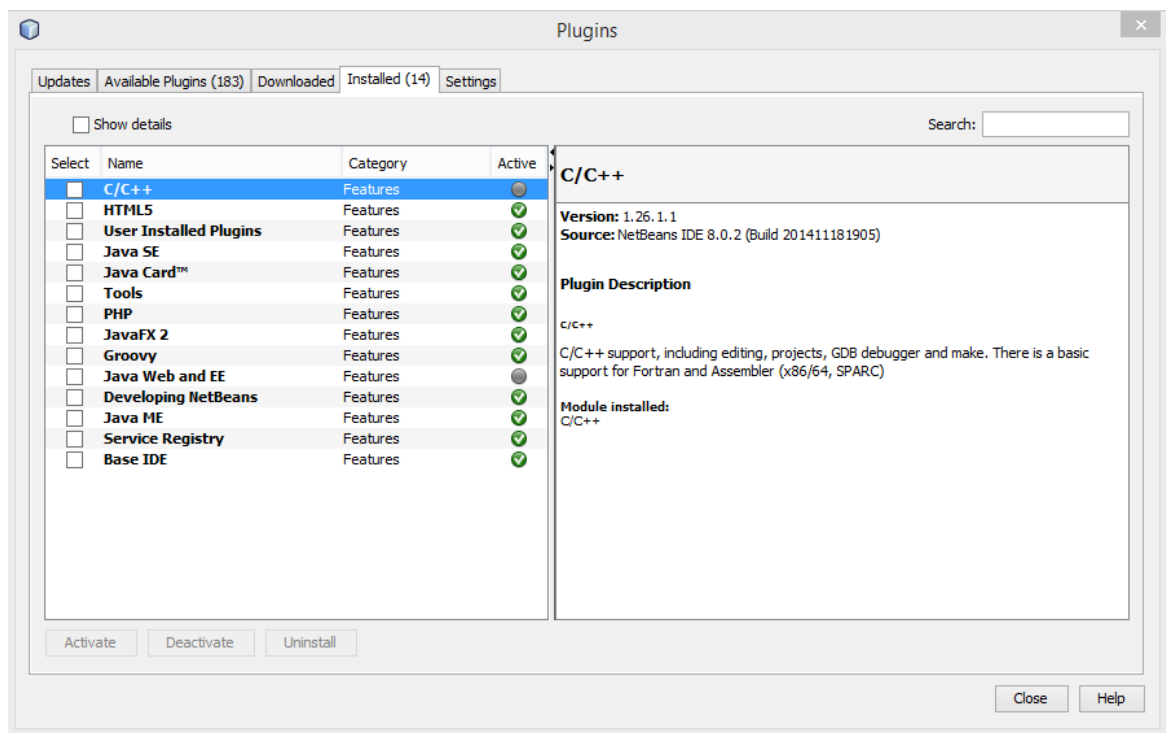


Obrázek 6 Gephi.org

### 5.1 Vytvoření vlastní NetBeans Platform Application

Pro vytvoření platformy NetBeans je potřeba mít nainstalovaný NetBeans IDE s aktivní plugin Developing NetBeans. Pokud plugin není aktivní nebo není nainstalován, je možné plugin aktivovat či doinstalovat následujícím způsobem. Ve správě pluginů (Tools/Plugins), v záložce Installed je zapotřebí aktivovat Developing NetBeans. Na Obrázku 7 je ukázka okna pro správu pluginů. Pokud plugin v dané instalaci zcela chybí, je možné plugin vyhledat a doinstalovat. Stačí zadat název pluginu do vyhledávacího pole (Search).

Pokud se nedaří aktivovat či nainstalovat plugin, je jediná možná oprava, a to reinstalací NetBeans, jenž ve svém základě tento plugin obsahuje.



Obrázek 7 Přehled aktivních modulů v NetBeans

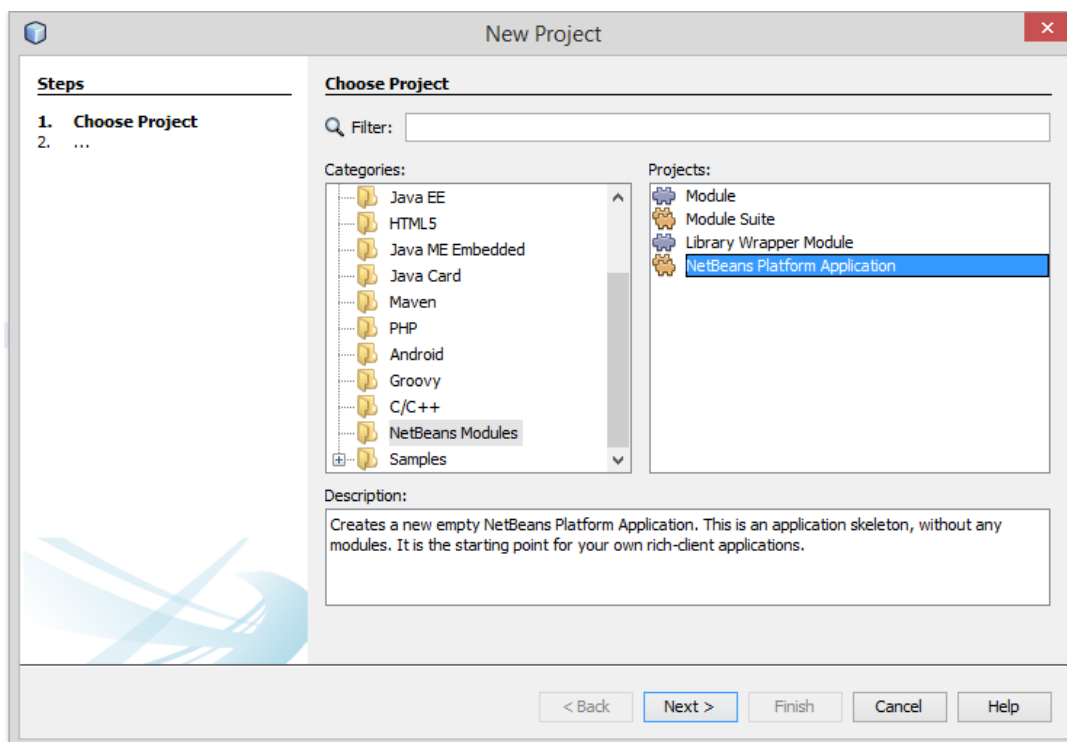
Pokud je plugin Developing NetBeans aktivní, je možné vytvořit vlastní projekt pro vývoj vlastní platformy NetBeans. Pro vytvoření projektu je možné využít klávesovou zkratku **Ctrl + Shift + N** nebo pomocí ikonky **krabičky se zeleným plus**, jenž je vlevo nahoře pod rozvíjejícím se menu vývojového prostředí NetBeans. Pro konvenční přístup vytvoření projektu lze zvolit v menu **File/New Project**.

Po zvolení akce **New Project** se zobrazí okno s nabídkou kategorií zobrazené na Obrázku 8, kde je zapotřebí zvolit kategorii NetBeans Modules. V kategorii jsou dostupné následující projekty: **Module**, **Module Suite**, **Library Wrapper Module** a **NetBeans Platform Application**.

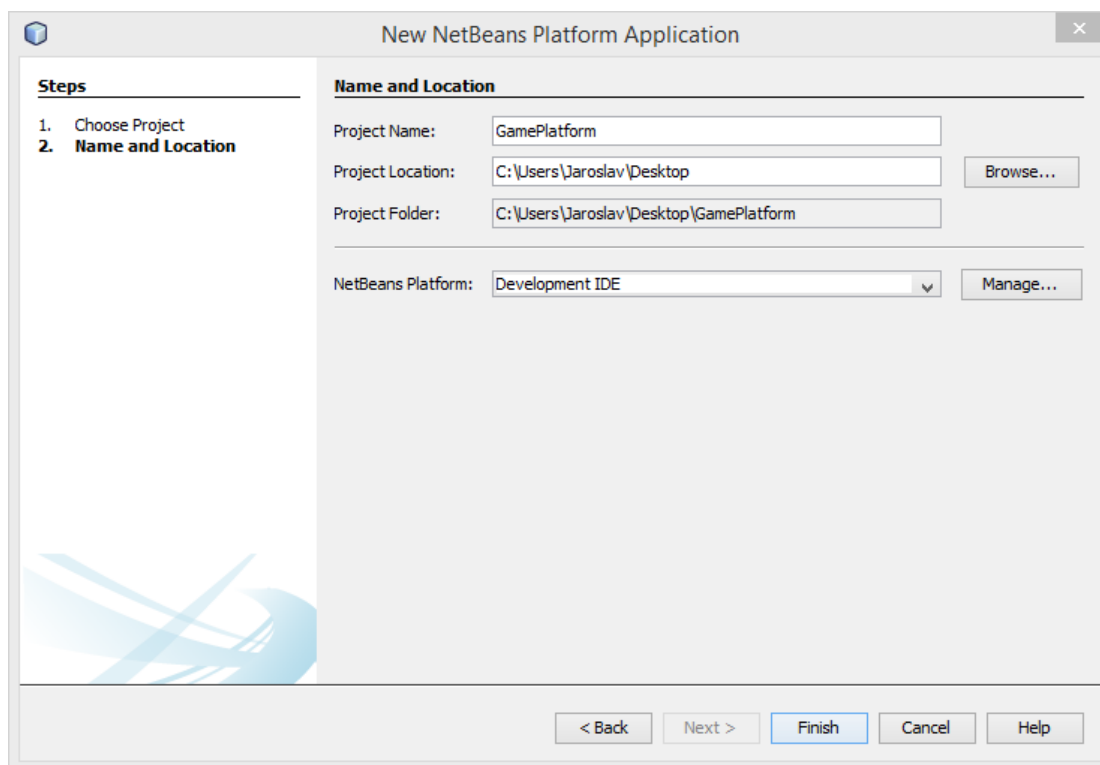
Projekt **Module** je pouze pro vytvoření prázdného modulu do platformy NetBeans a samostatně nemůže existovat. Musí být implementován buď do základního vývojového prostředí NetBeans nebo do vlastní platformy Netbeans. **Module Suite** projekt stejně jako projekt **Module** nemůže existovat samostatně. **Module Suite** projekt oproti **Module** obsahuje více modulů, které by měly být vyvíjeny spolu a tvořit jeden ucelený svazek. **Module Suite Library Wrapper** není nic jiného než obal neboli skupinou pro již existující JAR knihovny. Jedná se o obdobu **Module Suite**.

Pro vytvoření vlastní samostatné platformy je zde projekt **NetBeans Platform Application**. Může oproti všem ostatním existovat samostatně a může obsahovat jak projekty **Module Suite**, tak projekty **Module** či **Module Suite Library Wrapper**. Projekt **NetBeans Platform Application** může být neobvykle velký. Především při vytváření instalačního souboru. Velikost instalačního souboru může být v řádu gigabytů a je vhodné brát tuto skutečnost na vědomí. Není také vhodné použít automatické cloudové

úložiště např. OneDrive, které při opakované kompilaci projektu nestíhá aktualizovat na cloud a blokuje přístup k potřebným souborům. Dochází tak ke kolizím a nežádoucím chybám při importování existujících modulů.



Obrázek 8 Nový projekt NetBeans Platform Application

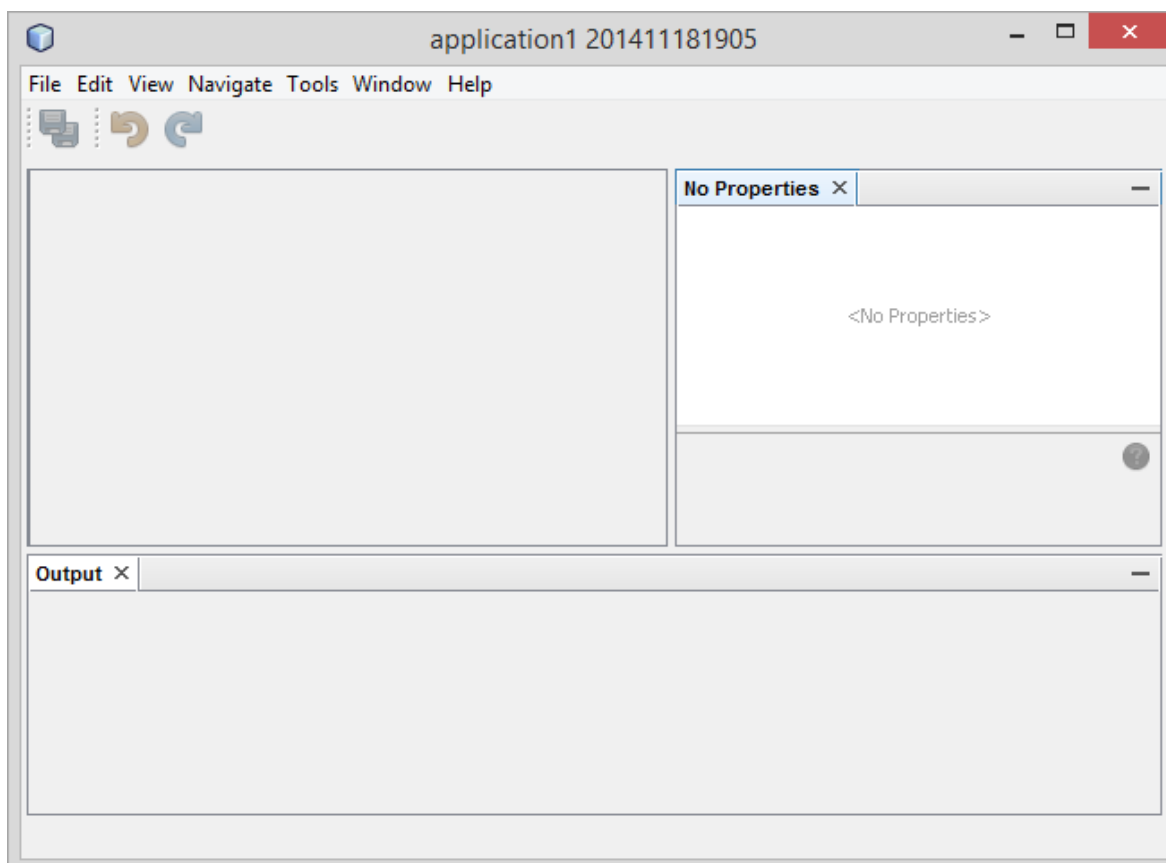


Obrázek 9 Zadání názvu a lokace nového projektu NetBeans Platform Application

Projekt NetBeans Platform Application v základu obsahuje moduly pro správu oken ze základního uzlu (Node) ide. Součástí projektu jsou také základní soubory obsahující informace o projektu. Tyto soubory jsou popsány v Tabulce 1. Obrázek 10 zachytává stav aplikace bez jakékoliv úpravy při prvním spuštění [10].

**Tabulka 1 Základní soubory při vytvoření nové platformy NetBeans**

Název	Popis
Build Script	Jedná se o dokument, jenž uchovává všechny akce týkající se projektu. Po stisknutí pravého tlačítka, je možné vidět všechny akce.
Project Properties	Definuje vlastnosti projektu jako například vlastní moduly, které jste vytvořili nebo které moduly patří do aplikace.
NetBeans Platform Config	Definuje token, který se používá při distribuci platformy.
Per-User NetBeans Platform Config	Ukazuje na soubor vlastností pro stavbu aplikace.



**Obrázek 10 Výchozí nastavení Netbean Platform Application**

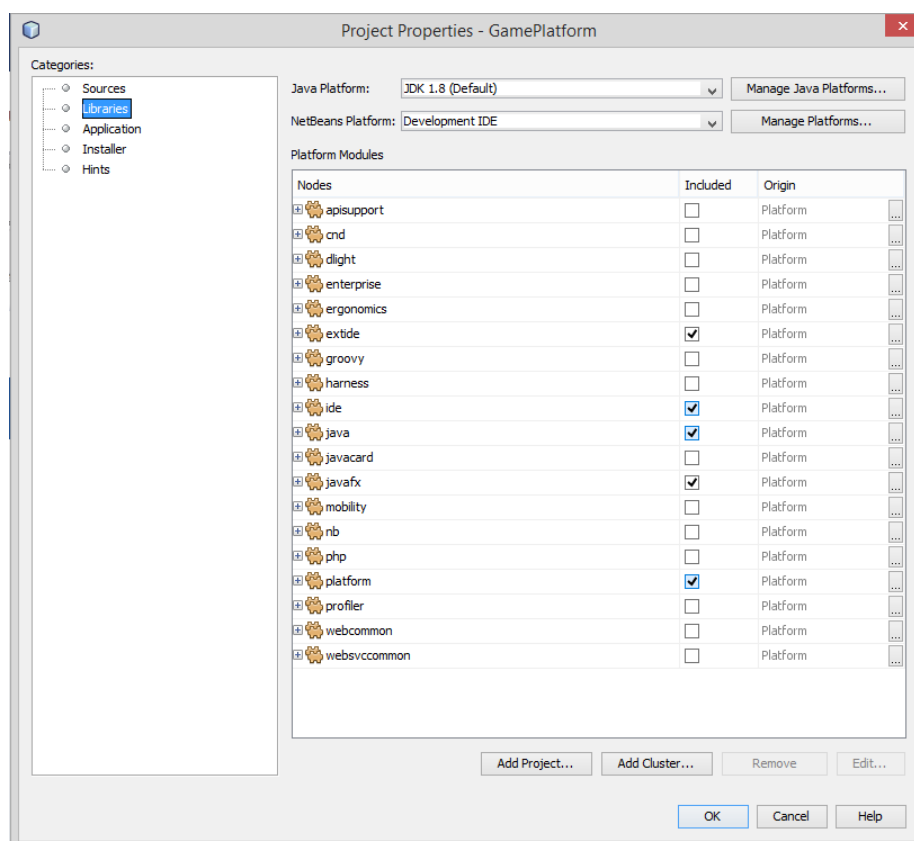
## 5.2 Základní tovární moduly a uzly

Uzlů a modelů je opravdu mnoho. Jejich užití je zcela subjektivní. Je možné použít již hotová řešení nebo nepoužít žádný a vytvořit si vše samostatně. Za základní uzly lze považovat uzly **ide** a **platform**. Tyto dva uzly obsahují správu oken projektů a ostatních známých funkcionalit známé z vývojového prostředí NetBeans.

Přidání továrních modulů je možné ve vlastnostech projektu. Vlastnosti projektu se zobrazí pravým kliknutím na projekt kolonka Properties - zobrazí se okno s vlastnostmi daného projektu. Pro zobrazení importovaných a dostupných základních modulů lze zjistit v kategorii **Libraries** zobrazenou na Obrázku 11. Mezi základní moduly patří **Project UI** a **Project API** z uzlu **ide**, jenž má na starosti projekty ve vývojovém prostředí NetBeans.

Pro přidání prohlížeče obrázků je zapotřebí přidat modul **User Utilites** a **Image** z uzlu **ide**. Modul **User Utilites** přidává do menu možnost otevřít soubor. Modul **Image** přidá filtr pro otevření základních typů obrázků a jejich zobrazení v platformě NetBeans.

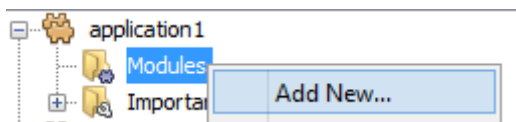
Dalším užitečným modulem, který by mohl přijít vhod, je z uzlu **platform** s názvem **Favorites**. Tento uzel přidává panel Favorites, který se vyskytuje ve vývojovém prostředí NetBeans. Modul Favorites přidává možnost okna obsahující oblíbené složky a umožňuje je procházet způsobem známým na příklad ze správce souborů ve Windows. Tím umožňuje snadnější správu souborů [10].



Obrázek 11 Přehled továrních uzlů

## 5.3 Vytvoření vlastních modulů

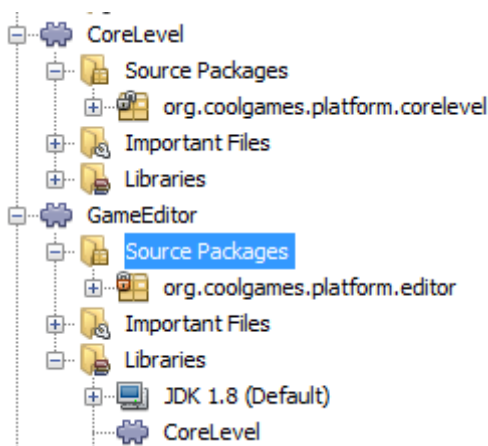
Vytvoření vlastního modulu je možné stejným způsobem jako vytvoření vlastní NetBeans Platform Application, který je popsán v kapitole 5.1. Nebo také snadnějším způsobem v případě již existujícího projektu NetBeans Platform Application nebo Module Suite. Za pomoci kliknutí pravého tlačítka myši na složku Modules a zvolení možnosti Add New.



Obrázek 12 Přidání modulu do existujícího projektu

Modul je samostatný projekt oddělený od celku, který může být součástí obalového projektu jako je NetBean Platform Application nebo Module Suite. Proto je zapotřebí zadat název projektu neboli název modulu a jeho umístění na disku. Dále Code Name Base, což není nic jiného, než název vlastního **package** známého z platformy Java.

Model by měl řešit pouze jednu problematiku, avšak může nastat situace, že modul bude potřebovat funkcionalitu jiného modulu nebo s ním komunikovat. Pro možnou komunikaci s určitým modulem je zapotřebí danému modulu přidělit práva balíčku public. Ve vlastnostech modulu v kategorii **API Versioning** je přehled všech balíčků (package) v daném modulu. Pro nastavení práva na public stačí pouze u daného balíčku zaškrtnout check box a potvrdit změnu vlastností modulu. Nyní stačí pouze přidat modul do knihovny [10].



Obrázek 13 Propojení modulů

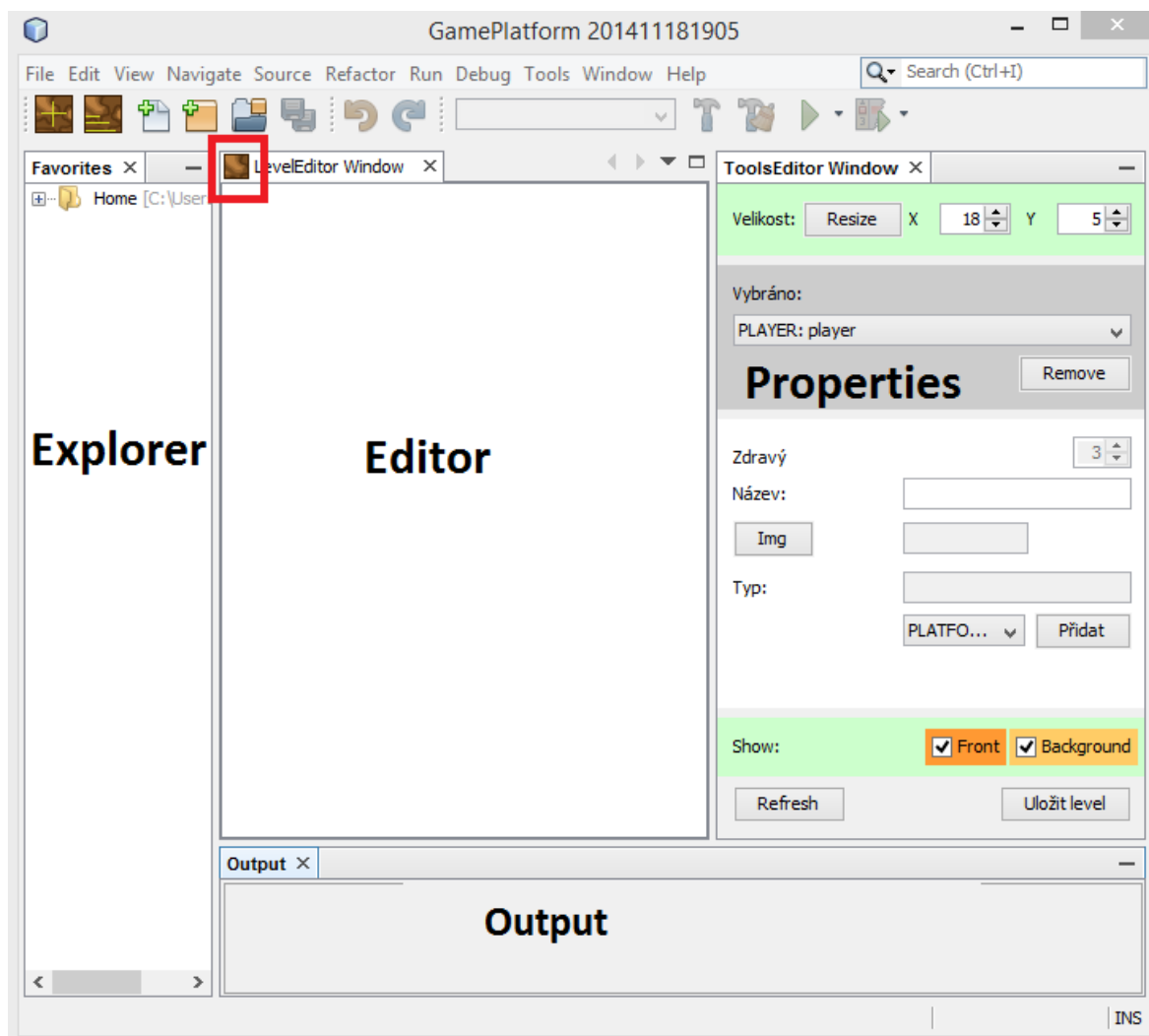
### 5.3.1 Window TopComonent

Pro komunikaci s uživatelem slouží GUI rozhraní. Přidání vlastního rozhraní do platformy NetBeans lze za pomoci okna. Okno lze vytvořit za pomoci wizard navigace. Okno může být zapnuto automaticky při startu aplikace s jeho výchozí pozicí v aplikaci.

Ukázka pozic oken je popsána na Obrázku 14. Mezi nejběžnější pozice patří.

- Editor,
- Explorer
- Output,
- Properties.

Také lze nastavit chování okna. Zda lze okno zavřít, oddokovat z pozice či se dá v rámci pozice okno posunout nebo prohodit s jiným oknem. K oknům lze nastavit ikonu pro lepší přehlednost v aplikaci. Podporované formáty obrázků na ikonu jsou gif a png. Ikona by měla mít velikost 16x16px [11].



Obrázek 14 Pozice oken v Platformě NetBeans

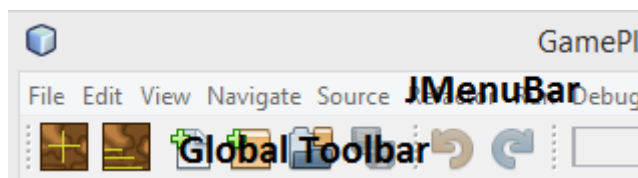
### 5.3.2 Vytvoření Action tlačítek

V rámci vlastní platformy lze vytvořit akční tlačítko neboli Action. Action může naslouchat vždy po celý běh aplikace nebo pouze na předem definované události.

Pro Action je zapotřebí nastavit typ odposlechu, nejjednodušší a nejvíce efektivní je stálý odposlech.

Nastavení kategorie slouží pro grupování akcí, lze nastavit již existující kategorii jako je **File** či **Edit** nebo vytvořit vlastní kategorii. Také lze přidat tlačítko do globálního **Toolbaru**, který se nachází ve výchozím nastavení přímo pod rolujícím menu aplikace nahoře neboli přímo pod pozicí JMenuBar. Jak vypadá globální Toolbar lze zjistit na Obrázku 15.

Při zvolení možnosti přidání tlačítka do globálního Toolbaru je požadováno nastavit ikonu tlačítka. V opačném případě je pouze doporučena. Podporované formáty obrázků na ikonu jsou gif a png. Ikona by měla mít dvě velikosti 16x16px a 24x24px. Při nastavení ikony je zapotřebí vybrat ikonu o velikosti 16x16px a ve stejné složce mít ikonu o velikosti 24x24px s názvem končící číslem 24, například mojeIkona24.png. Po nastavení názvu Action a názvu, který se bude zobrazovat na monitoru, jsou ikony automaticky zkopírovány do stejného balíku jako má právě vytvářená Action [12].

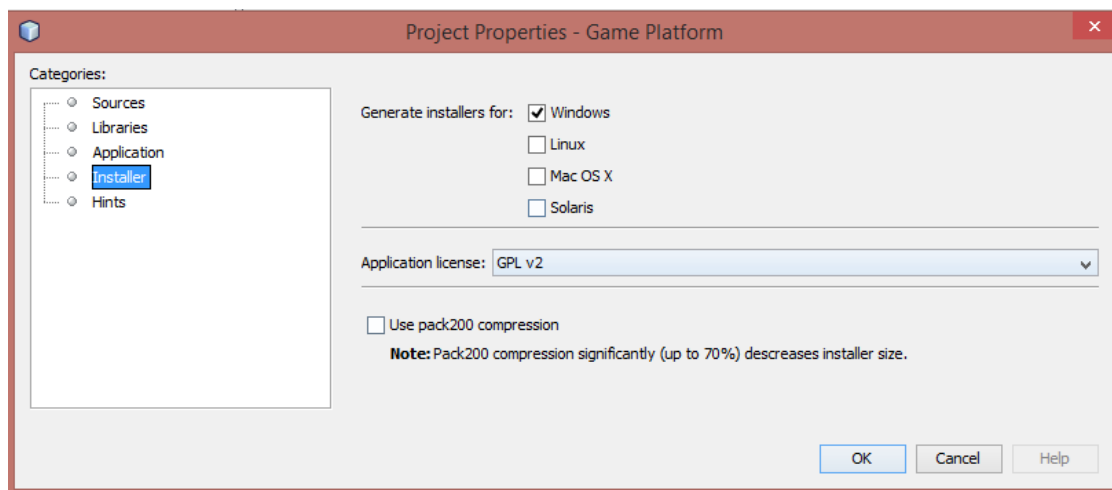


Obrázek 15 Action v Global Toolbar

## 5.4 Vytvoření instalačního souboru a značkování (Branding)

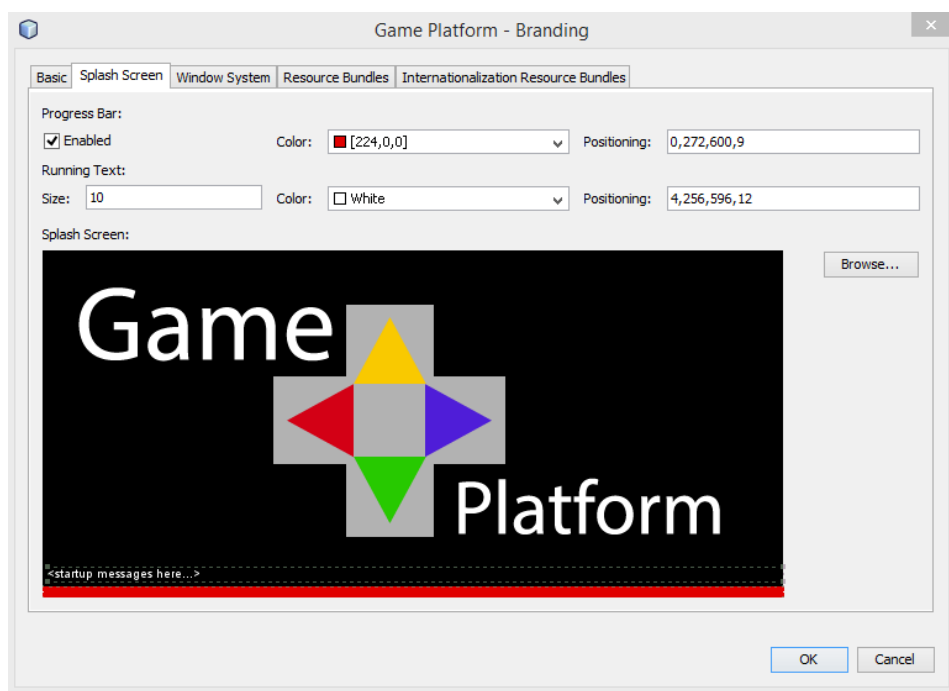
Aplikace vyvíjené pomocí NetBeans platformy určené pro samostatné šíření bez přímé vazby s původním vývojovým prostředím, lze šířit pomocí instalačního souboru. Instalační soubor vytváří samotné vývojové prostředí NetBeans pro OS Windows, Linux a Solaris.

Pro vytvoření instalačního souboru je zapotřebí nastavit ve vlastnostech projektu následující položky. V kategorii Application zvolit možnost vytvořit samostatnou aplikaci (Create standalone), a zadat značkovací název (Branding name). Tento název je zároveň nabídnut jako název výchozí složky při instalaci. V kategorii Installer lze zvolit, pro které operační systémy bude vytvořen instalační soubor. Nechybí zde ani nastavení licence pro vyvíjenou aplikaci. Licenci lze zvolit z již přednastavených licencí, např. GPL v2 nebo vložit vlastní textový soubor s licencí.



Obrázek 16 Nastavení vytváření instalačních souborů

Značkování (Branding) umožňuje nastavit základní vizuální komponenty aplikace např. ikonu aplikace, název aplikace zobrazovaný v titulku. Umožňuje především vytvoření vlastního vzhledu úvodní obrazovky (Splash Screen), která se zobrazuje při spouštění a načítání modulů aplikace. Zobrazení okna pro nastavení značkování zobrazené na Obrázku 17 lze provést pravým klikem na projekt a zvolením možnosti Branding.



Obrázek 17 Okno pro nastavení značkování (Branding)

## 6 Realizace vývojového prostředí Game Platform

Game Platform je vývojové prostředí založené na platformě NetBeans, které má sadu nástrojů pro tvorbu vlastních map určených pro 2D hry. Mapy jsou ukládány do XML souborů pro možnost pozdější editace a především pro možnosti importovat mapu do jakékoliv hry, která bude využívat tento nástroj pro vytváření map.

Dále prostředí umožňuje vytvářet projekty JavaFX a Java, pomocí nichž je možné následně vytvořit hry, které implementují vytvořené mapy. Využití výsledných map nemá přesné určení a mohou být použity jak pro Tower defense hry, tak pro hry typu Platformers. Využití mapy záleží pouze na konečném uživateli a jeho kreativitě.

Vývoj aplikace probíhal ve vývojovém prostředí NetBeans IDE 8.0.2 na platformě Netbeans Platform Application využívající platformy Java a JavaFX. Prostředí je distribuováno jako samostatný instalační soubor pro Windows, možnost vytvořit distribuci jak pro Mac OS X, Linux a Solaris.

### 6.1 Struktura prostředí

Struktura prostředí byla navržena především pro pozdější modifikace a rozšíření například o nové nástroje při tvorbě map nebo přidání designéru postavy či jiných aktivních prvků ve hře. Aplikace je tvořena dvěma vlastními moduly.

- **CoreLevel** – obsahující informace o aktuálním stavu tvořené mapy a rodičovské složce pro danou mapu. Pro komunikaci s ostatními moduly je veřejně přístupný balíček `org.coolgames.platform.corelevel`.
- **GameEditor** – obsahující nástroje pro tvorbu mapy. Komunikuje s modulem CoreLevel

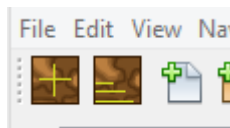
Také obsahuje dva tovární uzly od NetBeans pro vytvoření projektů JavaFX a Java.

- **Java** – obsahuje nástroje pro tvorbu projektů Java stejně jako ve vývojovém prostředí NetBeans.
- **JavaFX** – obsahuje nástroje pro tvorbu projektů Java stejně jako ve vývojovém prostředí NetBeans.

### 6.2 Action AddPlatformLevelListener & OpenPlatformMapListener

Tyto dvě Action se starají pouze o jediné dvě funkcionality. Vytváří tlačítko v globálním Toolbaru a naslouchají událostem na tyto tlačítka. Vyvolaná událost může být vytvoření nové mapy nebo otevření již existující mapy. Podle toho na jaké tlačítko uživatel klikl. Tlačítka jsou zobrazeny vlevo v global Toolbar na Obrázku 18.

Pro vytvoření nové mapy stačí pouze vybrat rodičovskou složku, neboli složku kde budou následně uloženy všechny použité obrázky v mapě včetně XML souborů uchovávajících stav mapy:



Obrázek 18 Tlačítka v global Toolbaru pro vytvoření a otevření mapy

### 6.3 Level Editor Window

Level Editor Window je součástí modulu GameEditor. Slouží pro vykreslování aktuálního stavu mapy v reálném čase. Pro vykreslování mapy byla zvolena platforma JavaFX. Jelikož jsou všechny nástroje pro tvorbu NetBeans platformy uzpůsobeny pro knihovnu Swing, nebylo možné použít GUI Builder pro zjednodušení realizace.

Pro propojení knihovny swing a JavyFX bylo zapotřebí použít JFXPanel, který umožňuje využívat technologii JavaFX ve Swingu. Pro správné fungování JavyFX je při každém větším úkonu, především při větším vykreslování například vytvoření nové mapy či načtení mapy, je tento úkon přidán do čekací fronty, kde je proveden za pomoci pomocného vlákna v metodě `runLater` ve třídě Platform. Třída Platform má na starosti komunikaci mezi platformami jako je například swing a JavaFX [13].

```
private void init() {
    jfxPanel = new JFXPanel();
    JScrollPane scroller = new JScrollPane(jfxPanel);
    add(scroller, BorderLayout.CENTER);

    Platform.setImplicitExit(false);
    Platform.runLater(new Runnable() {
        @Override
        public void run() {
            initScene();
        }
    });
}
```

#### 6.3.1 Vytvoření nové mapy

Level Editor Window obsahuje instanci třídy Level, která uchovává stav mapy. Obsahuje metody pro vytvoření, načtení, uložení či změnění velikosti mapy. Nová mapa je složená z bloků o velikosti 60x60px.

Mapa obsahuje dvě vrstvy přední (front) a zadní (background). Do obou vrstev jsou přidány bloky s výchozím vzhledem naslouchající události stisknutím tlačítka myši. Přední vrstva obsahuje bloky s průhledným vyplněním a zelenožlutým rámečkem. Zadní vrstva obsahuje bloky s bílou výplní a černým rámečkem. Počet bloků na šířku odpovídá velikosti konstanty `MIN_WIDTH` a počet bloků na výšku odpovídá velikosti konstanty `MIN_HEIGHT`.

```

public void createEmptyMap(Pane front, Pane background) {
    this.cleanLevel(front, background);
    for (int x = 0; x < LevelEditorTopComponent.getBlockX(); x++) {
        for (int y = 0; y < LevelEditorTopComponent.getBlockY(); y++)
        {
            backgroundBlocks.add(createBaseBlock(background, x * 60,
y * 60, 60, 60, BACKGROUND_COLOR, BACKGROUND_STROKE,
BACKGROUND_STROKE_WIDTH));
            frontBlocks.add(createBaseBlock(front, x * 60, y * 60,
60, 60, FRONT_COLOR, FRONT_STROKE, FRONT_STROKE_WIDTH));
        }
    }
}

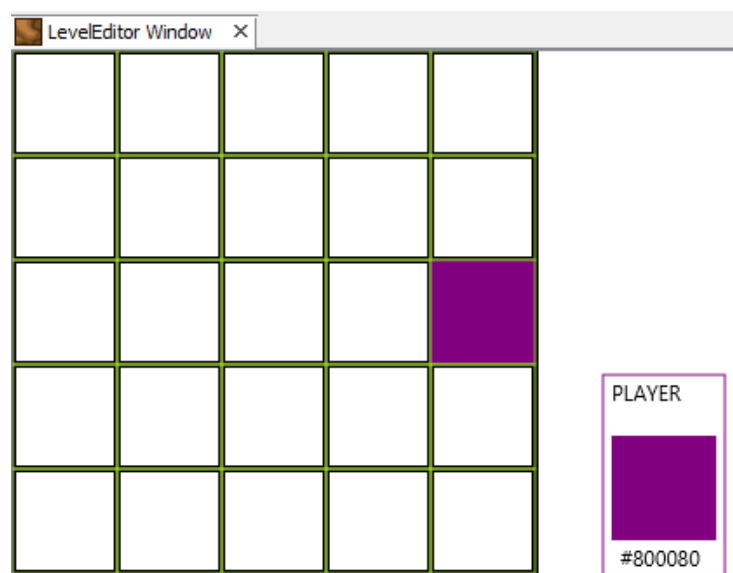
```

Celý zdrojový kód pro vytvoření prázdné nové mapy se nachází v příloze B.

### 6.3.2 Funkcionalita

Obsluha vykreslovací plochy je velice jednoduchá. Po vykreslovací ploše je možné se pohybovat za pomoci směrových šipek na klávesnici. Po pravé straně kurzoru myši je zobrazena lišta s informacemi o aktuálně vybraném bloku, který lze vidět na Obrázku 19. Pro použití bloku slouží levé tlačítko myši a pro jeho odstranění (vymazání z plochy) slouží pravé tlačítko myši.

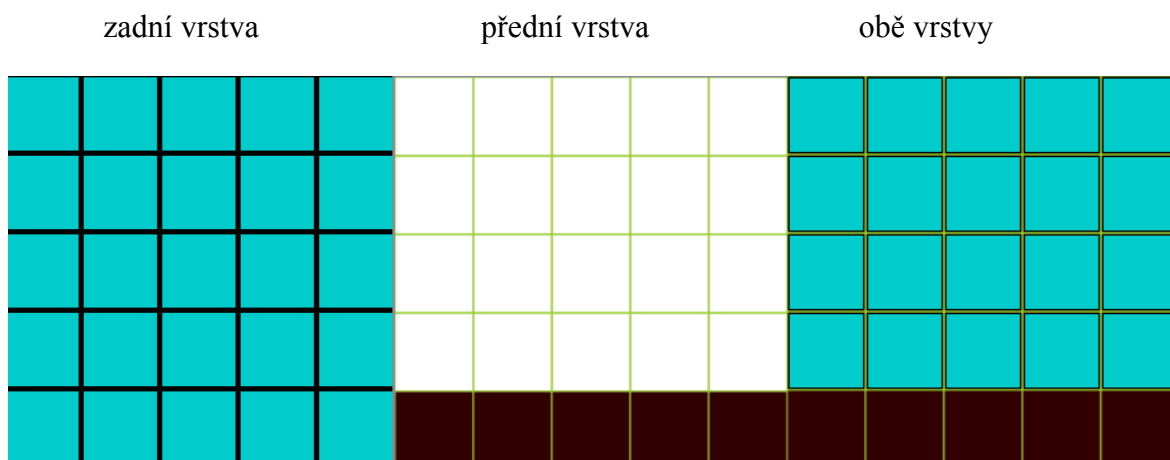
Velikost mapy lze nastavit jak na začátku, tak i během tvoření mapy, bez jakékoliv ztráty již zaznamenaných bloků. Ztráta bloků hrozí pouze v případě, že nová velikost mapy je menší než původní. Všechny bloky, které budou mimo nový rozsah mapy, budou ztraceny. Velikost mapy na šířku se rozšiřuje zleva doprava a snižuje zleva doprava. Na výšku se rozšiřuje z hora dolů a snižuje zdola nahoru.



Obrázek 19 LevelEditor Window

Jednou z hlavních výhod pro tvorbu mapy je možnost využít dvou vrstev. Primární využití vrstev záleží opět pouze na kreativě koncového uživatele. Doporučené použití je přední

vrstvu používat pouze pro aktivní prvky např. překážky, nepřátele, mince či jiné objekty, které mají vyvolat interaktivitu ve hře.



Obrázek 20 Ukázka vrstev

## 6.4 ToolsEditor Window

Jak již název napovídá, jedná se o nástrojové okno zobrazeno na Obrázku 21. Okno obsahuje nástroje pro vytváření a editování mapy. První položkou v nástrojovém okně je nástroj pro změnu velikosti mapy a jak zákon logiky napovídá, čím větší mapa, tím náročnější mapa je pro vykreslování i na vypracování.

Druhým nástrojem v okně je nástroj pro správu bloků. Lze vybrat jeden z definovaných bloků a použít jej pro tvorbu mapy nebo jej odebrat z definovaných bloků a tím jej smazat z celé mapy. Dále je zde zobrazována informace o aktuálním vybraném bloku.

Třetím nástrojem je nástroj pro vytvoření bloku. V současné verzi lze vytvořit 4 druhy bloku. PLAYER, ENEMY, PLATFORM a COIN. Minimum pro vytvoření bloku je vyplnění názvu bloku a jeho výchozí barvy. Lze také pro každý blok, vyjma bloku PLAYER, definovat vlastní obrázek. Obrázek má vyšší prioritu než barva a proto na mapě bude zobrazen blok s obrázkem, barva se zobrazí v případě nenalezení obrázku.

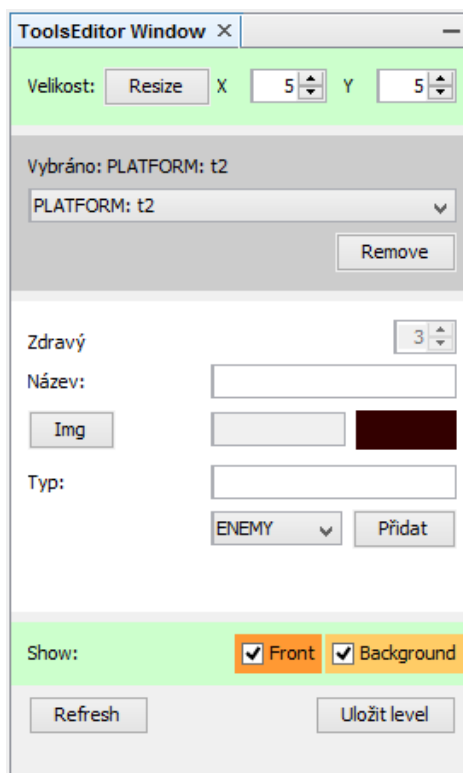
Pro blok ENEMY je také zapotřebí zadat jeho typ. Hodnota typ je čistě na uživateli a nijak neovlivňuje vývoj mapy. Slouží pouze pro hry, které danou mapu budou implementovat. Pro obrázky bloku je možné použít obrázky v příloze C.

Čtvrtým blokem nástrojů jsou dva check boxy pro vypínání a zapínání viditelnosti zadní či přední vrstvy na vykreslující ploše.

Pátým a v aktuální verzi posledním blokem jsou dvě tlačítka. Tlačítko refresh pro načtení bloku z aktuálně načtené mapy a tlačítko Uložit level sloužící pro uložení celé mapy do třech souborů XML

První soubor XML je pojmenovaný stejně jako rodičovská složka pro danou mapu. V souboru jsou zapsány informace o všech blocích použitých v mapě např. název, barva, obrázek, popřípadě jiné informace s blokem související. Druhý soubor XML s názvem

**front.xml** uchovává informace o pozici a druhu bloků z přední vrstvy. Třetí XML soubor s názvem **background.xml** obsahuje totožné informace pro zadní vrstvu.

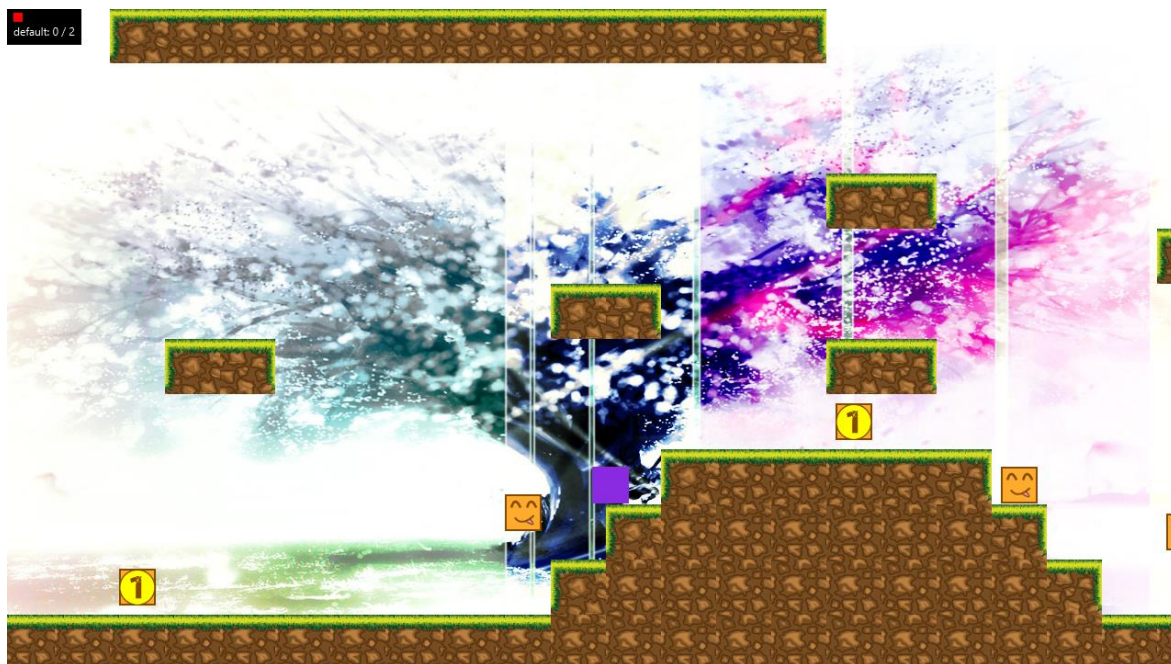


Obrázek 21 ToolsEditor Window

## 6.5 Vytvoření vlastní hry s názvem Platform

Hra je vytvořena za pomoci platformy JavaFX ve vlastním vývojovém prostředí Game Platform, která má stejné možnosti jako ve vývojovém prostředí NetBeans. Hra využívá vytvořených map nebo v tomto případě úrovní vytvořených za pomoci level editoru z vývojového prostředí Game Platform.

Jedná se o skákačí 2D hru zobrazenou na Obrázku 22, jejímž hlavním smyslem je přežít při sbírání všech mincí umístěných na aktuální úrovni. Po ukončení úrovně nebo úmrtí během sbírání mincí je aktuální hraná úroveň ukončena. Zobrazí se textové menu s následující nabídkou: pro opakování aktuální úrovně nebo pokračovat na další úroveň.



Obrázek 22 Hra Platforms

### 6.5.1 Specifikace úrovně pro hru Platform

Úroveň musí splňovat následující kritéria. Úroveň musí obsahovat hráče a minimálně jednu minci. Pokud úroveň nesplňuje kritéria je vyhodnocena jako ukončená. Optimální velikost úrovně pro hru Platforms je 40x40 bloků. V určitých velikostech úrovně dochází k zásekům kamery. Jelikož tato hra neprošla řádným testováním v praxi je málo informací k nápravě této závady a bylo by vhodné dodržovat optimální velikost úrovně.

Hráčův život se v každé úrovni může lišit. Velikost života je ve výchozím nastavení 3 životy, v Level Editoru lze tuto hodnotu změnit na maximální počet životů 20.

### 6.5.2 Typy nepřátel

V aktuální verzi hra podporuje dva typy nepřátel. Prvním typem je výchozí nepřítel ENEMY stojící nehybně na svém místě. V případě, že v souboru **front.xml** je zadán neznámý nepřítel, nahrazuje neznámého nepřítele výchozí nepřítel.

Druhý typ nepřítele je JUMP\_ENEMY. JUMP\_ENEMY nepřítel, jak název napovídá, skáče, ale aby jeho překonání nebylo příliš snadné, skáče v náhodných intervalech. Pro přidání nepřítele JUMP\_ENEMY ve vytváření úrovně za pomoci Level Editoru, stačí vyplnit při vytváření nového bloku druh ENEMY, zadat hodnotu typ na JUMP\_ENEMY a nyní stačí pouze nepřítele rozmístit po mapě.

Interakce mezi hráčem a nepřítelem dochází v okamžiku, kdy se hráč srazí s nepřítelem. Jelikož počítač je ohromně rychlý, během jedné kolize by odečetl všechny životy hráče. Z toho důvodu má hráč po kolizi s nepřítelem přibližně 2s imunitu. Tato imunita není na první pohled znát. Ověření funkčnosti imunity jde provést poměrně snadno. Při tvorbě úrovně stačí navýšit hráči počet životů například na 10. Poté stačí ve hře vyhledat nepřítele

a vyvolat kolizi a nijak ji nepřerušovat. V levém horním rohu dojde k postupné ztrátě životů. Zobrazení aktuální hodnoty života hráče je k vidění v levém horném rohu na Obrázku 22 nebo na Obrázku 23.



**Obrázek 23** Postup v úrovni default a přehled životů

### 6.5.3 Kolize

Kolize ve hře Platform je možná pouze s bloky mince, nepřítel a plošina. Instance těchto bloků jsou uloženy v příslušné kolekci ArrayList.

```
private ArrayList<Node> platforms = new ArrayList<>();
private ArrayList<Node> coins = new ArrayList<>();
private ArrayList<Sprite> enemys = new ArrayList<>();
```

V každém vykreslovacím cyklu dochází ke kontrole kolizí. V případě kolize hráče s blokem mince, je odebrán z ArrayListu coins. Ve chvíli, kdy ArrayList coins je prázdný, hráč úspěšně dokončil aktuální úroveň.

```
for (Node coin : actualLevel.getCoins()) {
    if
    (this.getNode().getBoundsInParent().intersects(coin.getBoundsInParent()))
    {
        coin.getProperties().put("alive", false);
    }
}

for (Iterator<Node> iterator = actualLevel.getCoins().iterator();
iterator.hasNext();) {
    Node coin = iterator.next();
    if (!(Boolean) coin.getProperties().get("alive")) {
        iterator.remove();
        actualLevel.getLevelRoot().getChildren().remove(coin);
    }
}
```

Kolize hráče s blokem enemy, dochází k odebrání jednoho života hráče. Pouze v případě, kdy hráč není imunní vůči poškození. Pokud hráč vyčerpall všechny životy úroveň končí neúspěšně.

```
enemys.stream().forEach((enemy) -> {
    enemy.update(this);
    enemy.collision(player);
});
```

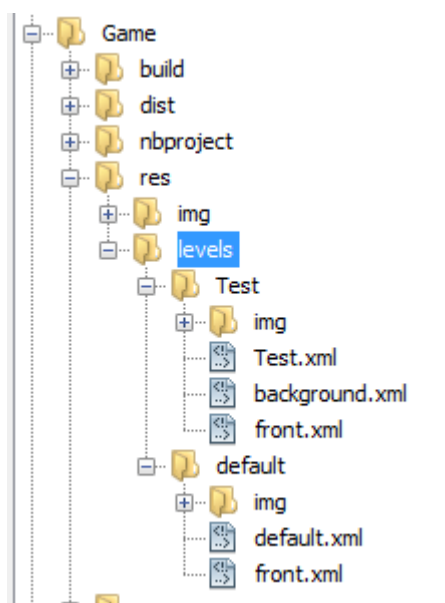
V poslední řadě existuje kolize hráče s plošinou (prostor určený pro pohyb hráče), omezuje pouze pohyb hráče. Zdrojový kód obsahující kolizi hráče s plošinou je v příloze D.

### 6.5.4 Vrstvy

Hra využívá obě vrstvy: přední i zadní vrstvu. Ovšem hra nemá žádnou interakci se zadní vrstvou. Zadní vrstva ve hře Platform slouží pouze ke grafickému zkrášlení dané úrovně. Z toho důvodu má hra ve skutečnosti pouze jednu vrstvu, a to pouze přední, která obsahuje i bloky zadní vrstvy. Bloky zadní vrstvy jsou přidány pouze kvůli vykreslování, ale už nejsou součástí žádné kolekce, která kontroluje kolizi mezi bloky a hráčem.

### 6.5.5 Přidání nové úrovně

Přidání nové úrovně do hry je velice snadné, stačí pouze zkopírovat rodičovskou složku zvolenou v Level Editoru do složky **levels**. Pokud je úroveň přidána během spuštění hry, je zapotřebí hru restartovat.



Obrázek 24 Přidání nové úrovně do hry

### 6.5.6 Ovládání hry

Ovládání postavy je za pomoci tlačítek **W** skok, **A** do leva, **D** doprava. Ovládání menu na konci po skončení úrovně je pomocí tlačítek: **mezerník** pro opakování úrovně a **enter** pro přechod na další úroveň. V případě, že hráč dohraje všechny dostupné úrovně, začíná se opět od začátku.

## 6.6 Distribuce

Celá výsledná aplikace neboli platforma je šířena za pomoci instalačního souboru. Stačí pouze nainstalovat a začít tvořit mapy a svoje vlastní mini hry. Ukázkovou hru Platform je možné také spustit pomocí souboru JAR, anebo otevřít přímo projekt s hrou ve vývojovém prostředí Game Platform a provádět vlastní modifikace a vylepšení. To vše je již na koncovém uživateli.

## **Závěr**

Cílem praktické části bylo vytvoření prostředí pro tvorbu her na platformě NetBeans s pomocí platform Java a JavaFX s využitím XML souborů. Při vytváření byl kladen důraz především na budoucí rozšíření.

V práci se mi povedlo dosáhnout všech vytyčených cílů. Potýkal jsem se s velkými potížemi, především při vývoji aplikace Game Platform. Postupně jsem však závažné chyby dokázal odstranit a uvést do použitelného provozu. Aplikace by si žádala delší čas na vývoj, především testování a zpracování podnětů od uživatelů. Jediný výskyt kritické chyby se projevoval pouze na notebooku, nikoliv však na desktopu.

Zkušenosti získané během vytváření hry Platformer jsou velmi cenné a určitě se zkušenosti z platformy JavaFX budou v budoucnu velmi hodit. Zkušenosti s vytvářením vlastní platformy NetBeans jsou určitě taky velmi ceněné, ale jak již bylo řečeno dříve, z důvodu nahrazení swingu JavouFX je velmi pravděpodobné, že si budu muset danou zkušenost zopakovat.

Rozšíření aplikace je celá řada, od vytváření celých postav pro hru nebo vytváření vodících čar, například pro umělou inteligenci. Cenným přínosem by mohl být generátor kódu například pro pohyb postavy nebo řešení kolize.

## Literatura

- [1] Hra.-. Wikipedia: the free encyclopedia [online]. San Francisco (CA): Wikimedia Foundation [cit. 2015-05-12]. Dostupné z: <http://cs.wikipedia.org/wiki/Hra>
- [2] NetBeans IDE Features. 2015. Netbeans [online]. [cit. 2015-05-12]. Dostupné z: <https://netbeans.org/features/index.html>
- [3] A Brief History of NetBeans. 2015. Netbeans [online]. [cit. 2015-05-12]. Dostupné z: <https://netbeans.org/about/history.html>
- [4] License Notices. 2015. Netbeans [online]. [cit. 2015-05-12]. Dostupné z: <https://netbeans.org/about/legal/product-licences.html>
- [5] Java Platform, Standard Edition. 2008. Wikipedia: the free encyclopedia [online]. San Francisco (CA): Wikimedia Foundation [cit. 2015-05-12]. Dostupné z: [http://en.wikipedia.org/wiki/Java\\_Platform,\\_Standard\\_Edition](http://en.wikipedia.org/wiki/Java_Platform,_Standard_Edition)
- [6] JavaFX. 2014. Wikipedia: the free encyclopedia [online]. San Francisco (CA): Wikimedia Foundation [cit. 2015-05-12]. Dostupné z: <http://cs.wikipedia.org/wiki/JavaFX>
- [7] Swing (Java). 2001. Wikipedia: the free encyclopedia [online]. San Francisco (CA): Wikimedia Foundation [cit. 2015-05-12]. Dostupné z: [http://cs.wikipedia.org/wiki/Swing\\_%28Java%29](http://cs.wikipedia.org/wiki/Swing_%28Java%29)
- [8] Extensible Markup Language. 2001. Wikipedia: the free encyclopedia [online]. San Francisco (CA): Wikimedia Foundation [cit. 2014]. Dostupné z: [http://cs.wikipedia.org/wiki/Extensible\\_Markup\\_Language](http://cs.wikipedia.org/wiki/Extensible_Markup_Language)
- [9] 5. díl - Zápis XML souborů SAXem v Javě. 2014. Itnetwork [online]. [cit. 2015-05-12]. Dostupné z: <http://www.itnetwork.cz/java-tutorial-zapis-xml-souboru-sax>
- [10] WEXBRIDGE, Jason a Walter NYLAND. NetBeans Platform for Beginners: Modular Application Development for the Java Desktop [online]. 1. vyd. Leanpub, 2014 [cit. 2014-10-20]. Dostupné z: <https://leanpub.com/nbp4beginners>
- [11] NetBeans Platform Quick Start. 2014. NetBeans [online]. [cit. 2015-05-12]. Dostupné z: <https://platform.netbeans.org/tutorials/nbm-quick-start.html>
- [12] NetBeans Platform Plugin Quick Start. 2014. NetBeans [online]. [cit. 2015-05-12]. Dostupné z: <https://platform.netbeans.org/tutorials/nbm-google.html>
- [13] Class Platform. 2015. Docs.oracle [online]. [cit. 2015-05-12]. Dostupné z: <https://docs.oracle.com/javase/8/javafx/api/javafx/application/Platform.html>

- [14] DEA, Carl, Mark HECKLER, Gerrit GRUNWLD, Jose PEREDA a Sean M PHILLIPS. Pro JavaFX 8: a definitive guide to building rich java clients. 2. vyd. Berkeley: Apress, 2014. ISBN 978-143-0265-740.
- [15] SHARAN, Kishori. Beginning Java 8 language features: interfaces, inner classes, threads, i/o and collections. Berkeley: Apress, 2014. ISBN 978-143-0266-587.

## Příloha A – Ukázka zpracování XML

```
package javafxapplication11;

import java.io.File;
import java.io.FileReader;
import java.io.FileWriter;
import javax.xml.stream.XMLInputFactory;
import javax.xml.stream.XMLOutputFactory;
import javax.xml.stream.XMLStreamConstants;
import javax.xml.stream.XMLStreamReader;
import javax.xml.stream.XMLStreamWriter;

/**
 *
 * @author Jaroslav.Janíček.CZE.2015.Cool-Games
 */
public class XMLExample {

    public void writeFile(String nameFile) {
        XMLStreamWriter xsw = null;

        try {
            xsw =
XMLOutputFactory.newInstance().createXMLStreamWriter(new
FileWriter(nameFile + ".xml"));
            xsw.writeStartDocument();
            xsw.writeStartElement("root");

            xsw.writeStartElement("clovek");
            xsw.writeAttribute("vek", "25");
            xsw.writeAttribute("jmeno", "Adam");
            xsw.writeEndElement();

            xsw.writeEndElement();
            xsw.writeEndDocument();
            xsw.flush();
        } catch (Exception e) {
            System.err.println("Errorr write: " + e.getMessage());
        } finally {
            try {
                if (xsw != null) {
                    xsw.close();
                }
            } catch (Exception e) {
                System.err.println("Error close: " + e.getMessage());
            }
        }
    }

    private void loadFile(String nameFile) {
        XMLInputFactory factory = XMLInputFactory.newInstance();
        XMLStreamReader xsr = null;
        try {
```

```

        xsr = factory.createXMLStreamReader(new FileReader(nameFile +
".xml"));
        String element = "";

        while (xsr.hasNext()) {
            if (xsr.getEventType() ==
XMLStreamConstants.START_ELEMENT) {
                element = xsr.getName().getLocalPart();
                switch (element) {
                    case "clovek":
                        System.out.println("Vek: " +
xsr.getAttributeValue(0)
                                + " Jemno: " +
xsr.getAttributeValue(1));
                        break;
                }
            } else if (xsr.getEventType() ==
XMLStreamConstants.CHARACTERS) {
                element = "";
            } else if ((xsr.getEventType() ==
XMLStreamConstants.END_ELEMENT)) {
                }
            xsr.next();
        }

    } catch (Exception e) {
        System.err.println("Error read file: " + nameFile + ".xml " +
e.getMessage());
    } finally {
        try {
            xsr.close();
        } catch (Exception e) {
            System.err.println("Error when close file: " + nameFile +
".xml " + e.getMessage());
        }
    }
}
}

```

## Příloha B – Zdrojový kód pro vytvoření nové mapy.

```
private ArrayList<MapBlock> backgroundBlocks = new ArrayList<>();
private ArrayList<MapBlock> frontBlocks = new ArrayList<>();

public void createEmptyMap(Pane front, Pane background) {
    this.cleanLevel(front, background);
    for (int x = 0; x < LevelEditorTopComponent.getBlockX(); x++) {
        for (int y = 0; y < LevelEditorTopComponent.getBlockY(); y++)
        {
            backgroundBlocks.add(createBaseBlock(background, x * 60,
y * 60, 60, 60, BACKGROUND_COLOR, BACKGROUND_STROKE,
BACKGROUND_STROKE_WIDTH));
            frontBlocks.add(createBaseBlock(front, x * 60, y * 60,
60, 60, FRONT_COLOR, FRONT_STROKE, FRONT_STROKE_WIDTH));
        }
    }
}

private MapBlock createBaseBlock(Pane pane, int x, int y, int width, int
height, Color fillColor, Color strokeColor, double strokeWidth) {
    Rectangle entity = new Rectangle(width, height);
    entity.setTranslateX(x);
    entity.setTranslateY(y);
    entity.setFill(fillColor);
    entity.setStroke(strokeColor);
    entity.setStrokeWidth(strokeWidth);
    MapBlock block = new MapBlock(x / 60, y / 60, entity,
EntityType.NONE);

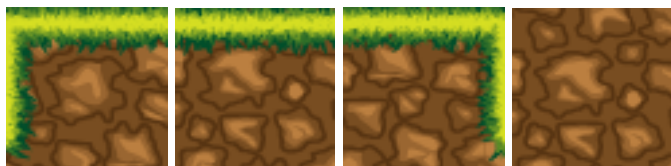
    block.rectangle.setOnMouseClicked((MouseEvent event) -> {
        if (event.getButton() == MouseButton.PRIMARY) {
            EntityProperties en =
LevelProperties.getActualEntityProperties();
            if (en == null) {
                return;
            }
            block.entityType = en.entityType;
            block.name = LevelProperties.getActualName();
            if (en instanceof EnemyProperties) {
                block.type = ((EnemyProperties) en).getType();
            }

            if (en instanceof LevelProperties.PlayerProperties) {
                block.heart = ((LevelProperties.PlayerProperties)
en).getHealth();
            }

            changeEntyty(block.rectangle, en.color, en.img);
        } else if (event.getButton() == MouseButton.SECONDARY) {
            block.entityType = EntityType.NONE;
            changeEntyty(block.rectangle, fillColor, null);
            block.rectangle.setStroke(strokeColor);
            block.rectangle.setStrokeWidth(strokeWidth);
        }
    });

    pane.getChildren().add(block.rectangle);
    return block;
}
```

## Příloha C – Základní obrázky pro tvorbu mapy



## Příloha D – Kolize hráče s plošinou

```
// vybrání směru pohybu ve třídě Players
if (KeyboardAction.isPressed(KeyCode.W) && this.node.getTranslateY() >=
5) {
    jump();
}

    if (KeyboardAction.isPressed(KeyCode.A) &&
this.node.getTranslateX() >= 5) {
        moveX(-5, actualLevel.getPlatforms());
    }

    if (KeyboardAction.isPressed(KeyCode.D) &&
this.node.getTranslateX() + 40 <= actualLevel.getLeveWidth() - 5) {
        moveX(5, actualLevel.getPlatforms());
    }

protected void jump() {
    if (this.isCanJump()) {
        //playerVelocity = playerVelocity.add(0, -30);
        this.velocity = velocity.add(0, jumpSize);
        this.canJump = false;
    }
}

// metody pro zjištění kolizí v abstraktní třídě sprite
roTECTED void setCanJump(boolean canJump) {
    this.canJump = canJump;
}

protected void jump() {
    if (this.isCanJump()) {
        //playerVelocity = playerVelocity.add(0, -30);
        this.velocity = velocity.add(0, jumpSize);
        this.canJump = false;
    }
}

protected void moveX(int value, ArrayList<Node> platforms) {
    boolean movingOnRight = value > 0;

    for (int i = 0; i < Math.abs(value); i++) {
        for (Node platform : platforms) {
            if
(this.node.getBoundsInParent().intersects(platform.getBoundsInParent()))
{
                if (movingOnRight) {
                    if (this.node.getTranslateX() + 40 ==
platform.getTranslateX()) {
                        return;
                    }
                } else {
                    if (this.node.getTranslateX() ==
platform.getTranslateX() + 60) {
                        return;
                    }
                }
            }
        }
    }
}
```

```

        this.node.setTranslateX(this.node.getTranslateX() +
(movingOnRight ? 1 : -1));
    }
}

protected void moveY(int step, ArrayList<Node> platforms) {
    boolean movinOnDown = step > 0;
    for (int i = 0; i < Math.abs(step); i++) {
        for (Node platform : platforms) {
            if
(this.node.getBoundsInParent().intersects(platform.getBoundsInParent()))
{
                if (movinOnDown) {
                    if (this.node.getTranslateY() + 40 ==
platform.getTranslateY()) {

this.node.setTranslateY(this.node.getTranslateY() - 1);
                    this.canJump = true;
                    return;
                }
            } else {
                if (this.node.getTranslateY() ==
platform.getTranslateY() + 60) {
                    return;
                }
            }
        }
    }

    this.node.setTranslateY(this.node.getTranslateY() +
(movinOnDown ? 1 : -1));
}
}

```