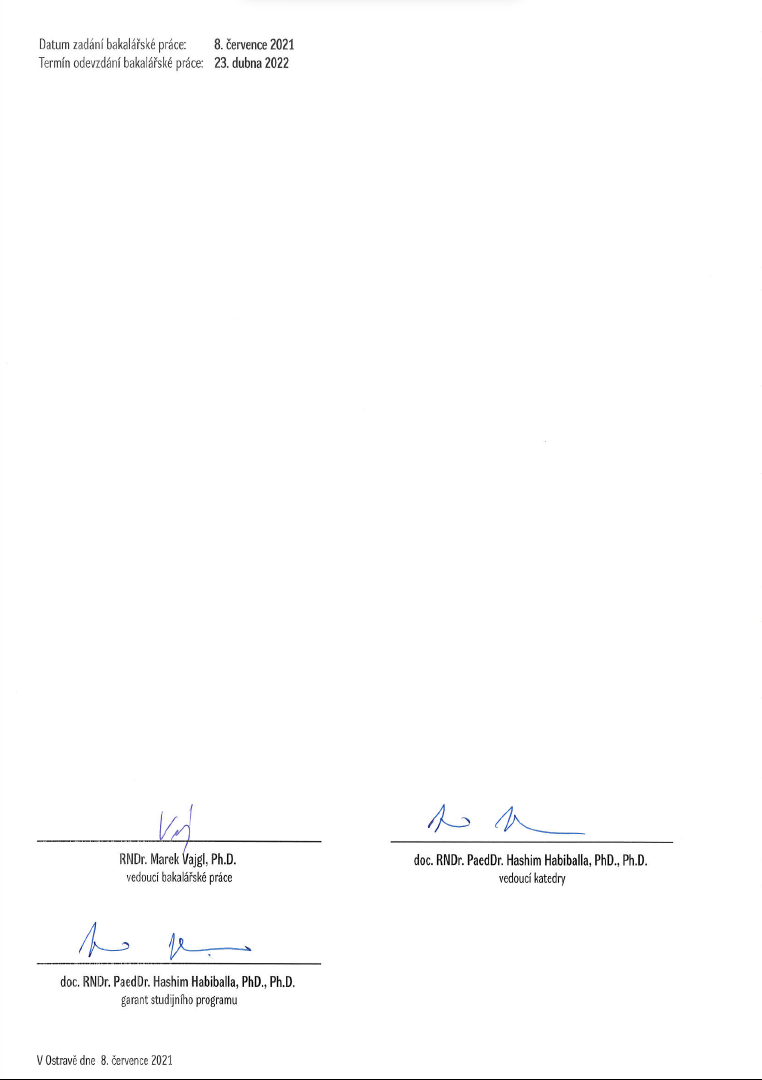
|  |
| --- |
| OSTRAVSKÁ UNIVERZITA  PŘÍRODOVĚDECKÁ FAKULTA  KATEDRA INFORMATIKY A POČÍTAČŮ |
| Plugin pro IDEA pro generování diagramu tříd do PlantUML  BAKALÁŘSKÁ PRÁCE |
| Autor práce: Marie Šrámková  Vedoucí práce: RNDr. Marek Vajgl, Ph.D. |
| 2022 |

|  |
| --- |
| UNIVERSITY OF OSTRAVA  FACULTY OF SCIENCE  DEPARTMENT OF INFORMATICS AND COMPUTERS |
| A plugin for IDEA to generate class diagram into PlantUML  BACHELOR THESIS |
| Author:  Marie Šrámková  Supervisor:  RNDr. Marek Vajgl, Ph.D. |
| 2022 |





Cílem práce je realizovat konfigurovatelný plugin pro vývojové prostředí IDEA, který bude umět převádět třídy definované v projektu do diagramu tříd jazyka PlantUML.

Řešení bude moci u otevřeného projektu vybrat oblast, která se má zahrnout do generování, a další parametry generování (vybraní členové, detailnost atd). Následně bude umět nástroj vygenerovat diagram do jazyka PlantUML. Konfigurace generování bude možno uložit.

Přínosem práce je vytvoření vlastního generovacího přístupu ze zdrojových kódu Java s možností poměrně podrobné specifikace generovaného obsahu.

Postup práce:

1) Definice cílů a analýza současného stavu

2) Definice metodiky řešení práce

3) Vytvoření vlastního řešení

4) Shrnutí vytvořeného řešení, přínosy

5) Shrnutí cílů práce, závěr

ABSTRAKT

Cílem této bakalářské práce je vytvořit konfigurovatelný plugin v prostředí IntelliJ IDEA, který bude do složky projektu generovat class diagram podle zvolených parametrů a který do složky projektu také uloží konfiguraci pro daný projekt pro příští generování. Plugin po kliknutí na kořenovou složku projektu zobrazí vstupní formulář, který umožní uživateli zvolit v otevřeném projektu atributy, třídy, metody a další parametry pro generování PlantUML class diagramu. Diagram se po vygenerování uloží a zobrazí. Zvolené hodnoty se uloží pro příští generování a při dalším generování se zobrazí možnost využití poslední zvolené konfigurace. Plugin také umožní uživateli spravovat již vzniklé konfigurace (smazání, změna názvu konfigurace, apod.).

*Klíčová slova:*

*(klíčová slova vypsaná na řádku, oddělená od sebe čárkami)*

**ABSTRACT**

The text of the abstract.

*Keywords:*

čestné prohlášení

Já, níže podepsaný/á student/ka, tímto čestně prohlašuji, že text mnou odevzdané závěrečné práce v písemné podobě je totožný s textem závěrečné práce vloženým v databázi DIPL2.

Ostrava dne

………………………………

podpis studenta/ky

|  |
| --- |
| Poděkování |
| Prohlašuji, že předložená práce je mým původním autorským dílem, které jsem vypracoval/a samostatně. Veškerou literaturu a další zdroje, z nichž jsem při zpracování čerpal/a, v práci řádně cituji a jsou uvedeny v seznamu použité literatury.  V Ostravě dne . . . . . . . . . . . .  . . . . . . . . . . . . . . . . . .  (podpis) |

OBSAH

[ÚVOD 8](#_Toc103134815)

[1 Definice základních pojmů 9](#_Toc103134816)

[1.1 Co je to plugin 9](#_Toc103134817)

[1.2 Co znamená výraz „konfigurovatelný“ 9](#_Toc103134818)

[1.3 Prostředí IntelliJ IDEA 9](#_Toc103134819)

[2 Cíle práce a analýza současného stavu 10](#_Toc103134820)

[2.1 Cíle práce 10](#_Toc103134821)

[2.2 Analýza současného stavu 10](#_Toc103134822)

[3 Metodika řešení práce 14](#_Toc103134823)

[4 Realizace řešení práce 15](#_Toc103134824)

[4.1 Analýza uživatelů 15](#_Toc103134825)

[4.2 Požadavky – user stories 15](#_Toc103134826)

[4.3 Definice rizik 16](#_Toc103134827)

[4.4 Implementace řešení 18](#_Toc103134828)

[4.4.1 Tvorba pluginu 18](#_Toc103134829)

[4.4.2 Tvorba formulářů 18](#_Toc103134830)

[4.4.3 Generování konfiguračního souboru 18](#_Toc103134831)

[4.4.4 Generování UML diagramu podle parametrů zvolených ve formuláři 18](#_Toc103134832)

[5 Představení vytvořeného řešení 19](#_Toc103134833)

[RESUMÉ 20](#_Toc103134834)

[SUMMARY 21](#_Toc103134835)

[SEZNAM POUŽITÉ LITERATURY 22](#_Toc103134836)

[SEZNAM POUŽITÝCH SYMBOLŮ 23](#_Toc103134837)

[SEZNAM OBRÁZKŮ 24](#_Toc103134838)

[SEZNAM TABULEK 25](#_Toc103134839)

[SEZNAM PŘÍLOH 26](#_Toc103134840)

ÚVOD

Technologie se již několik let šíří tak, že zasahují téměř do každého odvětví. S rostoucím počtem použití roste také jejich komplexita. Abychom se my či vývojáři těchto technologií v tak komplexních technologiích vyznali, používají se různé způsoby, jak popsat problematiku (textové znázornění, grafické znázornění a další způsoby). Vhodný a jednoduchý způsob vyjádření problematiky je pomocí grafického znázornění – diagramů neboli grafů. Diagramy jsou kombinací textu a obrázků, což nám pomáhá se v popisu určité problematiky lépe vyznat.

S problémem orientace v komplexních projektech se setkává většina vývojářů a programátorů. Popis projektu může napomoci jak programátorům vyznat se ve svém vlastním projektu a předejít tak možným chybám, tak i velkým firmám, uvnitř kterých kolaboruje více lidí na jednom projektu. V tomto případě může vhodný diagram ulehčit workflow mezi jednotlivými zaměstnanci, odděleními apod., a tak zlepšit chod firmy a její prosperitu.

//proč zrovna diagram tříd?

Tento plugin slouží primárně programátorům, kteří vyvíjí své projekty v prostředí IntelliJ IDEA v jazyce Java, kteří používají UML a jejich diagramy, jako je např. diagram tříd.

Ve své bakalářské práci se zabývám problematikou generování diagramu tříd pomocí konfigurovatelného pluginu v prostředí IntelliJ IDEA, který generuje diagram ze zdrojových kódů v jazyce Java. Tento plugin by měl napomoci řešení výše zmíněného problému orientace v komplexním projektu tak, že si uživatel vygeneruje vlastní diagram tříd a uvidí tak přehlednější grafické zobrazení svého projektu. Pro usnadnění generování je plugin konfigurovatelný, což znamená, že je možné si uložit parametry generování a jednoduše je spravovat. //proč zrovna diagram tříd a ne si to psát na papír

1. Definice základních pojmů

Pro pochopení této bakalářské práce je potřeba si předem ujasnit pár základních pojmů, abychom se v problematice orientovali.

* 1. Co je to plugin

Plugin, nebo také plug-in, je označení pro zásuvný modul, což je software, který přidáváme k jinému softwaru za účelem jeho rozšíření, či

* 1. Co znamená výraz „konfigurovatelný“

Text

* 1. Prostředí IntelliJ IDEA

Text

1. Cíle práce a analýza současného stavu
   1. Cíle práce

Cílem této bakalářské práce je vytvořit konfigurovatelný plugin pro prostředí IntelliJ IDEA, který bude generovat diagram tříd podle zvolených parametrů a umožní uložit konfiguraci pro daný projekt pro příští generování. Plugin po kliknutí na složku projektu zobrazí vstupní formulář, který umožní uživateli zvolit v otevřeném projektu parametry pro generování PlantUML diagramu tříd a uložení konfigurace. Při uložení konfigurace generování se zvolené hodnoty uloží pro příští generování a při dalším generování se zobrazí možnost využití poslední zvolené konfigurace. Uživatel si sám zvolí, zda bude chtít vygenerovat pouze diagram nebo bude chtít také uložit konfigurační soubor. Diagram se po vygenerování uloží a zobrazí. Plugin také umožní uživateli spravovat již vzniklé konfigurace (smazání, změna názvu konfigurace, apod.).

* 1. Analýza současného stavu

V úvodu jsme si vysvětlili, co je to diagram a k čemu nám slouží. Michal Obluk se problematikou diagramů okrajově zabývá v práci [1].

Pro práci bylo zvoleno vytvoření generátoru diagramu tříd PlantUML pro programovací jazyk Java v prostředí IntelliJ IDEA. Programovací jazyk Java je objektově orientovaný (více o OOP např. v práci [2]), tzn. že všechny objekty reálného světa můžeme reprezentovat jako objekty (instance třídy) a seskupovat je do tříd (entit). K popisu projektu můžeme využít více typů diagramů (diagram aktivit, diagram tříd, které jsou popsány v práci [1], sekvenční diagram a další typy diagramů). Pro práci byl zvolen diagram tříd [3], který popisuje základní stavební prvky objektů programovacího jazyka Java.

V rámci průzkumu aktuálního stavu v problematice generování diagramu tříd do PlantUML jsem se zaměřila na dva základní okruhy – analýzu již existujícího komplexního řešení a analýzu obecných metod, přístupů a nástrojů.

Z první části (analýzy existujících řešení) jsem zjistila, že již existuje nástroj pro zobrazení diagramu PlantUML ve vývojovém prostředí IntelliJ IDEA. Plugin PlantUML integration [4] umožňuje zobrazit textový soubor puml v diagramu PlantUML, tzn. v grafické podobě. Práce tento plugin nezahrnuje z důvodu ztráty přehlednosti. Pokud bychom se rozhodli vygenerovat více diagramů najednou a chtěli bychom je rovnou zobrazit, mohlo by nám to v liště otevřených souborů otevřít všechny, což v některých případech může znamenat i desítky nových záložek či i větší počet. Nevhodné by také bylo se pro každý diagram dotazovat, zda jej chceme také zobrazit. Pro větší počet generování by to znamenalo zdlouhavé zamítání či povolování. Jelikož již tento plugin existuje a je volně dostupný ve vývojovém prostředí IntelliJ IDEA a v práci není použit z výše zmíněných důvodů, poukazuji na jeho možné využití pro zobrazení námi vygenerovaného diagramu tříd.

V roce 2020 byl navržen a vydán plugin PlantUML Parser [5], jenž umožňuje generovat diagram PlantUML za pomoci výše zmíněného pluginu. Nyní už však ne na základě manuálně napsaného puml souboru, ale už přímo ve vývojovém prostředí. Plugin lze najít ve vývojovém prostředí při kliknutí na projekt. Zde najdeme možnost PluntUML Parser, jež nám otevře dialogové okno, ve kterém konkretizujeme podmínky generování. Tento plugin však nenabízí potřebné funkcionality a je nedostačující. Chybějícími funkcionalitami (chybějící funkcionality) z výše zmíněného pluginu, které tvoří také požadavky, jsou:

* volba umístění vygenerovaného diagramu a konfiguračního souboru a jejich následné uložení
* volba názvu vygenerovaného diagramu a konfiguračního souboru
* uložení konfigurace generování (parametry generování) pro jednotlivé složky/soubory/moduly/projekty
* načtení uložených konfigurací
* možnost výběru generování tříd/rozhraní
* možnost výběru generování vnitřních tříd
* definice, v jakém jazyce bude výsledný soubor vygenerován (formát „Java“, nebo formát diagramu tříd UML)
* možnost generování všech vybraných projektů najednou
* možnost generování všech projektů pro jednotlivé moduly
* možnost definovat komentář
* možnost definovat orientaci zobrazení tříd (levá/pravá)

V tabulce č. 1 jsem vybrala pár základních parametrů, které budu porovávat.

|  |  |  |
| --- | --- | --- |
| Parametry | PlantUML Parser | Vlastní plugin |
| Generování diagramu tříd | Ano | Ano |
| Uložení konfiguračního souboru | Ne | Ano |
| Vlastní volba umístění a názvu pro diagram tříd | Ne | Ano |
| Vlastní volba umístění a názvu pro konfigurační soubor | Ne | Ano |
| Správa konfigurací | Ne | Ano |
| Načtení starých konfigurací | Ne | Ano |
| Generování podle starých konfigurací | Ne | Ano |

Tabulka 1 - srovnání PluntUML Parseru s vlastním řešením

Z tabulky č. 1 vidíme, že náš plugin bude na rozdíl od existujícího řešení konfigurovatelný. Do tabulky jsem nezanášela parametry generování, ve kterých se naše řešení bude také lišit. Ty jsou uvedený v tabulce č. 2.

|  |  |  |
| --- | --- | --- |
| Parametry generování | PlantUML Parser | Vlastní plugin |
| Třídy | Ne | Ano |
| Rozhraní | Ne | Ano |
| Balíčky | Ne | Ano |
| Třídní atributy | Ano | Ano |
| Třídní metody | Ano | Ano |
| Vnitřní třídy | Ne | Ano |
| Atributy pro rozhraní | Ne | Ano |
| Metody pro rozhraní | Ne | Ano |
| Modifikátory pro třídy | Ne | Ano |
| Modifikátory pro rozhraní | Ne | Ano |
| Modifikátory pro položky třídy | Ano | Ano |
| Modifikátory pro položky rozhraní | Ne | Ano |

Tabulka 2 - srovnání PluntUML Parseru s vlastním řešením (parametry generování)

Z tabulky č. 2 vidíme, že náš plugin je detailnější a umožňuje uživateli podrobnější výběr. V takovém případě však musíme počítat s možnými výkonnostními a také paměťovými nároky.

1. Metodika řešení práce
2. Realizace řešení práce

Tato kapitola je zaměřena na počáteční části vývoje, podle kterých se budeme řídit v části implementace řešení. Počátečními částmi vývoje jsou myšleny kroky, které činíme, abychom předešly chybám a nemuseli celé řešení zahazovat a sestavovat od začátku. Tato část má v některých zdrojích svá pojmenování, v unified process (jednotý proces) se setkáme s fázemi. Počáteční části vývoje bychom mohli tedy přirovnat počáteční fázi (inception phase) a dále fázi zpracování (elaboration phase), které jsou definovány v tzv. jednotném procesu.

* 1. Analýza uživatelů

Uživateli používajícími náš plugin budou lidé, kteří budou pracovat v prostředí IntelliJ IDEA a budou potřebovat převést svůj projekt psaný v jazyce Java do vizuální formy v podobě PlantUML diagramu tříd. Ať už uživatelé pracují s prostředím na úrovni začátečníka, či profesionála, práce s pluginem by měla být jednoduchá a intuitivní, avšak plugin by měl být dostatečně komplexní, aby dokázal uspokojit požadavky uživatelů.

* 1. Požadavky – user stories

Požadavky chápeme jako specifikaci bodů, kterých se budeme snažit dosáhnout ve vývoji. Určují nám, co se od řešení očekává a stanovují, od čeho se ve vývoji budeme dále odrážet. Nedostatečná specifikace požadavků je častým důvodem neúspěchu.

//shrnout cíle – rozlišit text uml od grafického zobrazení a konkretizovat konfig. soubor

Požadavky lze specifikovat různými způsoby, např. use case diagramem, scénáři, pomocí user stories. User stories popisují požadavek z pohledu zákazníka způsobem, jako by si jej psal on sám. Požadavek však musí být minimalistický.

|  |
| --- |
| 1. Jako uživatel chci vytvořit PlantUML diagram tříd, abych mohl svůj projekt kdykoliv převést ve vizuální formě. |
| 1. Jako uživatel chci uložit PlantUML diagram tříd, abych si jej mohl opakovaně zobrazit. |
| 1. Jako uživatel si chci zvolit umístění generovaného PlantUML diagramu tříd, abych mohl soubor následně separátně použít či přidat jej do projektového repozitáře. |
| 1. Jako uživatel chci zvolit, jak se bude generovaný PlantUML diagram tříd jmenovat, abych i podle názvu souboru mohl poznat, o jaký soubor se jedná. |
| 1. Jako uživatel chci zvolit, co má PlantUML diagram tříd zahrnovat, abych mohl přizpůsobit obsah vygenerovaného diagramu různým situacím. |
| 1. Jako uživatel si chci uložit zvolené parametry generování, abych si mohl PlantUML diagram tříd vygenerovat opakovaně. |
| 1. Jako uživatel chci vidět výpis všech konfiguračních souborů, které jsem si uložil, abych se v nich zorientoval. |
| 1. Jako uživatel chci mít možnost odstranit již vytvořené konfigurace ve vývojovém prostředí, abych je nemusel hledat. |
| 1. Jako uživatel chci mít možnost znovu zobrazit parametry generování podle uložené konfigurace ve vývojovém prostředí, abych mohl provést opětovné generování bez nutnosti opakovaného vyplňování. |

* 1. Definice rizik

Při vývoji projektu je zapotřebí brát v úvahu rizika, která mohou ohrozit jeho úspěšné dokončení. Tato rizika je nutné si uvědomit již na počátku a pokusit se je redukovat. Pozdní zjištění, že se vyskytuje riziko, jež nejsme schopni vyřešit, může vést k prodloužení vývoje, či k úplnému selhání. Ve svém vývoji jsem nalezla rizika, která jsem zobrazila v tzn. risk-listu. Ten obsahuje mimo jiné také úroveň složitosti, dopad, který by riziko mohlo způsobit při jeho nevyřešení a také způsob řešení.

|  |  |  |  |
| --- | --- | --- | --- |
| Popis | Úroveň složitosti | Dopad | Řešení |
| 1. Neznalost tvorby pluginů, nedostatečné zkušenosti. | vysoká | velký | Vytvoření jednoduchého pluginu. |
| 1. Komplikované vytvoření vstupního formuláře. | vysoká | velký | Vytvoření vstupního formuláře se základními funkcionalitami. |
| 1. Komplikované vytvoření algoritmu pro procházení .java souborů v projektu. | vysoká | velký | Vytvoření prototypu algoritmu pro procházení .java souborů v projektu. |
| 1. Komplikované vytvoření algoritmu pro procházení souboru projektu a ukládání důležitých (ve formuláři zvolené) hodnot. | vysoká | velký | Vytvoření jednoduchého algoritmu pro čtení souboru a ukládání hodnot ze souboru. |
| 1. Komplikované vygenerování a uložení souboru podle zjištěných a uložených údajů.   //rozdělit | vysoká | velký | Uložit soubor na určité místo, s určitým názvem.  //neeliminuje - specifikovat |
| 1. Komplikace při správě konfiguračního souboru.   //co za komplikace? konkretizovat | vysoká | velký | Uložit informace vyplněné ve vstupním formuláři, znovu je vytáhnout a zobrazit ve formuláři. |
| 1. Složitá orientace, návaznost na řešení. Složité opravy. | nízká | nízký | Zkontrolovat funkcionalitu a čitelnost, popř. přepsat na čistý kód.  //refactoring, pravidla kódu |
| 1. Nefunkčnost.   //kdy je to nefunkční? | nízká | střední | Otestovat na více verzích prostředí a různých zařízeních.  Otestovat vytíženost, nároky na paměť, apod. //specifikovat víc, ne apod. |
| 1. Špatné, či složité použití.   //snaha o přehlednost a čistý design | nízká | nízký | Otestovat, zda je práce s pluginem intuitivní (zda splňuje požadavky na UX (user experience)). |

* 1. Implementace řešení

Při implementaci řešení jsem vycházela z požadavků zákazníka a také z risk-listu, který je znázorněn v kapitole č. 4.3- Definice rizik. Po definování rizik jsem začala jednotlivá rizika minimalizovat v pořadí podle dopadu a následně dále rozvíjet řešení do finální podoby.

* + 1. Tvorba pluginu

Při tvorbě pluginu jsem vycházela z návodu popsaném na webových stránkách   
IntelliJ Platform Plugin SDK [8].

* + 1. Tvorba formulářů
    2. Generování konfiguračního souboru
    3. Generování UML diagramu podle parametrů zvolených ve formuláři

1. Představení vytvořeného řešení

//dělá to tohle takhle

RESUMÉ

//metodika funguje, aplikace funguje

SUMMARY

//zda jsem splnila požadavky

SEZNAM POUŽITÉ LITERATURY

1. **Příjmení, Jméno.** *Název knihy.* Město vydání: Vydavatelství, 2003. 123-4-56-789123-4.

2. **Příjmení1, Jméno1 a Příjmení2, Jméno2.** Název webové stránky. *Název webu.* [Online] Produkční společnost, 23. Září 2006. [Citace: 19. Září 2008.] http://www.urladresa.cz. 12-3456-789-12.

[1] OBLUK, Michal. *Modelování Business procesů s pomocí UML: BUSINESS PROCESS MODELING WITH UML*. Ostrava, 2006. Bakalářská. Ostravská univerzita. Vedoucí práce Lukasík Petr.

[2] Stuart Hirshfield and Raimund K. Ege. 1996. Object-oriented programming. ACM Comput. Surv. 28, 1 (March 1996), 253–255. DOI:https://doi.org/10.1145/234313.234415

[3] Berardi, Daniela & Calvanese, Diego & De Giacomo, Giuseppe. (2005). Reasoning on UML class diagrams. Artificial Intelligence. 168. 70-118. 10.1016/j.artint.2005.05.003.

[4] STEINBERG, Eugene, MAMONTOV, Ivan, Henady ZAKALUSKY, Max GORBUNOV, Vojtěch KRÁSA a Andrew KOROLEV, ed. PlantUML integration [online]. In: . [cit. 2021-11-16]. Dostupné z: https://plugins.jetbrains.com/plugin/7017-plantuml-integration/

[5] SHUZIJUN. *PlantUML Parser* [online]. In: . [cit. 2021-11-16]. Dostupné z: https://plugins.jetbrains.com/plugin/15524-plantuml-parser

[6] A GroupLayout Example. *Oracle Java Documentation: The Java™ Tutorials* [online]. [cit. 2022-01-03]. Dostupné z: <https://docs.oracle.com/javase/tutorial/uiswing/layout/groupExample.html>

[7] How to Use GroupLayout. *Oracle Java Documentation: The Java™ Tutorials* [online]. [cit. 2022-01-03]. Dostupné z: <https://docs.oracle.com/javase/tutorial/uiswing/layout/group.html>

[8] Creating Your First Plugin. *IntelliJ Platform Plugin SDK* [online]. [cit. 2022-04-06]. Dostupné z: https://plugins.jetbrains.com/docs/intellij/getting-started.html

SEZNAM POUŽITÝCH SYMBOLŮ

|  |  |  |
| --- | --- | --- |
| ABC |  | Význam první zkratky. |
| B |  | Význam druhé zkratky. |
| C |  | Význam třetí zkratky. |
|  |  |  |

SEZNAM OBRÁZKŮ

SEZNAM TABULEK

SEZNAM PŘÍLOH