Západočeská univerzita v Plzni

Fakulta aplikovaných věd

Katedra informatiky a výpočetní techniky

Diplomová práce

Plugin do Eclipse pro tvorbu GUI s konfigurovatelným generováním zdrojového kódu

Plzeň, 2013 Zdeněk Tumpach

Prohlášení

Prohlašuji, že jsem diplomovou práci vypracoval samostatně a výhradně s použitím citovaných pramenů.

V Plzni dne 12.5.2013 ……

Zdeněk Tumpach

Poděkování

Tímto bych chtěl poděkovat vedoucímu mé diplomové práce panu Ing. Tomáši Potužákovi za jeho podporu, přátelský přístup a cenné rady při vypracování této práce. Dále bych rád poděkoval své rodině za velkou podporu během studia.

Abstract

This work provides detailed information on the development plugin for the Eclipse IDE. It further describes the basic components and layout managers of the Java Swing library. The second part contains a description of the created plugin. It allows users to create a GUI with a configurable generating source code. The user can generate a source code by individual placement of all components constituting the source code or by using of a general template. The template can be also designed by using this plugin. The created plugin with the Eclipse IDE is saved on the DVD.

Obsah

1 Úvod

Každý programátor vyvíjející nějakou aplikaci, používá jiné strukturování zdrojového kódu. Někteří vývojáři člení zdrojový kód do více metod, jiným nevadí napsat celý program do jedné řádky. Jednu vlastnost má ale většina programátorů společnou. Tou je snaha co nejvíce si usnadnit často jednotvárnou práci, k čemuž lze využít řadu nástrojů či integrovaných vývojových prostředí. Ta kromě standardních pomůcek jako je zvýrazňování syntaxe či našeptávání zdrojového kódu často nabízejí i nástroje umožňující „naklikání“ grafického uživatelského rozhraní místo psaní jeho zdrojového kódu. Výhodou je urychlení tvorby rozhraní. Nevýhodou je pak často nepřehledný či nepochopitelný vygenerovaný zdrojový kód, který se může výrazně lišit od představ programátora. Struktura vygenerovaného kódu lze totiž zřídka nastavit.

Tato diplomová práce je zaměřena na tvorbu pluginu do vývojového prostředí Eclipse. Plugin bude umožňovat tvorbu GUI s konfigurovatelným generováním zdrojového kódu. Bude tedy možné v rámci možností nastavit, jak bude vygenerovaný zdrojový kód vypadat. Z toho vyplývá, že plugin bude rozdělen na dvě části. První částí bude rozhraní umožňující navržení vlastního GUI a následné individuální rozmístění všech komponent, akcí, metod atd. Druhou částí bude vytváření všeobecných šablon, dle kterých se následně bude generovat zdrojový kód.

V první části této diplomové práce popíši vývojové prostředí Eclipse včetně jeho architektury. Následně se zaměřím na popis všeobecného pluginu a jeho realizaci ve vývojovém prostředí Eclipse. Dále se zaměřím na základní řadu komponent knihovny Java Swing. Závěrem teoretické části bude popsáno a porovnáno několik existujících nástrojů pro tvorbu GUI.

V druhé části nejprve specifikuji jednotlivé požadavky a následně provedu analýzu celého projektu. Následně detailně popíši vytvoření a otestování samotného Eclipse pluginu. Závěrem bude ověřena funkčnost a použitelnost na konkrétním případě.

2 Eclipse

Eclipse je open source vývojová platforma, která je spíše známa jako integrované vývojové prostředí (IDE) určené pro programování v jazyce Java. Projekt Eclipse (Eclipse 1.0) vznikl zveřejněním kódu IBM pod EPL licencí [1].

V rámci tohoto projektu byl vyvinut grafický framework SWT. Velkou výhodou SWT je nativní vzhled všech aplikací na každé platformě, kde je SWT portován (SWT využívá nativního kódu operačního systému). Naproti tomu konkurenční framework Swing využívá pouze služby JVM. To umožňuje lepší přenositelnost (omezení je dáno pouze dostupností Javy pro danou platformu) [1].

V základní verzi obsahuje Eclipse pouze integrované prostředky pro vývoj standardní Javy např. debugger, kompilátor, atd., ale neobsahuje například nástroj pro návrh grafických uživatelských rozhraní – toto rozšíření je nutné dodat pomocí pluginu. Z tohoto důvodu souběžně s vývojem Eclipse vznikly takzvané subprojekty, které vytvářejí rozšíření pro jednotlivé oblasti softwarového vývoje v Javě. Tyto subprojekty zajišťují integraci potřebných rozšíření do samotného vývojového prostředí Eclipse [1].

2.1 Verze Eclipse

V současné době (květen 2013) je možné využívat tyto aktuální verze Eclipse [2]:

- Eclipse IDE for Java EE Developers
- Eclipse Classic 4.2.2
- Eclipse IDE for Java Developers
- Eclipse IDE for C/C++ Developers
- Eclipse for Mobile Developers
- Eclipse IDE for Java and DSL Developers
- Eclipse IDE for Java and Report Developers
- Eclipse for RCP and RAP Developers
- Eclipse Modeling Tools
- Eclipse for Testers
- Eclipse IDE for Automotive Software Developers
- Eclipse for Scout Developers
- Eclipse for Parallel Application Developers

2.2 Architektura Eclipse

V této části popíšeme vnitřní a vnější architekturu integrovaného vývojového prostředí Eclipse.

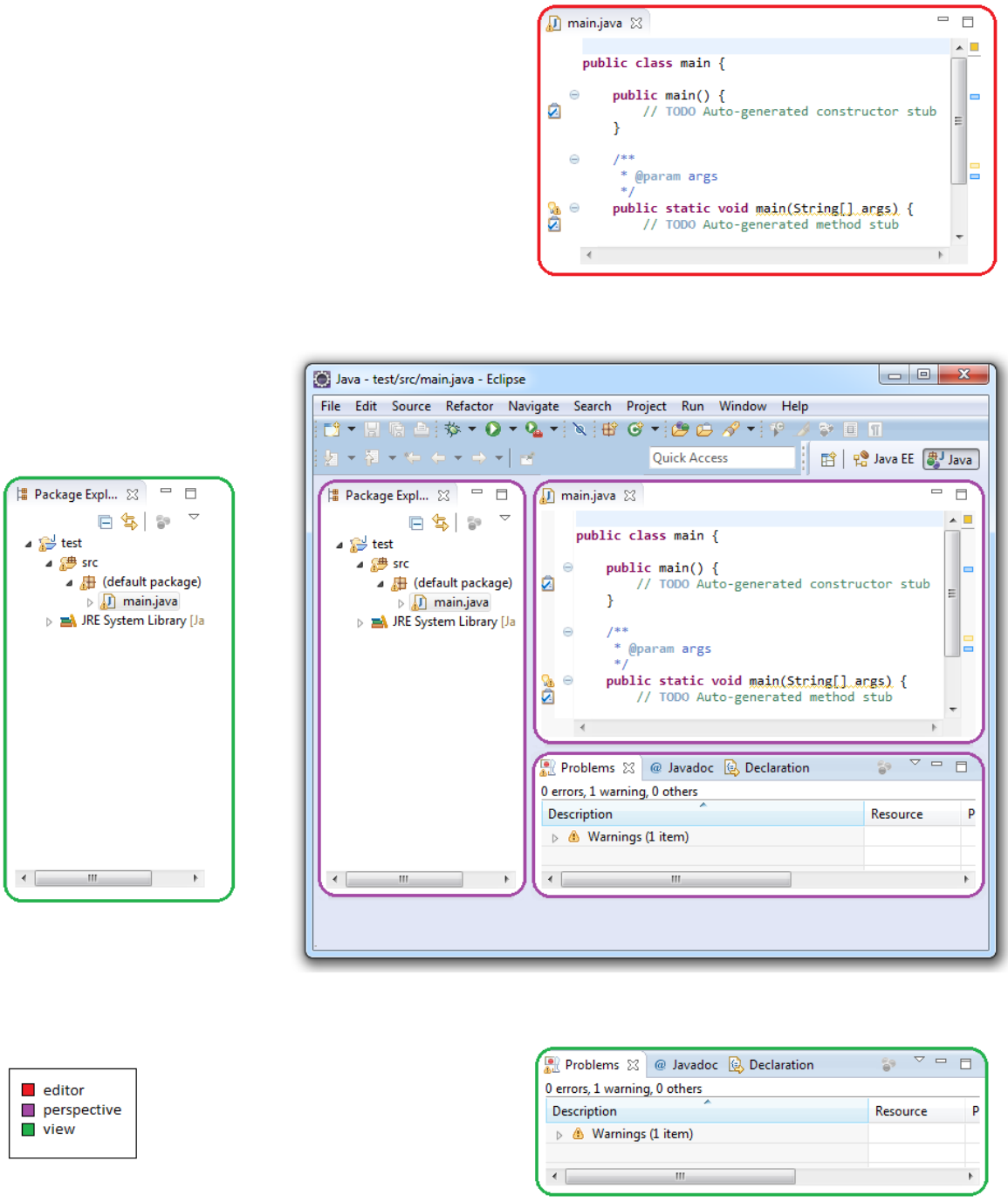
2.2.1 Vnitřní architektura Eclipse

Eclipse je postaven jako množství pluginů, které definují svoji API a jsou na sobě navzájem závislé. Základem pro tuto architekturu je implementace komponentového modelu *OSGi* [3] – tzv. komponentový framework – Equinox [4]. Díky tomuto frameworku mohou být pluginy nahrávány dynamicky. Ačkoliv Eclipse používá termín „Plugin“ v OSGi modelu se používá termín *bundle*. Oba pojmy však znamenají to samé – softwarovou komponentu. Celková funkčnost Eclipse může být rozšířena pomocí knihoven nebo rozšířením platformy [3].

2.2.2 Vnější architektura Eclipse

Vývojové prostředí Eclipse je tvořeno pomocí pohledů (view) a editorů (editor). Společně tyto prvky vytvářejí perspektivy (perspective) [5].

- *Editor* je používán k úpravě jednotlivých elementů. Ve vývojovém prostředí Eclipse je to například editor Java tříd nebo HTML. Editory mohou mít funkci kompletace kódu, podporu změny předešlých úprav (zpět / vpřed), atd. [5].
- *Pohled* můžeme zjednodušeně popsat jako vizuální součást, která slouží pro snadnou, rychlou, tedy efektivní práci v IDE. Používá se zpravidla k navigaci v hierarchii informací nebo k otevření editoru. Úpravy provedené v pohledu se okamžitě ukládají [5].
- *Perspektivu* si můžeme představit jako seskupení pohledů a editorů (viz Obrázek 1). Uvnitř každé perspektivy je možné měnit pozice a velikosti jednotlivých editorů a pohledů. Mimo toho je také možné libovolně otevírat a zavírat jednotlivé části perspektivy. Důležitý je fakt, že v jeden okamžik může být aktivní pouze jedna perspektiva [5].



Obrázek 1: Ukázka perspektivy v Eclipse

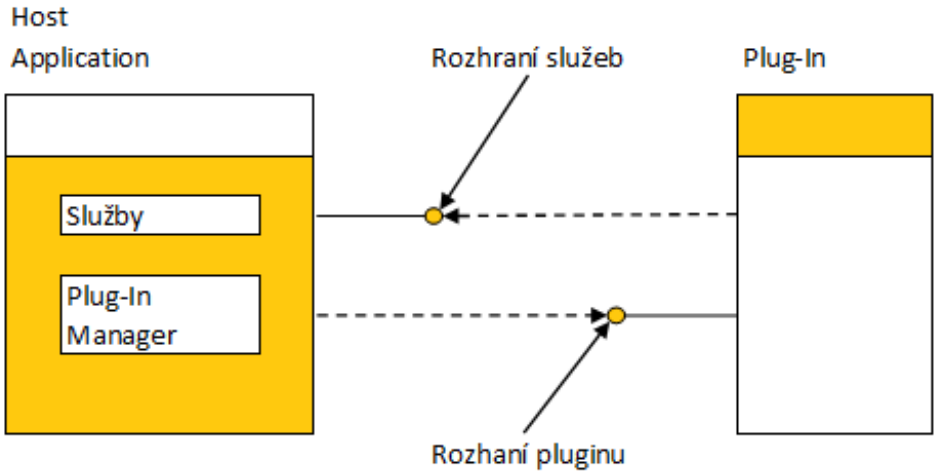
3 Plugin

Plugin neboli zásuvný modul je software, který nepracuje samostatně, ale jako doplňkový modul jiného programu a rozšiřuje tak jeho funkčnost. Plugin obvykle využívá připraveného rozhraní aplikace (API). Řada programů nabízí vývojářům možnost využití jejich API a tak i možnost rozšířit funkčnost tohoto programu [6].

3.1 Mechanismus pluginu

Hostitelské aplikace (Host Application) poskytují služby (Services), které může plugin využívat (viz Obrázek 2). Služby jsou uvedeny včetně způsobu registrace pluginu v hostitelské aplikaci a typu protokolu používaného pro vzájemnou výměnu dat.

Hostitelské aplikace fungují nezávisle na pluginech. Umožňují tak jednotlivým uživatelům dynamickou aktualizaci a přidávání různých pluginů. To vše bez nutnosti provádět změny na straně hostitelské aplikace [6].



Obrázek 2: Ukázka rozhraní pluginu

3.2 Plugin v Eclipse

Jak již bylo řečeno dříve, Eclipse IDE umožňuje rozšířit funkčnost svého vývojového prostředí pomocí pluginů. Takto je možné do Eclipse přidat nové funkce jako je například editor nebo menu [6].

Každý plugin může definovat „*rozšiřující body*“ (extension-points). Tyto body určují možnost rozšiřitelnosti pluginu. Toto rozšíření může být např. podpora nápovědy [6].

V případě Eclipse jsou pluginy bundly komponentového modelu OSGi (viz kapitola 2.2.1). Z toho vyplývá, že je plugin reprezentován standardním *JAR* souborem, obsahujícím všechny přeložené *CLASS* soubory, obrázky, konfigurační soubory a ostatní potřebné soubory. Nejdůležitějším souborem je *MANIFEST.MF*. Zde jsou uloženy základní metainformace o archivu, jako je jeho verze a název hlavního spouštěcího souboru. Velkou výhodou *JAR* souboru je nezávislost na platformě [6].

3.2.1 Model pluginu

Plugin je v Eclipse nejmenší komponentou, která poskytuje určitý druh služeb uvnitř pracovního prostředí. Jedná se o objekt, který může být nakonfigurován do systému v době jeho nasazení. Běhové prostředí (runtime) Eclipse poskytuje infrastrukturu pro aktivaci a provoz řady pluginů. Uvnitř běžící instance Eclipse je plugin umístěn v instanci třídy `plug-in runtime` nebo *`plug-in`*. Tyto třídy nabízí konfiguraci a podporu pro všechny instance pluginu. Dále musí rozšiřovat `org.eclipse.core.runtime.Plugin`, což je abstraktní třída poskytující obecné metody pro správu pluginů [6].

Uvnitř instalace Eclipse se nachází adresář plugins. Zde jsou uloženy jednotlivé pluginy. Každý plugin je umístěn ve svém adresáři a obsahuje soubor *plugin.xml*. Tento soubor obsahuje informace o aktivaci pluginu, které využívá běhové prostředí [6].

3.2.2 Tvorba pluginu

Pro vytváření pluginu do prostředí Eclipse je primárně určena verze *Eclipse for RCP and RAP Developers*. Není však problém vytvářet pluginy i ve většině ostatních verzí Eclipse. K tomu je ale zapotřebí speciální plugin [7].

Samotný plugin je vlastně jen zdrojový kód v jazyce Java, který využívá rozšíření a může tedy provádět různé akce ovlivňující prostředí Eclipse. Konkrétně lze například vložit `org.eclipse.ui.menus` do rozšíření našeho pluginu a tak přidat novou položku do hlavního menu Eclipse IDE [7].

Pro vytvoření nového pluginu je vhodné použít průvodce, který vede uživatele krok za krokem. Je zde umožněn výběr z celé řady základních vzorů, jako je například vytvoření pohledu. Vývojář může vytvořený kód libovolně rozšiřovat. Přímo v průvodci je umožněno přidat existující rozšiřující body [7].

3.2.3 Aktivace pluginu

Před samotnou aktivací je nutné nejprve plugin nasadit. To provedeme nakopírováním souborů, které představují plugin, do složky pluginů. Ta je umístěna přímo v instalaci Eclipse. Teprve poté může být tento plugin aktivován běhovým prostředím Eclipse. Aktivací pluginu rozumíme načtení jeho běhového prostředí a vytvoření instance. Během aktivace či deaktivace pluginu je prováděno speciální zpracování (např. alokace zdrojů). Tato zpracování jsou zpravidla nutná jen u složitějších a náročnějších pluginů. Pro jednoduché pluginy není zapotřebí žádné speciální zpracování pro aktivaci či deaktivaci. Proto nemusí návrháři pluginu uvádět žádnou konkrétní třídu. V tomto případě automaticky poskytuje běhové prostředí Eclipse výchozí třídě pluginu její instanci [7].

4 Java Swing

Swing je knihovna uživatelských prvků na platformě Java. Rozšiřuje původní knihovnu pro uživatelské rozhraní, která se nazývala AWT. Jeden ze základních rozdílů je v tom, že zatímco AWT tvořila pouze mezivrstvu mezi programem a nativními komponentami uživatelského rozhraní daného operačního systému, Swing využívá vlastní knihovnu komponent a komponenty operačního systému pro svou činnost nevyužívá. Výhodou je fakt, že aplikace vypadá všude stejně a chová se všude stejně. Nevýhodou je to, že se aplikace liší od ostatních aplikací v daném systému. Jako další nevýhodu uvedeme vyšší nároky na strojový čas procesoru. Aplikace ve Swingu se tedy může jevit jako pomalá a to zejména při spuštění na starším počítači [8].

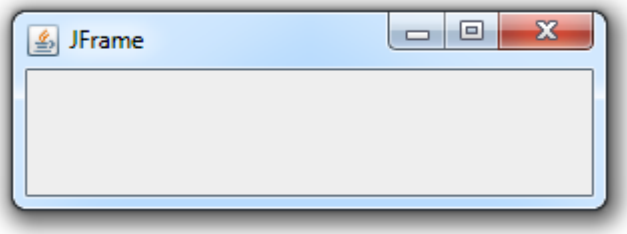
4.1 Základní Java Swing komponenty

V této kapitole jsou popsány základní grafické komponenty knihovny Java Swing, které se běžně vyskytují v programech s grafickým uživatelským rozhraním.

4.1.1 JFrame

Třída: `javax.swing.JFrame`

Standardní aplikační okno (viz Obrázek 3). Je využito jako prostor pro uložení dalších komponent, které chceme v okně zobrazit. `JFrame` poskytuje titulkový pruh a tlačítka pro minimalizaci, maximalizaci a uzavření komponenty. Vznik instance `JFrame` lze zajistit například vytvořením potomka třídy `JFrame`. Nově vytvořené okno je defaultně nastaveno jako neviditelné [9].

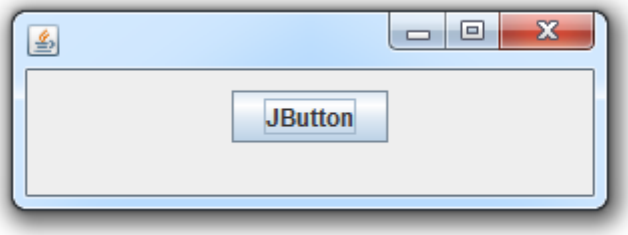


Obrázek 3: Ukázka komponenty JFrame

4.1.2 JButton

Třída: `javax.swing.JButton`

`JButton` je základní tlačítko (viz Obrázek 4). Jedná se o jednu z nejpoužívanějších Swingových komponent, která slouží zpravidla ke spuštění nějaké akce [10].

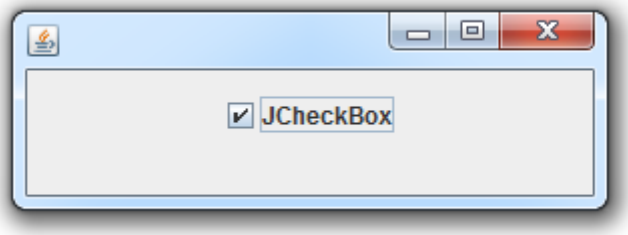


Obrázek 4: Ukázka komponenty JButton

4.1.3 JCheckBox – zaškrtávací pole

Třída: `javax.swing.JCheckBox`

Představuje přepínač mezi stavy zapnuto a vypnuto (viz Obrázek 5). Tato komponenta obsahuje zaškrtávací pole a návěští s popiskem. Pokud bude vloženo více těchto komponent, budou na sobě nezávislé. Tím je myšleno to, že mohou být všechny zaškrtnuty / odškrtnuty v libovolné kombinaci. Každý nově vložený `JCheckBox` je defaultně nastaven jako odškrtnutý [10].



Obrázek 5: Ukázka komponety JCheckBox

4.1.4 JComboBox – rozbalovací nabídka

Třída: `javax.swing.JComboBox`

Tento prvek představuje rozbalovací nabídku (viz Obrázek 6). Viditelná je vždy pouze jedna řádka. Po kliknutí na šipku se seznam rozbalí a umožní zvolit některou z položek seznamu. Komponenta je tedy vhodná pro výběr z většího množství možností. Položky `JComboBoxu` mohou být editovatelné a lze je dynamicky přidávat a ubírat [11].

Obrázek 6: Ukázka komponenty JComboBox

4.1.5 JLabel – popisek

Třída: `javax.swing.JLabel`

Tato komponenta slouží především k popisu funkce jiné komponenty (viz Obrázek 7). `JLabel` je jedním z nejjednodušších Swingových komponent. Může zobrazovat text, ikonu případně oboje zároveň. Text však bude vždy překrývat ikonu. Pokud není `JLabelu` nastaven žádný obsah, je tato komponenta neviditelná. Text zobrazený v `JLabelu` není možné editovat a je standardně zarovnán směrem vlevo [12].

Obrázek 7: Ukázka komponenty JLabel

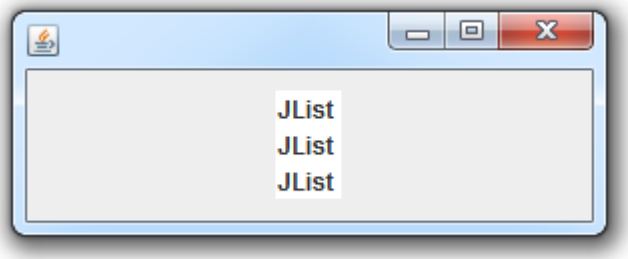
4.1.6 JList – seznam

Třída: `javax.swing.JList`

`JList` je víceřádková komponenta, která klasicky slouží k zobrazení více prvků v seznamu (viz Obrázek 8). Na rozdíl od `JComboBoxu` umožňuje trvalé zobrazení celého seznamu. Pokud je délka seznamu větší než délka komponenty nebo šířka položky větší než šířka komponenty, můžeme pro listování využít `JScrollBar` (popsán v kapitole 4.1.9) [13].

Uživatelský výběr ze seznamu je zajištěn pomocí modelu (objekt implementující interface `ListSelectionModel`). Výběr ze seznamu je možné provádět třemi způsoby [13]:

- Výběr jedné položky
- Výběr souvislého intervalu více položek
- Výběr nesouvislého intervalu více položek



Obrázek 8: Ukázka komponenty JList

4.1.7 JPanel – kontejner

Třída: `javax.swing.JPanel`

`JPanel` je nejjednodušším typem kontejnerové komponenty. Zajišťuje organizaci jednotlivých komponent, které jsou v něm umístěny. Rozmístění komponent je určeno pomocí layout manageru (defaultně je použit `FlowLayout`). Tuto komponentu lze využít i jako oblast pro kreslení grafiky [14].

4.1.8 JRadioButton – přepínač

Třída: `javax.swing.JRadioButton`

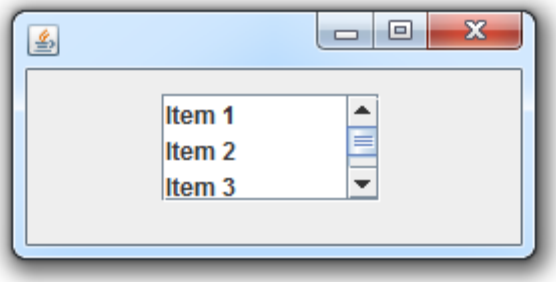
Představuje přepínač mezi stavy zapnuto a vypnuto (viz Obrázek 9). Tato komponenta obsahuje zaškrtávací pole a návěští s popiskem. `JRadioButtony` mohou být vkládány samostatně nebo sdružovány do skupin pomocí komponenty `ButtonGroup`. Pokud bude samostatně vloženo více `JRadioButtonů`, mohou být vybírány nezávisle na sobě. V případě skupiny `JRadioButtonů`, jsou na sobě komponenty závislé. Tím je myšleno to, že v jednom okamžiku může být vybráno pouze jedno tlačítko. Každý nově vložený `JRadioButton` je defaultně nastaven jako odškrtnutý [10].

Obrázek 9: Ukázka komponenty JRadioButton

4.1.9 JScrollPane – posuvník

Třída: `javax.swing.JScrollPane`

Každá komponenta nebo kontejner může být umístěn v `JScrollPanelu` (viz Obrázek 10). Pokud bude například šířka položky v `JListu` větší než šířka `JListu`, může být vložen `JScrollPanel` a tím bude umožněno zobrazení celé položky `JListu`. `JScrollPanely` lze nastavit jako vertikální, horizontální nebo oboje zároveň [15].

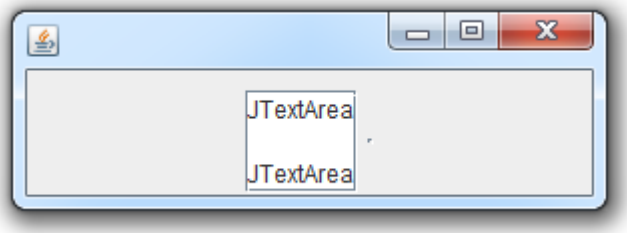


Obrázek 10: Ukázka komponenty JScrollPane

4.1.10 JTextArea – víceřádkové editační políčko

Třída: `javax.swing.JTextArea`

Komponenta představuje víceřádkové editační políčko, do kterého mohou být zadávány údaje z klávesnice (viz Obrázek 11). U této komponenty je možné nadefinovat šířku jedné řádky i počet zobrazených řádků. Pokud bude zapsáno více řádků, než je počet zobrazených, lze tento problém vyřešit pomocí `JScrollPanelu` nebo kurzorových šipek [16].



Obrázek 11: Ukázka komponenty JTextArea

4.1.11 JTextField – jednořádkové editační políčko

Třída: `javax.swing.JTextField`

Komponenta představuje jednořádkové editační políčko, do kterého mohou být zadávány údaje z klávesnice (viz Obrázek 12). Zadaný text může být delší, než šířka `JTextFieldu`. Ukončení editace vstupních dat je potvrzeno stisknutím klávesy Enter. U této komponenty je možné nadefinovat velikost zobrazeného textu [17].

Obrázek 12: Ukázka komponenty JTextField

4.2 Layout managery

Knihovna Swing, je tvořena řadou komponent. Komponenty, které mohou obsahovat jiné komponenty, se jmenují kontejnery (nejčastěji využíváme kontejner s názvem `JPanel` a `JFrame`, viz výše) [18].

Každý kontejner využívá právě jednu instanci třídy, která bývá označována jako správce rozvržení (layout manager). Uvnitř této třídy je obsažena logika pro rozmisťování komponent v kontejneru. Ovlivňuje tedy ukotvení a změnu velikostí jednotlivých komponent dle případných změn velikosti nadřazeného kontejneru [18].

Správců rozvržení existuje ve standardní knihovně několik. Jejich rozdíl je dán především poměrem mezi flexibilitou a složitostí použití. Vhodně zvoleným správcem rozvržení a vkládáním jednotlivých komponent do sebe, lze vytvořit téměř libovolný návrh uživatelského rozhraní [18].

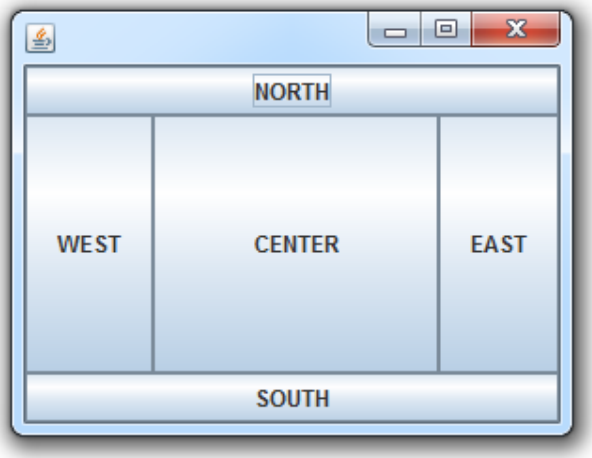
4.2.1 BorderLayout

Třída: `java.awt.BorderLayout`

Layout manager, který umožňuje vložit maximálně pět komponent. Jejich umístění nezáleží na pořadí vložení, ale je přesně dáno „světovou stranou“. Pro vložení komponenty lze použít konstanty NORTH, SOUTH, WEST, EAST a CENTER [19].

Vkládané komponenty jsou zvětšeny tak, že západní a východní komponentu roztáhne do maximální výšky, ale ponechá její šířku. Severní a jižní komponentu naopak roztáhne na maximální šířku, ale ponechá její výšku. Všechen zbylý prostor je vyplněn

středovou komponentou (viz Obrázek 13). Není-li v některé „světové straně“ umístěna komponenta, ostatní komponenty se roztáhnou do toho směru [19].



Obrázek 13: Ukázka rozmístění komponent při použití BorderLayoutu

4.2.2 BoxLayout

Třída: `javax.swing.BoxLayout`

Správce rozvržení, umísťující komponenty do řádku nebo sloupce. Komponenty jsou, je-li to nezbytně nutné, oříznuty, a ne přemístěny do dalšího řádku nebo sloupce. Souvisí s ním také třída `Box`, což je speciální kontejner s tímto layout managerem [20].

4.2.3 FlowLayout

Třída: `java.awt.FlowLayout`

Velice jednoduchý layout manager. Vkládané komponenty se postupně umísťují do řádky a nemění jejich velikosti. Pokud se již nevejdou do jedné řádky, pokračuje se na další řádce. Výška řádky je dána výškou nejvyšší komponenty v řádce [19].

4.2.4 GridLayout

Třída: `java.awt.GridLayout`

Layout manager, který vkládané komponenty rozmísťuje v pořadí vložení do předem určené mřížky. Na další řádek přechází až po vyplnění všech sloupců na stávající řádce [19].

Výšku a šířku komponenty nastavuje podle velikosti největší komponenty v řádku / sloupci pro všechny řádky a sloupce jednotně. Vždy je vyplněna celá plocha kontejneru [19].

4.2.5 GridBagLayout

Třída: `java.awt.GridBagLayout`

`GridBagLayout` využívá strukturu tabulky (mřížky), kde každá komponenta může obsadit i více řádek anebo sloupců. Další výhodou je možnost individuálně stanovit každé komponentě její velikost, okraje, roztažitelnost a umístění v rámci jí vyhrazeného prostoru [19].

4.2.6 SpringLayout

Třída: `javax.swing.SpringLayout`

Umožňuje zadání polohy komponenty na základě popisu vzdálenosti mezi jednotlivými komponentami. Vzdálenost komponenty může být určena od hrany kontejneru nebo od hrany další komponenty v kontejneru [21].

4.3 Události

Ke komponentě je možné přidat posluchače (tzv. listener) pro různé události. Posluchač je pro danou komponentu obvykle definován jako nějaké rozhraní. Pokud v komponentě proběhne daná událost (např. zmáčknutí tlačítka, vybrání položky atd.), komponenta všem příslušným posluchačům odešle zprávu o provedené akci. Události jsou doručeny pouze těm posluchačům událostí, kteří odpovídají typu události [22]. Základní přehled událostí a komponent, které sledují tyto události je zobrazen v Tabulce 1. Kompletní přehled je uveden v tutoriálu na internetových stránkách [22].

Tabulka 1:Základní přehled událostí, rozhraní a komponent sledující tyto události

Událost, rozhraní	**Komponenty sledující tyto události**

`ActionEvent`
`ActionListener`
	`JButton, JList, JTextField,`
	`JMenuItem` a její potomci, včetně tříd:
	`JMenu, JCheckBoxMenuItem` a `JPopupMenu`

`AdjustmentEvent` `AdjustmentListener`	`JScrollbar`

`ComponentEvent`
`ComponentListener`
	Třída `Component` a její potomci, včetně tříd:
	`JApplet, JButton, JDialog, JCanvas,`
	`JCheckbox, JComboBox, Container, JPanel,`
	`JScrollPane, Window, JFileChooser, JFrame,`
	`JLabel, JList, JScrollbar, JTextArea,`
	`JTextField`

`ContainerEvent`
`ContainerListener`
	Třída `Container` a její potomci, včetně tříd:
	`JDialog, JFrame, JPanel, JApplet,`
	`JScrollPane, Window, JFileChooser`

`FocusEvent`
`FocusListener`
	Třída `Component` a její potomci, včetně tříd:
	`JApplet, JButton, JCanvas, JCheckbox,`
	`JComboBox, Container, JPanel, JScrollPane,`
	`JDialog, JFileChooser, JFrame, JLabel,`
	`JList, JScrollbar, JTextArea, JTextField,`
	`Window`

`KeyEvent`
`KeyListener`
	Třída `Component` a její potomci, včetně tříd:
	`JApplet, JButton, JCanvas, JCheckbox,`
	`JComboBox, Container, JDialog, JPanel,`
	`JScrollPane, JFileChooser, JFrame, JLabel,`
	`JList, JScrollbar, JTextArea, JTextField,`
	`Window`

`MouseEvent`
`MouseListener`
	Třída `Component` a její potomci, včetně tříd:
	`JApplet, JButton, JCanvas, JCheckbox,`
	`JComboBox, Container, JDialog, JFrame,`
	`JPanel, JScrollPane, Window, JFileChooser,`
	`JLabel, JList, JScrollbar, JTextArea,`
	`JTextField`

`WindowEvent` `WindowListener`	`JDialog, JFileChooser, JFrame`
`ItemEvent` `ItemListener`	`JCheckBox, JCheckBoxMenuItem, JComboBox,` `JList`
`TextEvent` `TextListener`	Potomci `JTextComponent,` včetně tříd `JTextField` a `JTextArea`

5 Existující nástroje pro generování GUI

V současné době máme k dispozici celou řadu různých nástrojů pro generování GUI. V této kapitole se budeme zabývat jen nástroji, které pro generování GUI využívají knihovnu Java Swing.

5.1 Netbeans IDE 7.3

Tato verze Netbeans obsahuje opravdu propracovaný profesionální nástroj pro tvorbu GUI. Tento nástroj podporuje základní funkce, které mohou využívat především zkušení vývojáři. Jeho ovládání je intuitivní. Netbeans IDE 7.3 je vhodný jak pro pokročilé vývojáře, tak i pro ty méně zkušené.

Výhody:

- Uživateli je umožněno vkládání velkého množství komponent, které jsou přehledně členěny do skupin.
- Nástroj podporuje změnu a nastavení všech běžných layout manageru.
- Možnost řízení událostí.
- U každé komponenty je možné nastavit celou řadu vlastností.
- Všechna vstupní data nastavená uživatelem jsou dle očekávání validována.
- Nástroj je přímo součástí Netbeans IDE 7.3, není nutná žádná další instalace.

Nevýhody:

- Zdrojový kód se generuje automaticky, ale je zamčen proti přepsání. To může být pro zkušenější vývojáře nepříjemné.
- Kód je generován jen jedním způsobem, není možné generování nastavit

5.2 Java Gui Designer

Java Gui Designer je program pro operační systém Microsoft Windows (95/98/ME/NT/2000/XP/Vista). Testovaná byla verze s datem vydání 26.3.2013.

Java Gui Designer je opravdu velmi jednoduchý nástroj pro tvorbu GUI. Jeho ovládání je intuitivní. Nástroj obsahuje jen základní funkce pro návrh GUI. Java Gui Designer nelze porovnávat s ostatními uvedenými nástroji. V tomto případě se nejedná o profesionální nástroj. Tento program je určen především pro začínající vývojáře.

Výhody:

- Kopírování jednotlivých grafických komponent.
- Uložení rozpracovaného projektu.
- Zdrojový kód je chráněn proti přepsání. To může být vhodné pro začínající vývojáře.

Nevýhody:

- Uživateli je umožněno vkládání jen základních komponent.
- Nástroj nepodporuje změnu či editaci layout manageru.
- Jednotlivé názvy komponent není možné změnit.
- Hlavní okno tohoto nástroje nepodporuje možnost maximalizace.
- Smazání komponenty je možné pouze pomocí tlačítka na nástrojové liště.
- Kód je generován jen jedním způsobem, není možné generování nastavit.

5.3 Plugin do Eclipse – WindowBuilder Pro

Testována byla verze pro Eclipse 3.8 / 4.2 s instalací k datu 15.4.2013

WindowBuilder je velice propracovaný profesionální nástroj na tvorbu GUI. Tento nástroj obsahuje řadu různých funkcí, které ocení zejména zkušení vývojáři. Jeho ovládání je intuitivní. Plugin je vhodný jak pro pokročilé vývojáře, tak i pro ty méně zkušené.

Výhody:

- Uživateli je umožněno vkládání velkého množství komponent, které jsou přehledně členěny do skupin.
- Nástroj podporuje změnu a nastavení téměř všech existujících layout manageru.
- U každé komponenty je možné nastavit celou řadu vlastností.
- Všechna vstupní data nastavená uživatelem jsou dle očekávání validována.
- Zdrojový kód se generuje automaticky a není zamčen proti přepsání.
- Možnost řízení událostí.
- Možnost rychlé výměny komponenty za jiný typ.
- Tvorba výsledných objektů pomocí návrhového vzoru *Factory*.
- Změny provedené ve zdrojovém kódu jsou okamžitě přeneseny i do části designu.

Nevýhody:

- Celkově pomalejší běh a vykreslování nástroje.
- Kód je generován jen jedním způsobem, není možné generování nastavit.

5.4 Zhodnocení

Oba testované profesionální nástroje umožňují programátorům využívat různé doplňkové funkce. Ty mohou výrazně ulehčit práci vývojáře. Zároveň poskytují uživatelsky příjemné a intuitivní prostředí. Nicméně oba nástroje generují zdrojový kód jen jedním způsobem, což nemusí všem programátorům vyhovovat. Z toho plyne potřeba vytvořit nový, vlastní plugin.

6 Analýza

V této kapitole je popsána analýza pluginu pro Eclipse, který byl v rámci této práce vyvíjen.

6.1 Specifikace požadavků

Vyvíjený plugin pro Eclipse by měl splňovat požadavky, které jsou popsány v následujících kapitolách.

6.1.1 Požadavky na generování kódu

Zdrojový kód by mělo být možné vygenerovat individuálním rozmístěním aktuálně využitých komponent. Zároveň by mělo být možné nadefinovat vzorovou šablonu, ve které by bylo jednoznačně určeno, která část zdrojového kódu, pro které komponenty se bude v daném místě generovat.

6.1.2 Požadavky na umísťování komponent

Konfigurovatelnost by měla vypadat co nejvíce obecně. Mělo by být možné přesně určit umístění, kde se bude generovat která část kódu (například část pro vytvoření nebo nastavení tlačítka, volání metody, atd.).

6.1.3 Obsluha událostí

Mělo by být možné vybrat si způsob obsluhy událostí. Pokud uživatel vybere pojmenované vnitřní třídy nebo akce, mělo by být možné dát více prvkům obsluhu jednou třídou.

6.1.4 Metody a konstruktory

Mělo by být možné vložit do zdrojového kódu konstruktory a metody včetně možnosti nadefinování vstupních parametrů a případného návratového typu.

6.1.5 Vkládané komponenty

Mělo by být možné vložit všechny následně uvedené komponenty a určit jak se budou generovat. Dále by mělo být možné nastavit případnou obsluhu událostí.

- *standardně užívané komponenty:* `JButton`, `JTextField`, `JLabel`, `JCheckbox`, `JRadioButton` (včetně možnosti vytvořit skupinu), `JComboBox` (včetně možnosti vyplnění položek), `JTextArea`, `JList`
- *další komponenty:* komponenty pro vytvoření menu a nástrojové lišty (`JToolBar`, `JMenuBar`, `JMenu`, atd.).

6.2 Případy užití

6.2.1 Diagram případů užití

Diagram případů užití je uveden v Příloze D.1.

6.2.2 Popisy případů užití

Popisy případu užití pro diagram užití, který je uveden v Příloze D.1.

- *Spravovat panely* - Umožňuje nastavit vlastnosti a layout managery pro panely v návrhu GUI.
- *Spravovat objekty v návrhu GUI* - Uživatel může vkládat nebo mazat objekty v návrhu GUI.
- *Zvolit způsob obsluhy událostí pro komponenty v návrhu* – Umožňuje zvolit komponenty nebo skupiny komponent a určit způsob obsluhy událostí.
- *Editovat metody* - Umožňuje vytvořit nebo editovat metody a konstruktory, které budou použity ve zdrojovém kódu.
- *Uspořádat prvky zdrojového kódu nebo šablony* - Uživatel uspořádá pořadí generování prvků zdrojového kódu pomocí metody *Drag and Drop*.
- *Nastavit rozměry a pozici objektu v* GUI – Uživatel určí rozměry a pozici komponent pomocí metody *Drag and Drop* nebo zadáním hodnot do vlastností.
- *Nastavit vlastnosti objektu v GUI* – Umožní nastavit vlastnosti pro objekty v návrhu GUI.
- *Uložit vytvořenou šablonu* – Uživatel uloží navrženou šablonu pro další použití.
- *Použít šablonu* - Uživatel zvolí šablonu pro generování zdrojového kódu.
- *Vygenerovat zdrojový kód* - Umožňuje vygenerování zdrojového kódu pomocí individuálního rozmístění komponent nebo využitím šablony.

6.3 Grafické knihovny použité pro vykreslení GUI v pluginu

Jak již bylo zmíněno v kapitole 2, Eclipse používá vlastní grafický framework SWT. Pro vytváření GUI je tedy přednastavena tato knihovna. Pokud budeme chtít využít knihovny Swing (AWT), můžeme využít konverzi, kterou poskytuje samotný Eclipse.

Třída: `org.eclipse.swt.awt.SWT_AWT`

Můžeme se tedy libovolně rozhodnout, kterou knihovnu použijeme pro zobrazení komponent v pluginu. Následující kapitoly budou obsahovat základní informace o obou knihovnách.

6.3.1 Swing

Swing je součástí standardní Java knihovny a není tedy třeba přidávat další nativní knihovny. Má podporu řady oficiálních rozšíření (například OpenGL). Swing je platformě nezávislý a má výborně propracovaný tutoriál na internetových stránkách [23]. O veškerou správu paměti se stará garbage collector, což výrazně omezuje riziko vyčerpání systémových zdrojů. Jistým problémem může být to, že swing stále využívá pro svou funkčnost AWT komponenty [24].

6.3.2 SWT

SWT je vlastním grafickým frameworkem Eclipse IDE. Používá pro svou funkčnost nativní prvky a z toho plyne i nativní chování GUI. SWT vyžaduje pro svou funkčnost sadu knihoven pro každou podporovanou platformu. Právě využití nativního kódu a nativních zdrojů operačního systému může být problémem. V tomto případě je vyžadováno explicitní uvolňování objektů barva, štětec, font apod., čímž ztrácíme výhody, které nám poskytuje garbage collector a vzniká potenciální zdroj chyb [24].

6.3.3 Zhodnocení

Obě knihovny mají své výhody i nevýhody. Knihovna Swing má ale lépe propracované tutoriály, její komponenty jsou považovány za více flexibilní a snadněji použitelné. Pro zobrazení komponent v pluginu využijeme knihovny Java Swing.

6.4 Celkové provedení návrhářské části pluginu

Návrhářskou částí pluginu je myšlena část pluginu, ve které uživatel rozmisťuje komponenty do okna a nastavuje jejich vlastnosti.

6.4.1 Zobrazení hierarchie grafických komponent

První možností je využít prvek `JTree`. Ten může znázornit složení celého GUI. Kořenem stromu by byl `JFrame`. Uživatel by mohl vkládat další panely (instance třídy `JPanel` – uzly stromu) a následně i další komponenty (listy stromu).

Základním problémem tohoto návrhu je to, že uživatel nevidí vzhled celého okna. Jeho podobu si může pouze představit a to může být v případě komplikovanějšího návrhu okna dosti problematické.

6.4.2 Zobrazení výsledné podoby GUI

Bylo by tedy vhodnější zvolit takový návrh, ve kterém uživatel přímo uvidí výslednou podobu GUI. Uživatel bude moci vybírat libovolné komponenty a pomocí funkce tažení (*Drag and Drop*) je skládat přímo do návrhu okna. K vykreslení okna lze použít například `JPanel`.

Vzhledem k tomu, že toto řešení umožňuje podstatně lepší orientaci uživatele v návrhu okna, bude toto řešení v pluginu použito.

6.5 Grafické zobrazení GUI komponent

Zde se budeme zabývat způsoby, kterými je možno zobrazit komponenty v návrhářském okně.

6.5.1 Využití obrázku pro zobrazení komponent

Jednou z možností pro zobrazení GUI komponent, je uložit všechny prvky jako obrázky. Ty následně vkládat do okna s návrhem GUI. Jistým problémem by mohlo být nastavování barev jednotlivých prvků. To lze vyřešit omezením množství povolených barev. Pro každou variantu by byl vytvořen odpovídající obrázek komponenty. Tento problém lze vyřešit také tak, že si uživatel bude moci barvu komponenty zvolit, po překladu vygenerovaného zdrojového kódu bude mít tlačítko odpovídající podobu, ale v návrhu se změna neprojeví. To však opět vede k nepřehlednosti.

6.5.2 Vykreslování komponent pomocí Graphics2D

Vhodnější variantou je kompletní vykreslení všech komponent například pomocí objektu *Graphics2D*, který poskytuje velké množství možností pro práci s grafickými primitivy. Po vykreslení komponenty sice nebude jejich vizuální podoba přesná, nicméně bude možné libovolně měnit všechny vlastnosti.

Toto řešení umožňuje lépe zobrazit skutečnou výslednou podobu GUI, a proto bude v pluginu použito.

6.6 Objekty GUI komponent

Jelikož bude celý plugin vznikat v jazyce Java, je vhodné jednotlivé komponenty reprezentovat pomocí objektů.

6.6.1 Jedna třída pro všechny grafické komponenty

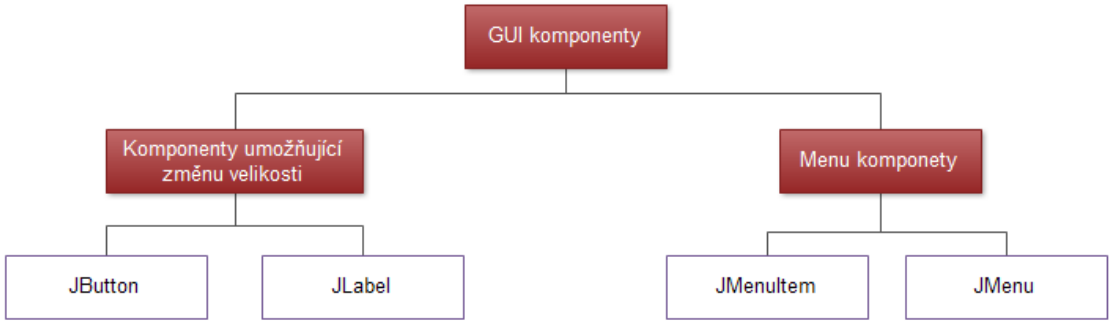
Lze vytvořit jednu univerzální třídu, která bude obsahovat metody pro nastavení všech různých vlastností pro všechny komponenty pluginu. Zde by bylo nutné rozlišovat aktuální typ komponenty například nastavením hodnoty specifického atributu.

Toto řešení však vede k tomu, že u mnoha grafických komponent bude velká část atributů jejich instance nevyužita.

6.6.2 Specifická třída pro každou grafickou komponentu

Lepší variantou je vytvoření specifické třídy pro každou komponentu. Můžeme také využít dědičnost a nadefinovat společné abstraktní metody. Ty lze ve třídách konkrétních komponent upravit. I když se jedná o lepší řešení, zůstává problém s vytvářením metod pro nastavování všech vlastností všech komponent pluginu.

Tento problém lze vyřešit vhodným použitím dědičnosti. Můžeme tedy vytvořit stromovou strukturu, ve které budeme potomkům předávat jen určité metody, a tím se tento problém alespoň částečně odstraní. Ukázka struktury je na Obrázku 14.



Obrázek 14: Návrh struktury GUI komponent

6.7 Uložení struktury návrhu GUI v paměti

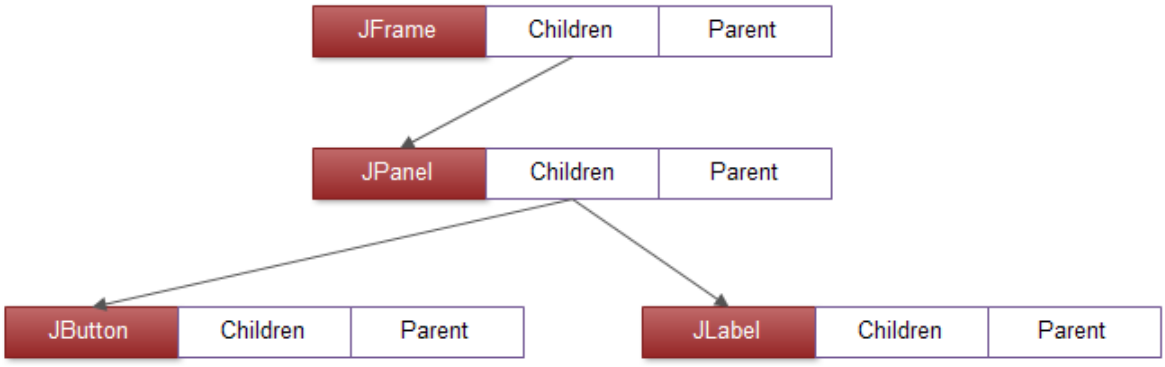
Zde se budeme zabývat způsoby, kterými je možné navrhnout strukturu pro uchování a další práci s návrhem GUI. Budeme předpokládat, že je každá komponenta reprezentována pomocí objektu. V každém prvku bude nutné udržovat seznam potomků a předka této komponenty. To především z důvodu nutnosti procházení celé struktury.

6.7.1 Ukládání grafických komponent do seznamu

Jelikož nikdy nevíme, kolik komponent bude chtít uživatel vložit, můžeme objekty ukládat například do ArrayListu případně do spojového seznamu. Při procházení struktury by bylo nutné neustále prohledávat všechny prvky a testovat, zda se jedná o aktuálně hledaného potomka / předka.

6.7.2 Ukládání grafických komponent do stromové struktury

Lepším řešením je ukládat prvky do stromové struktury. Kořen bude tvořit objekt `JFrame`. Nový prvek bude vždy uložen přímo do seznamu potomků (viz Obrázek 15). Toto zjednodušuje práci například při smazání celého panelu nebo při vykreslení GUI.



Obrázek 15: Uložení struktury návrhu GUI v paměti

6.8 Konfigurovatelnost generovaného zdrojového kódu

Největší možné konfigurovatelnosti dosáhneme tak, že umožníme uživatelům rozhodnout o libovolném rozmístění jednotlivých komponent jejich návrhu ve zdrojovém kódu.

Pro individuální generování zdrojového kódu je možné předložit uživateli všechny komponenty, umožnit vytvoření vlastních metod, přiřadit akce ke komponentám, atd. Zbývá jen nechat uživatele libovolně přeskládat pořadí jejich vytvoření, nastavení, volání atd. To lze zajistit například přiřazením čísel určující pořadí generování. Nebo efektivněji pomocí funkce *Drag and Drop* (tuto možnost využijeme).

Zároveň je dobré umožnit generování zdrojového kódu pomocí nějaké šablony, kterou si může uživatel definovat, uložit a pak opakovaně používat při návrhu GUI.

6.8.1 Využití regulárních výrazů

Šablonu lze nadefinovat pomocí regulárního výrazu. Ten můžeme načítat ze souboru nebo ho můžeme ručně zadat na vstup generátoru. Zde ovšem hrozí poměrně velká pravděpodobnost chyby. Samotný návrh podoby regulárního výrazu by byl poměrně problematický. Mimo toho je toto řešení uživatelsky nepřívětivé.

6.8.2 Definování šablony pomocí tažení (Drag and Drop)

Lepším řešením bude umožnit uživateli vkládat jednotlivé části šablony pomocí tažení (*Drag and Drop*) a takto vytvořenou šablonu následně uložit do souboru. Tento soubor je pak možné načíst a použít pro vygenerování zdrojového kódu.

Pro vytváření šablony bude tedy použit podobný postup, jako při návrhu konkrétního GUI. Zde ovšem nebudou řazeny konkrétní prvky ale všechny možné typy komponent. Toto řešení je uživatelsky přívětivější a bude použito.

6.9 Objekty pro sestavení výsledné šablony nebo zdrojového kódu

Jak již bylo uvedeno, každá komponenta v návrhu výsledného zdrojového kódu bude tvořena objektem. Zdrojový kód pro vložení každého prvku lze rozdělit na více částí (viz kapitola 6.10). To znamená, že pro každou komponentu z navrženého GUI, by mělo být možné určit místo ve kterém bude vytvořena, nastavena, atd.

6.9.1 Vytváření kopií objektů

To lze zajistit například tak, že budou ke každému objektu z GUI vytvořeny kopie. Těm bude individuálně určena požadovaná část kódu, kterou budou generovat. Nicméně nesmíme zapomenout na to, že zdrojový kód je tvořen také pomocí metod, tříd, atd. Zde by vznikal problém s jejich reprezentací.

6.9.2 Využití objektu spojujícího konkrétní prvek s generovaným kódem

Je tedy vhodné vytvořit nějaký neutrální objekt, pomocí kterého by bylo možno pracovat se všemi prvky tvořící zdrojový kód. Tento objekt, bude obsahovat zdrojovou komponentu (prvek GUI, metodu, atd.) a informaci o požadované části kódu, která bude pro komponentu generována (dle jazyka popsaného v kapitole 6.10.). Ukázka návrhu objektu je na Obrázku 16. Toto řešení bude v pluginu využito.

Obrázek 16: Návrh Objektu pro sestavení výsledné šablony nebo zdrojového kódu

6.10 Rozpoznání požadované části zdrojového kódu komponenty

Zdrojový kód nutný k vložení GUI komponenty lze rozdělit na jednotlivé části. Tím je myšlena část vytvoření, nastavení a vložení prvku. Ve skutečnosti můžeme podobné rozdělení najít i u metod, přiřazení akcí atd. Můžeme tedy navrhnout vlastní jazyk. Tím bude jednoznačně popsáno, která část zdrojového kódu komponenty je nyní požadována. (např. `Create and Set` – vytvoř a nastav prvek).

Takto bude snadno a zároveň jednoznačně definováno, kterou část zdrojového kódu chceme pro konkrétní komponentu vygenerovat.

6.11 Získání zdrojového kódu komponent

Pro každý objekt je možné generovat různé části zdrojového kódu. Jejich výběr lze provádět pomocí jazyka zmíněného v kapitole 6.10.

6.11.1 Speciální třída pro generování zdrojového kódu

Budeme předpokládat, že každý prvek v návrhu je reprezentován objektem. Je tedy možné si připravit třídu jen pro generování zdrojového kódu. Zde by každý objekt měl k dispozici vlastní metody pro získání žádaného zdrojového kódu. Na základě typu GUI objektu by se volala konkrétní metoda.

6.11.2 Metoda pro generování zdrojového kódu ve třídě GUI komponenty

Poněkud jednodušší variantou je umístění metody pro generování zdrojového kódu přímo do třídy GUI komponenty. Tyto metody je pochopitelně nutné individuálně upravit. Jelikož mají tyto komponenty jisté části zdrojového kódu podobné, je možné společný kód umístit do jedné metody a tu následně volat.

Toto řešení umožňuje lepší propojení grafických komponent s jejich generovaným zdrojovým kódem a bude použito v pluginu.

6.12 Rozdělení prvků dle jejich typu

Jelikož šablony předpokládají hromadnou práci s více prvky jednoho typu (např. `JButtons`), je nutné tyto komponenty v aktuálním návrhu nějak rozlišit.

6.12.1 Opakované prohledávání grafických komponent v návrhu GUI

První možností je opakované prohledávání všech vložených komponent v návrhu GUI. Pokaždé by proběhl test shody typu komponenty s aktuálně hledanou skupinou komponent. To je ale velmi nepraktické. Navíc by vznikal problém při určování prvků skupiny *Others*. To je skupina, ve které se bude pracovat s prvky, pro které není v šabloně inicializace.

6.12.2 Nalezení všech grafických komponent jedním průchodem

Mnohem lepším řešením je jedním průchodem šablony nalézt všechny typy inicializovaných skupin komponent. Následně rozdělit všechny prvky z návrhu GUI do nalezených skupin. Prvek, pro který není nalezena skupina, se automaticky stává členem skupiny *Others*. Pro jejich rozdělení je možné využít např. hash tabulku kombinovanou s ArrayListem. Tak bude možné podle jednoho klíče opakovaně získat všechny prvky skupiny.

Toto řešení je efektivnější z hlediska časové náročnosti výpočtu a bude tedy využito v pluginu.

6.13 Správné pořadí vkládání prvků do panelů

Při generování pomocí šablony bude nutné zajistit správné pořadí vkládání prvků do `JPanelů`. Pokud přidáme komponenty ve špatném pořadí, bude podoba vygenerovaného okna jiná (dle využitého layout manageru). Jelikož předem neznáme počet využitých panelů ani počet komponent, není prakticky možné nijak odděleně nadefinovat vkládání jednotlivých komponent.

Bude tedy dobré nechat uživatele rozhodnout o místě, ve kterém se hromadně vloží všechny prvky. Správné pořadí vkládání bude zajištěno pomocí algoritmu.

6.14 Kontrola přeložitelnosti generovaného zdrojového kódu

Kontrola přeložitelnosti zdrojového kódu je sama o sobě velice náročná a mimo toho překračuje rozsah této práce. Z tohoto důvodu nebude tato část implementována.

Předpokládáme, že s tímto pluginem bude pracovat programátor, který má zkušenosti s použitím knihovny Java Swing pro tvorbu GUI. Tento uživatel bude mít k dispozici uživatelský návod.

7 Realizační část

Práce je vytvořena v programovacím jazyce Java. Jako vývojové prostředí pro tvorbu pluginu je zvoleno Eclipse IDE for Java EE Developers. Grafické uživatelské rozhraní v celém pluginu je navrženo ve vývojovém prostředí Netbeans IDE 7.3.

V této kapitole se budeme zabývat samotnou implementací pluginu. Nejprve bude popsána celková struktura aplikace. Následovat bude detailní popis datové a aplikační části pluginu, přičemž aplikační část bude rozdělena na další dvě části. První částí je *navržení nového GUI* a druhá část se týká *vygenerování zdrojového kódu*.

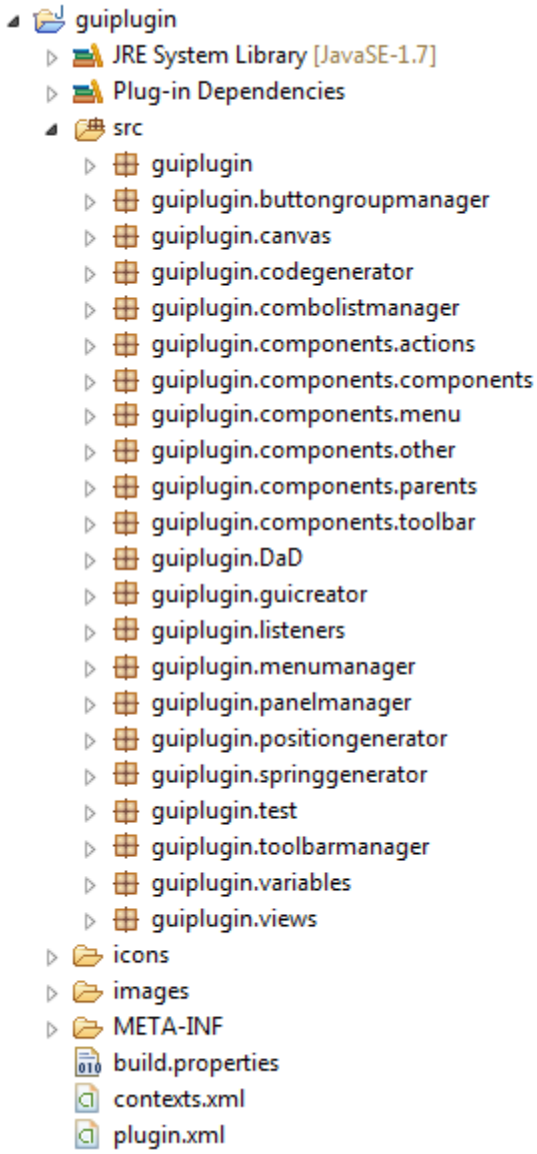
7.1 Struktura aplikace

Aplikace se skládá z několika balíků, které obsahují jednotlivé funkce pluginu. Další součástí jsou konfigurační soubory a ostatní složky (obsahují ikony, obrázky, atd.). Struktura pluginu je zobrazena na Obrázku 17, UML diagram tříd je uveden v Příloze D.2.

Základní struktura vytvořeného pluginu vypadá takto:

- Balík `guiplugin.buttongroupmanager` obsahuje třídu, která se stará o vytvoření a editaci skupin, do nichž mohou být vkládány `JRadioButtony` a `JCheckBoxy`.
- Balík `guiplugin.canvas` obsahuje třídu, která zajišťuje správné vykreslování aktuálního návrhu GUI. Umožňuje také smazání či změnu velikosti komponent.
- Balík `guiplugin.codegenerator` je složen ze tříd, které jsou nutné k navržení, vygenerování a uložení zdrojového kódu nebo vzorové šablony. Dále obsahuje třídy pro navržení metod a akcí.
- Balík `guiplugin.combolistmanager` obsahuje třídy, které se starají o možnost editace jednotlivých prvků uložených v komponentech typu `JList` a `JComboBox`.
- Balík `guiplugin.components.actions` je složen ze tříd reprezentující jednotlivé akce, které je možné ke komponentám přiřadit.
- Balík `guiplugin.components.components` obsahuje třídy popisující všechny standardní prvky, které je možné vložit do GUI (`JButton`, `JComboBox`, atd.) Nejsou zde uloženy komponenty pro menu nástrojovou lištu.
- Balík `guiplugin.components.menu` je složen ze tříd, které reprezentují prvky vkládané do menu.
- Balík `guiplugin.components.other` obsahuje třídy, které popisují všechny ostatní využívané prvky (nové metody a třídy, metoda Main, atd.)

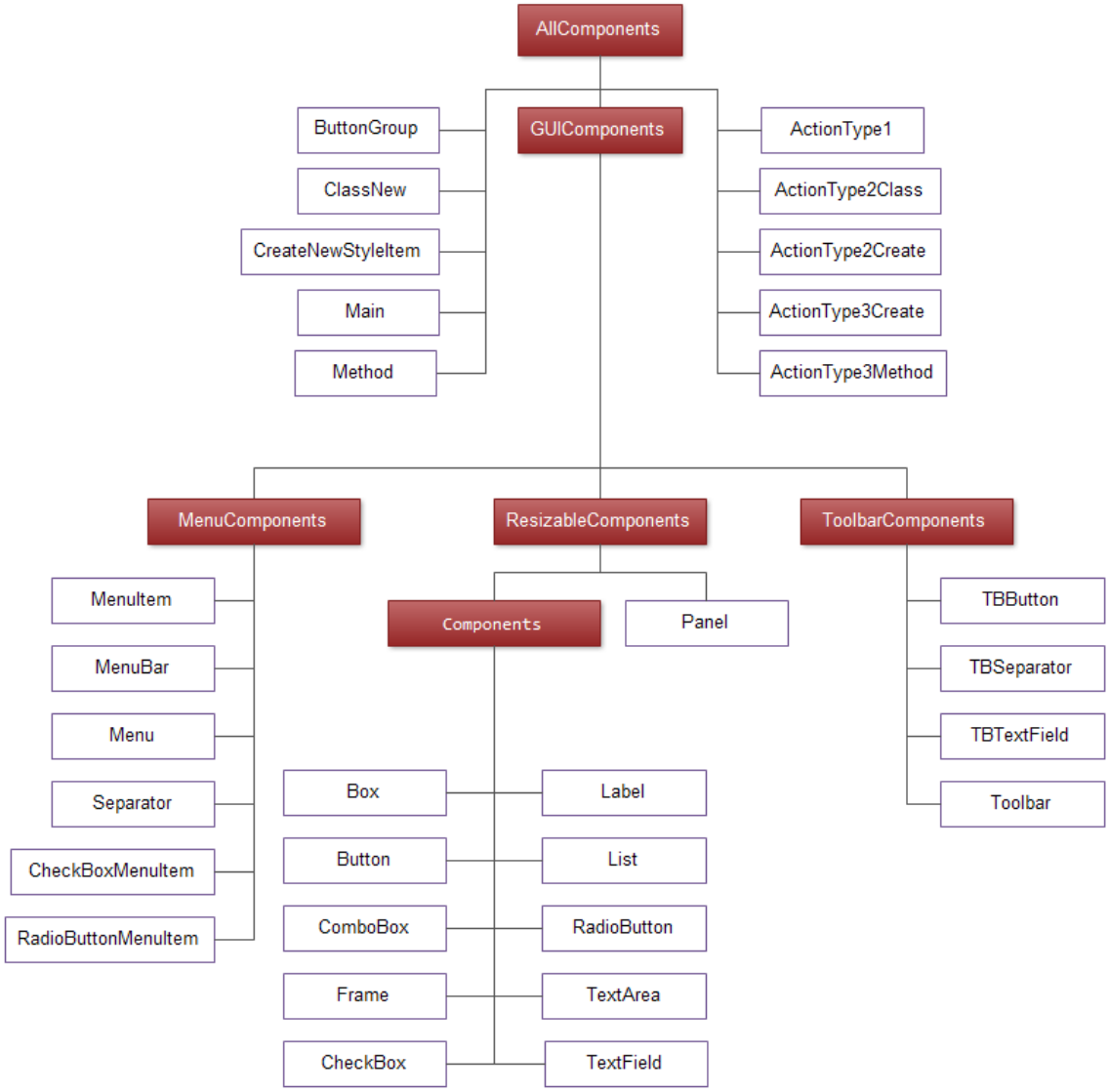
- Balík `guiplugin.components.parents` je složen ze tříd od kterých mohou dědit všechny další komponenty (třída od které dědí všechny prvky menu atd.).
- Balík `guiplugin.components.toolbar` obsahuje třídy, které popisují prvky vkládané do `JToolBaru`.
- Balík `guiplugin.DaD` obsahuje pouze jednu třídu. Ta se stará o správné přemístění prvků GUI mezi `JPanely`. Zároveň zajišťuje i možnost prohození dvou prvků uvnitř jednoho panelu.
- Balík `guiplugin.guicreator` je složen ze tříd, které zajišťují vytvoření hlavního view s návrhářem GUI.
- Balík `guiplugin.listeners` obsahuje třídy a posluchače pro hlavní okno návrháře GUI. Zajistí vložení nových prvků GUI nebo nastavení hodnot jejich vlastností.
- Balík `guiplugin.menumanager` je složen ze tříd, které umožňují uživateli vytvořit vlastní menu a nastavit prvkům libovolné vlastnosti.
- Balík `guiplugin.panelmanager` obsahuje třídy, které umožňují vytvářet a nastavovat `JPanely`. Dále umožňuje přiřazení a konfiguraci layout managerů.
- Balík `guiplugin.positiongenerator` obsahuje pouze jednu třídu, zde dochází k přepočtu pozic a rozměrů jednotlivých prvků v závislosti na zvoleném layout manageru.
- Balík `guiplugin.springgenerator` obsahuje pouze jednu třídu, která umožňuje uživateli přesně zvolit pozici jednotlivých komponent při použití `SpringLayoutu`.
- Balík `guiplugin.test` obsahuje pouze jednu třídu, která se stará o validaci vstupních dat. Ověřuje, zda bylo zadáno číslo, zda se nevyskytuje jiná komponenta se stejným jménem, atd.
- Balík `guiplugin.toolbarmanager` je složen ze tříd, které umožňují uživateli vytvořit vlastní toolbar a nastavit jeho prvkům libovolné vlastnosti.
- Balík `guiplugin.variables` je složen ze tříd, které obsahují proměnné a konstanty nutné pro běh pluginu. Jsou zde uloženy názvy layout managerů, strom reprezentující všechny prvky GUI, konstanty pro vytvoření šablon, atd.
- Balík `guiplugin.views` obsahuje pouze jednu třídu pro vytvoření view.
- Balík `guiplugin` obsahuje pouze třídu `Activátor.java`, která řídí běh pluginu. Stará se také o přiřazování potřebných rozšíření a jsou zde uvedeny i závislosti pluginu na ostatních pluginech.
- Soubor `plugin.xml` je souborem manifestu, který popisuje plugin. Obsahuje informace pro běhové prostředí aplikace, kde je uvedeno vše co je potřeba k zavedení a aktivaci pluginu.
- Soubor `MANIFEST.MF` je konfigurační soubor obsahující konfigurační informace OSGi.



Obrázek 17: Struktura pluginu

7.2 Datová část pluginu

Datová část pluginu je tvořena objekty. Za pomoci dědičnosti je vytvořena stromová struktura, která je zobrazena na Obrázku 18. Objekty tedy dědí jen ty metody, které jsou pro jeho skupinu charakteristické. Takto je omezeno vytváření zbytečných metod popsaných v kapitole 6.6.



Obrázek 18:Struktura datové části pluginu

7.2.1 Objekty pro návrh GUI

Jde o všechny potomky třídy `GUIComponents`. Všem prvkům je přiřazena určitá priorita, kterou je charakterizováno jejich zařazení. Priorita je dána především dle pomyslné vrstvy, do které je komponenta vkládána (Frame = 1, Panel = 2, atd.).

Jednotlivé objekty jsou ukládány do stromové struktury (viz kapitola 6.7), ta je uložena ve třídě `Variables`. Každý objekt obsahuje referenci na komponentu, ve kterém je vložen (rodič) a zároveň má seznam všech objektů, které jsou v něm vloženy (potomci).

Každý objekt obsahuje metody charakteristické pro danou skupinu. Tyto metody lze rozdělit na:

- *Identifikační* metody sloužící pro jednoznačné určení objektu. Zde mohou být metody pro vykreslení objektu, práci s objektem předka, práci s objekty potomků, atd.
- *Popisné* metody slouží k nastavení atributů popisující daný prvek. Zde mohou být metody pro nastavení popisku objektu, práci se jménem objektu, vygenerování zdrojového kódu, atd.
- *Speciální metody* pracují s daty, které jsou určeny jen pro tento objekt. Například metody pro práci s prvky uloženými uvnitř `JComboBoxu`.

7.3 Aplikační část pluginu – navržení nového GUI

Hlavní okno pluginu pro návrh GUI je vytvořeno třídou `SetMainGUI`, pochopitelně je nutné přiřadit posluchače pro jednotlivé komponenty tvořící GUI. K tomuto účelu slouží třídy `ListenerComponents` a `ListenerProperties`. Vstupní data jsou validována pomocí metod třídy `Test`.

7.3.1 Vykreslení navrženého GUI

Všechny objekty, které jsou vloženy do okna návrhu, obsahují vlastní metody pro své vykreslení. Jedná se o všechny potomky třídy `ResizableComponents`. Jako prostor pro kreslení je použit `JPanel`. O správné vykreslení celého GUI se stará třída `Canvas`. Před vykreslením může být voláno přegenerování pozic komponent, dle použitých layout managerů (třída `PositionGenerator`). Při vykreslování prvků by mohlo dojít k nechtěnému překrytí objektů. Správné pořadí je dáno dle priorit prvků:

a) Prvky s prioritou Frame
b) Prvky s prioritou Panel
c) Všechny ostatní prvky

Dále je vykresleno ohraničení aktuálních komponent a panelů. Závěrem je vykreslen obrys komponenty, nad kterou je v současné době ukazatel myši.

7.3.2 Vložení nové komponenty do návrhu

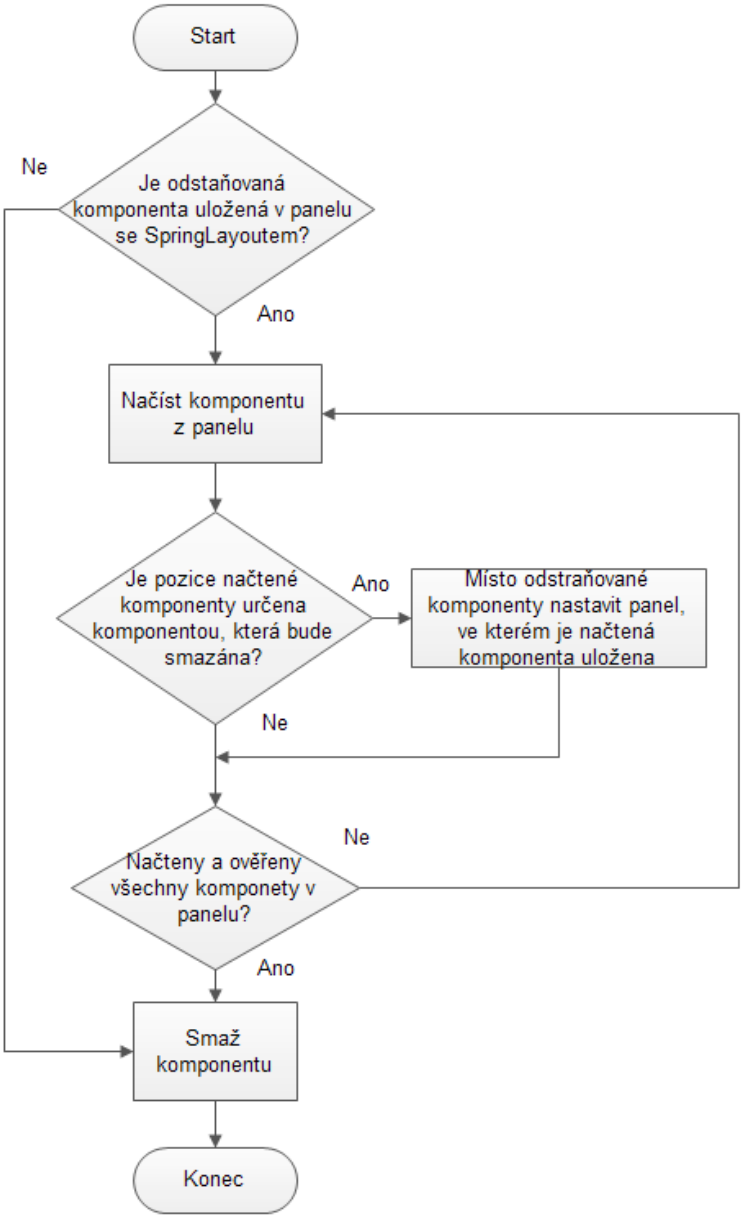
Před vložením nové komponenty musí být vybrán aktuální panel, do kterého bude nový prvek umístěn (viz kapitola 7.3.4). O vložení komponenty se stará třída `ListenerComponents`, která na základě výběru uživatele z nabídky komponent v `JListu` (třída `SetMainGUI`), vytvoří nový objekt. Ten vloží do seznamu potomků

aktuálního panelu. Dále je nutné zajistit zobrazení nové komponenty v okně s návrhem. To zařídí metody třídy `Canvas`, pro překreslení a přegenerování okna.

Dříve zmíněný `JList`, který slouží k výběru nově vkládané komponenty, využívá třídu `GUIListRender`. Ta dědí od třídy `DefaultListCellRenderer`. Je tedy využívána pro grafickou úpravu položek zobrazených v `JListu`.

7.3.3 Smazání komponenty

Před smazáním komponenty musí být zvolena aktuální komponenta, která bude smazána (viz kapitola 7.3.4). Smazání komponenty nastane po stisknutí pravého tlačítka myši. Postup smazání komponenty zobrazuje Obrázek 19.



Obrázek 19: Vývojový diagram zobrazující postup smazání komponenty

Informace o referenčních objektech, které slouží pro nadefinování pozice komponenty ve `SpringLayoutu` jsou uloženy uvnitř každého objektu grafického návrhu. O smazání komponenty se stará třída `Canvas`.

7.3.4 Aktuální panel a komponenta

Při navrhování nového GUI je nutné mít přesně určený panel a komponentu, které jsou v daný okamžik aktuální. Jen tak je možné vkládat komponenty, nastavovat vlastnosti, měnit velikost nebo přesouvat objekty. O určení aktuální komponenty a panelu řídí třída `Canvas`.

Jelikož je každá komponenta určena přesnou pozicí a rozměry, je možné na základě aktuální pozice ukazatele myši určit, zda je kurzor uvnitř objektu. Nejprve je určen aktuální panel.

Určení aktuálního panelu

- O aktuálním panelu rozhoduje průchod stromovou strukturou GUI návrhu. Pokud je zjištěno, že je kurzor uvnitř testovaného objektu a objekt má prioritu panelu (nebo frame), pokračuje další průchod pouze touto větví. Toto se opakuje do té doby, dokud jsou testovány prvky s výše uvedenou prioritou.
- Poslední nalezený prvek je nastaven jako aktuální panel.

Určení aktuální komponenty

- Je nutný průchod celou stromovou strukturou návrhu GUI a vždy ověřit, zda je kurzor uvnitř testovaného objektu.
- Pokud již byl nalezen prvek, v jehož objektu je umístěn kurzor myši, bude rozhodováno dle tohoto postupu:
    1) Komponenta s vyšší prioritou bude nastavena jako aktuální.
    2) Pokud jsou priority stejné, rozhoduje panel, ve kterém jsou komponenty uloženy. Komponenta, jejíž panel není stejný jako aktuální panel, bude nastavena jako aktuální.

Vyhledávání komponent začíná po stisknutí levého tlačítka myši. Aktuální komponenta i aktuální panel jsou uloženy ve třídě `Variables`.

7.3.5 Změna velikosti a pozice komponenty

Pro lepší uživatelskou přívětivost je zaveden objekt podobný aktuální komponentě (viz kapitola 7.3.4). Jeho vyhledávání probíhá stejným způsobem, ale je určován permanentně při pohybu ukazatele myši. Lze ho popsat jako objekt, nad kterým je v daný okamžik ukazatel myši. Tento objekt se využívá pro změnu pozice a velikosti komponenty.

Každý objekt v GUI obsahuje v každém svém rohu čtvercový prostor, který je také testován při posuzování, zda je kurzor myši uvnitř objektu. Tyto prostory jsou testovány dříve, než plocha celého objektu. Pokud je ukazatel myši uvnitř rohového prostoru, může uživatel stisknout levé tlačítko myši a tahem měnit velikost objektu. Nové rozměry komponenty jsou dopočteny z pozice kurzoru myši při stisknutí a uvolnění tlačítka. Tato hodnota je okamžitě ukládána do změněné komponenty. O změnu velikosti komponenty se stará třída `Canvas`.

Pokud není ukazatel myši uvnitř rohového prostoru komponenty, ale je uvnitř objektu, jde o změnu její pozice. Tuto správu obstarává také třída `Canvas`. Změna pozice je možná pouze v panelu, ve kterém je nastaven `SpringLayout`. Nicméně je možné komponenty přesouvat z panelu do panelu nebo zaměnit dvě komponenty. O to se stará třída `DaD`.

O přesunu komponenty mezi panely rozhoduje panel, nad kterým je v okamžiku uvolnění tlačítka ukazatel myši. Pokud je stejný jako panel, ve kterém je uložena přesouvaná komponenta, jde o přesun v rámci jednoho panelu. V tomto případě dojde pouze k záměně objektů v seznamu potomků tohoto panelu. V opačném případě je objekt ze zdrojového seznamu potomků vyjmut a je vložen na konec seznamu potomků cílového panelu.

Závěrem jsou volány metody třídy `Canvas` pro překreslení návrhu.

7.3.6 Generování pozic komponent v návrhu (simulace layout manageru)

Tuto práci obstarává třída `PositionGenerator`. Primárním úkolem této třídy je simulace činnosti layout managerů pro jednotlivé panely. Dochází tedy k postupnému průchodu všech panelů v návrhu. Dle použitého layout manageru jsou volány odpovídající metody, které nastavují pozice a případně i rozměry všech komponent v panelu. Pořadí generování pozic prvků je určeno jejich pořadím v seznamu potomků daného panelu. Při výpočtu pozic komponent jsou pochopitelně zohledněny i další nastavené vlastnosti panelu zadané uživatelem.

7.3.7 Správce panelů v návrhu

Třída `PanelManagerListeners` odpovídá za správné nastavení layout managerů pro panely aktuálního návrhu. Poté co uživatel zvolí typ layout manageru ve vytvořeném okně správce panelu (třída `PanelManagerGUI`, dojde k nastavení možných podtypů pro zvolený layout manager. Hodnoty, které si uživatel zvolí, se okamžitě ukládají do objektu panelu. Seznamy typů a podtypů panelů jsou uloženy ve třídě `Variables`. Pokud se uživatel rozhodne použít typ `BorderLayout` (layout manager který je složen z více částí), dojde ke smazání celého panelu a je nahrazen novými objekty. Pro všechny ostatní layout managery zůstanou vložené komponenty zachovány.

7.3.8 Tvorba menu

O vytvoření menu se starají třídy `MenuManagerGUI` a `MenuManagerListeners`. Menu není ukládáno do stejné stromové struktury jako ostatní prvky GUI. Vytváří vlastní stromovou strukturu, ale je založena na stejném principu. O vložení prvku do struktury se stará třída `MenuManagerListeners`, ta na základě zvolené komponenty v GUI (třída `MenuManagerGUI`) určí prvek, který má být vložený a také prvek, do kterého má být vložen (dle vybraného listu v `JTree`). Nově vytvořený objekt je pak vložen na konec seznamu potomků cílové komponenty. Struktura menu je uložena ve třídě `Variables`.

7.3.9 Tvorba toolbaru (nástrojové lišty)

O vytvoření toolbaru (nástrojové lišty) se starají třídy `ToolbarManagerGUI` a `ToolbarManagerListeners`. Toolbar není ukládán do stejné stromové struktury jako ostatní prvky GUI. Vytváří vlastní stromovou strukturu ale je založena na stejném principu jako ostatní prvky GUI. O vložení prvku do struktury se stará třída `ToolbarManagerListeners`, ta na základě zvolené komponenty v GUI (třída `ToolbarManagerGUI`) určí prvek, který má být vložen a také prvek, do kterého má být vložen (dle vybraného prvku v `JListu`). Nově vytvořený objekt je pak vložen na konec seznamu potomků cílové komponenty. Struktura toolbaru je uložena ve třídě `Variables`.

7.3.10 Nastavení pozice komponenty při využití SpringLayoutu

O nastavení konkrétní pozice aktuální komponenty se stará třída `GUIsetSpringLayoutComponents`. V této třídě jsou vždy uživateli nabídnuty referenční komponenty, uložené ve stejném panelu jako aktuální komponenta (včetně panelu). Defaultně je nastaven dopočet hodnoty mezery mezi komponenty pomocí třídy `GapGenerátor`. Uživatel může pochopitelně hodnotu mezery změnit. Dojde k přepočtu nové vzdálenosti na základě zvolených hran komponent a zadané mezery mezi nimi. Nové souřadnice jsou do aktuálního prvku uloženy. Zvolené referenční komponenty, které slouží k nadefinování pozice aktuální komponenty, jsou uloženy v objektu aktuální komponenty.

7.3.11 Ostatní

V této kapitole budou popsány jednodušší části pluginu.

- *Správu ButtonGroup* obstarává třída `GUIButtonGroupManager`. Ta slouží k zobrazení grafického okna pro práci s `ButtonGroups`. Umožňuje přidání nebo editaci stávajících skupin (uloženy ve třídě `Variables`). Aktuální

komponentě je možno přiřadit jednu konkrétní skupinu. Název skupiny, do které komponenta patří, je uložen v jejím objektu.

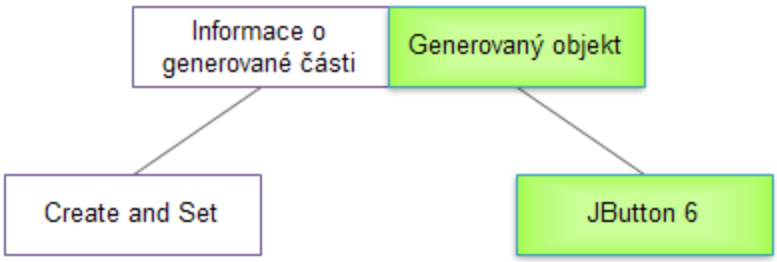
- *Naplnění komponent JList a JComboBox* obstarává třída `GUIComboList`. Třída slouží k zobrazení grafického okna pro správu položek zobrazených v aktuálním `JComboBoxu` a `JListu`. Tato třída je společná pro oba typy komponent. Každá komponenta má seznam prvků umístěn uvnitř svého objektu.

7.4 Aplikační část pluginu – Generování zdrojového kódu

Generování zdrojového kódu je umožněno dvěma způsoby. Buď se jedná o generování individuální, nebo o generování za pomoci šablony. Obě metody využívají společné třídy.

Ani v jednom případě není výsledný zdrojový kód nijak validován. Umístění všech částí zdrojového kódu je tedy jen na uživatelově uvážení. Tím je myšleno to, že pokud uživatel bude chtít například vytvořit tlačítko mimo hlavní třídu, bude v tomto místě skutečně vygenerováno. Kontrola přeložitelnosti zdrojového kódu nebyla součástí zadání. Její vytvoření by vyžadovalo poměrně značnou práci navíc.

Při vytváření zdrojového kódu se využívají objekty třídy `GeneratedItem`. Tento objekt je složen ze dvou částí. První částí je jednoznačně stanoveno, která část zdrojového kódu komponenty je nyní požadována (viz kapitola 7.4.4). Druhou částí je objekt, který je potomkem třídy `AllComponents` (viz Obrázek 18). Složení objektu využívaného při generování zdrojového kódu je na Obrázku 20.



Obrázek 20: Ukázka složení objektu GeneratedItem

7.4.1 Úvodní okno generátoru

Třída `CodeGeneratorMain` vytváří úvodní grafické uživatelské rozhraní, které dále umožňuje vytvoření nové šablony nebo volbu způsobu generování zdrojového kódu.

Při zvolení možnosti generování dle šablony, dojde k vytvoření dialogového okna pro načtení souboru. Načtená data jsou následně předána třídě zajišťující generování zdrojového kódu dle šablony (`CodeGeneratorGenerateTemplate`). Složka, ve které je uložena šablona, bude použita jako výchozí pro následující dialogové okno.

Tím je okno pro uložení souboru. Zde je přednastaven název nového souboru na základě názvu třídy, který byl získán ze šablony při generování souboru.

V ostatních případech je volán konstruktor třídy `CodeGeneratorHowToGUI`.

7.4.2 Výběr způsobu a úrovně vytvářených komponent

Výběr způsobu a úrovně vytvářených komponent je zajištěn třídou `CodeGenerator-HowToGUI`. Ta nejprve vytvoří grafické uživatelské rozhraní, které umožní uživateli zvolit způsob vytváření a nastavení komponent (vytvoř, nastav, vytvoř a nastav).

V případě individuálního generování je umožněna i volba úrovně zobrazených komponent (frame, panel, componet). Pro vytváření šablony je defaultně nastavena úroveň component. Tato úroveň definuje mez, do které si může uživatel navolit umístění jednotlivých částí komponent při generování. Komponenty pod touto úrovní budou generovány zároveň s prvním umístěným předkem.

7.4.3 Návrh zdrojového kódu nebo šablony

O sestavení výsledného zdrojového kódu se starají třídy `CodeGeneratorGUI` a `CodeGeneratorListener`.

Třída `CodeGeneratorGUI` vytváří hlavní grafické uživatelské rozhraní, pro generování zdrojového kódu. Okno je tvořeno mimo jiné několika `JListy`. Ty obsahují objekty, které lze vložit od výsledného zdrojového kódu. Všem `JListům` v tomto okně je nastaven transferhandler (třída `ListTransferHandler` která je potomkem třídy `TransferHandler`). S jeho pomocí je umožněno rozšířit funkci *Drag and Drop* (přesun objektu) také mezi dva `JListy`.

7.4.4 Jazyk používaný generátorem

Pro účely generátoru bylo nutné navrhnout speciální jazyk, kterým bude přesně specifikována požadovaná část zdrojového kódu. Každá část kódu, která může být vygenerována, má své specifické zastoupení v jazyce. Metody všech objektů, které vrací vygenerovaný kód, mají přístup k tomuto jazyku a na základě uvedené informace vrací požadovanou část. Tímto jazykem je tedy přesně stanoveno, která část kódu, pro který požadavek bude vrácena. Vytvořený jazyk je uložen ve třídě `VariablesGenerator`.

7.4.5 Vytvoření základních objektů pro generování zdrojového kódu

Pro načtení objektů, které budou vkládány do výsledného zdrojového kódu, slouží primárně třída `LoadComponents`. Na základě zvoleného typu vytváření a nastavení

komponent (viz kapitola 7.4.2), jsou vytvořeny nové objekty třídy `GenratedItem` a následně vloženy do konkrétních `JListů` třídy `CodeGeneratorGUI`. Struktura prvků je zobrazena na Obrázku 20.

V případě individuálního generování zdrojového kódu jsou zpracovány všechny komponenty, které uživatel použil při vytváření návrhu GUI. Tyto objekty jsou potomky třídy `AllComponents` (viz Obrázek 18). V tomto případě je možné nastavit maximální úroveň načtených komponent (viz kapitola 7.4.2). Komponenty jsou filtrovány dle hodnoty priority.

V případě vytváření nové šablony se nezpracovávají objekty z návrhu GUI, ale vytváří se nové objekty třídy `CreateNewStyleItem` Nový objekt se vytváří pro každou skupinu známých komponent (buttons, panels, atd.). Seznam skupin je uložen ve třídě `Variables`.

Pro každý zpracovávaný prvek (například tlačítko) je vytvořena sada nových objektů (objekty třídy `GeneratedItem`), které reprezentují jednotlivé části kódu nutné pro vytvoření a nastavení tohoto prvku (`Create, Set`, atd.). Tento jazyk je popsán v kapitole 7.4.4.

7.4.6 Vytvoření a editace metody

O vytvoření objektu nové metody se starají třídy `CodeGeneratorNew-MethodListener` (slouží k nastavení posluchačů pro prvky z návrhu GUI) a `CodeGeneratorNewMethodGUI` (vytváří grafické uživatelské rozhraní).

V návrhu nové metody je umožněno uživateli zvolit si návratové hodnoty a parametry. Zde jsou použity komponenty získané z návrhu GUI. V případě návratových hodnot je pochopitelně nutné filtrovat typy použitých komponent, aby nedocházelo k zobrazení duplicit.

Pokud je metoda navrhována při vytváření šablony, mohou být jako parametry využity všechny skupiny podporovaných komponent (buttons, panels, atd.). Seznam těchto skupin je uložen ve třídě `VariablesGenerator`. V tomto případě může být vytvořena pouze metoda bez návratové hodnoty nebo metoda návratového typu `void`.

Jde-li o změnu existující metody, jsou data předána v konstruktoru třídy `CodeGeneratorNewMethodGUI` a hodnoty jsou přednastaveny do GUI.

Objekt metody je zakládán ve třídě `CodeGeneratorNewMethodListener`. Zde dojde k vytvoření a nastavení objektu třídy `Method` (reprezentující vytvořenou metodu). Vytvořený objekt spolu s popisem požadované části zdrojového kódu tvoří nový objekt třídy `GeneratedItem`. Jazyk popisující požadované části zdrojového kódu je popsán v kapitole 7.4.4.

Budou vytvořeny dva objekty nutné pro práci s novou metodou. První je objekt pro volání a druhý je pro nastavení metody. Nově vytvořené objekty jsou vloženy do `JListů` ve třídě `CodeGeneratorGUI`.

7.4.7 Vytvoření akce

O vytvoření objektu nové metody se starají třídy `CodeGeneratorNewActionGUI` a `CodeGeneratorNewActionListener`.

Třída `CodeGeneratorNewActionGUI` vytváří grafické uživatelské rozhraní, pro návrh nových akcí pro komponenty zdrojového kódu.

Pokud jsou akce generovány pro individuální generování zdrojového kódu, jsou použity komponenty z návrhu GUI. Tyto komponenty jsou filtrovány na základě priorit.

Pokud jsou akce generovány při vytváření šablony, mohou být vygenerovány pouze pro ty skupiny komponent, které umožňují přiřazení akcí (buttons, atd.). Seznam těchto skupin je uložen ve třídě `VariablesGenerator`.

Objekty akcí (balík `guiplugin.components.actions`) jsou vytvořeny ve třídě `CodeGeneratorNewMethodListener`. Zde dojde k vytvoření a nastavení objektů pro konkrétní akci (dle zvolené akce). Vytvořené objekty spolu s popisem požadované části zdrojového kódu tvoří nový objekt třídy `GeneratedItem`. Dojde k vytvoření jednoho nebo několika objektů nutných pro vytvoření zvolené akce (dle zvoleného typu). Jazyk popisující požadované části zdrojového kódu je popsán v kapitole 7.4.4. Nově vytvořené objekty jsou vloženy do `JListů` ve třídě `CodeGeneratorGUI`.

7.4.8 Individuální generování zdrojového kódu

O individuální generování zdrojového kódu se stará třída `CodeGenerator-GeneratorActualComponents`. Jednotlivé položky zdrojového kódu jsou dány obsahem `JListu` ve třídě `CodeGeneratorGUI`. Každá položka je objektem třídy `GeneratedItem` (viz Obrázek 20).

Prochází se tedy všechny prvky v `JListu` a pro každý generovaný objekt je zavolána metoda pro generování zdrojového kódu. Jako parametr je předána informace o generované části. Generovaný objekt na základě požadavku rozpozná žádanou část zdrojového kódu a ta je vrácena.

Zároveň je nutné udržovat informace o tom, zda je generovaný prvek uvnitř metody či uvnitř třídy a uzavírat tak jednotlivé bloky kódu. To je zaručeno opět pomocí navrženého jazyka. Generátor rozpoznává jednotlivé prvky a dle předem daného seznamu rozhoduje o vložení složených uzavíracích závorek.

Generátor umožňuje zvolit *úroveň komponent pro generování* (viz kapitola 7.4.2). To znamená, že pokud je zvolena vyšší úroveň než component, dochází k vygenerování zdrojového kódu potomků současně s posledním vloženým prvkem určeným nastavenou úrovní. To znamená, že dojde k průchodu všech potomků tohoto objektu a každý bude volat vlastní metodu pro generování zdrojového zdrojového kódu. Jako parametr je předána informace o generované části. Generovaný objekt na základě požadavku rozpozná žádanou část zdrojového kódu a ta je vrácena generátoru. Jako příklad uvedeme, že při úrovni panels se budou s panelem generovat i tlačítka v něm umístěná.

Závěrem je vytvořena instance třídy `SaveFile` a zavolána metoda pro uložení výstupu v podobě souboru. Ta vždy nabídne uživateli doporučený název souboru na základě jména třídy ve vygenerovaném kódu. Složka, která je využita pro uložení dat je nadále použita jako výchozí pro všechny operace načítání a ukládání dat do souboru.

7.4.9 Generování zdrojového kódu pomocí šablony

Třída `CodeGeneratorGenerateTemplate` slouží k vygenerování zdrojového kódu dle připravené šablony.

Nejprve je vytvořeno dialogové okno pro načtení dat ze šablony. Složka obsahující šablonu bude využita jako výchozí, pro všechny další operace načtení nebo uložení dat ze souboru.

Na každém řádku šablony je pomocí jazyka generátoru (viz kapitola 7.4.4) uveden požadavek na vygenerovaný kód a skupina prvků, pro který bude kód generován. Je zavedena také skupina prvků s názvem *Others*. Tu tvoří komponenty, které nejsou nikde v šabloně inicializovány.

Je tedy nutné zjistit, které skupiny prvků jsou inicializovány. Ostatní patří do skupiny *Others*. To je zajištěno průchodem všech řádek šablony a hledání klíčových slov pro inicializaci (*Create* ,*Create and Set*, atd. - jsou dány jazykem generátoru). Do nalezených skupin jsou rozděleny prvky z návrhu GUI. Komponenty, jejichž skupina nebyla inicializována, se stávají členy skupiny *Others*.

Nyní je nutné znovu projít celou šablonu. Na každém řádku hledat shodu s prvky jazyka generátoru a tak určit část kódu, která má být pro danou skupinu prvků vygenerována. Seznam všech známých požadavků je předem dán používaným jazykem generátoru (viz kapitola 7.4.4). Pochopitelně je ověřován i název skupiny a nalezené prvky z návrhu GUI.

Pokud je požadavek rozeznán, je pro všechny prvky skupiny zavolána metoda pro generování zdrojového kódu. Jako parametr je předána informace o požadavku na generovanou část. Generovaný objekt na základě požadavku rozpozná žádanou část zdrojového kódu a ta je vrácena. Testování shody požadavků je uspořádáno tak, aby nemohlo dojít k záměně. Například (*Create and Set* bude testováno před *Create*).

V případě požadavků na vytvoření zdrojového kódu akce, metody Main, a třídy jsou vytvářeny nové objekty, které tyto komponenty reprezentují. Ty následně volají metodu pro vytvoření zdrojového kódu

Pro požadavky na generování metod není vytvářen nový objekt metody, ale tento řádek šablony je převeden na zdrojový kód. Dále je pouze doplněna skupina prvků, kterých se požadavek týká.

Pokud dojde k problémům při rozpoznávání řádku, je celý řádek přeskočen. Zároveň je nutné udržovat informace o tom, zda je generovaný prvek uvnitř metody či uvnitř třídy a uzavírat tak jednotlivé bloky kódu. To je zaručeno opět pomocí navrženého jazyka. Generátor rozpoznává jednotlivé prvky a dle předem daného seznamu rozhoduje o vložení složených uzavíracích závorek.

Závěrem je vytvořena instance třídy `SaveFile` a je zavolána metoda pro uložení výstupu v podobě souboru. Ta vždy nabídne uživateli doporučený název souboru na základě jména třídy ve vygenerovaném kódu. Složka, která je využita pro uložení dat je nadále použita jako výchozí pro všechny operace načítání a ukládání dat do souboru.

8 Testování

Pro ověření správné funkčnosti tohoto pluginu, bylo nutné vytvořit několik sérií testů. Každý test se skládal z návrhu GUI a následného vygenerování zdrojového kódu. Pro vytvoření zdrojového kódu bylo využito jak generování za pomoci šablony, tak i individuální rozmístění komponent. Každý výstupní soubor byl přeložen a výsledné okno bylo porovnáno s oknem navrženým v pluginu.

Po úspěšném otestování byl plugin předán několika testerům. Každý tester je programátor, který používá knihovny Java Swing pro návrh GUI. Jejich úkolem bylo seznámit se s uživatelskou dokumentací a následně vytvořit a vygenerovat dva návrhy. Jeden návrh okna byl libovolný a druhý byl dán dle předem připraveného vzoru. Tento vzor byl stejný jako ukázka výstupu pluginu, která je zobrazena v Příloze C. Hodnocení bylo na stupnici 1 – 5 (jako ve škole). Výsledky hodnocení jsou uvedeny v Tabulce 2.

Tabulka 2: Hodnocení pluginu testery

	Tester č. 1	**Tester č. 2**	**Tester č. 3**	**Tester č. 4**	**Průměr**
Práce s komponentou při návrhu GUI (vlastnosti, změna pozice…)	2	2	1	2	**1,75**
Správce panelů	2	2	2	2	**2**
Správce menu	2	3	3	2	**2,5**
Správce toolbaru	1	3	2	2	**2**
Chování komponent při změně layout manageru	2	2	2	2	**2**
Celkové hodnocení návrhářské části	2	2	2	2	**2**
Návrh pomocí šablony – použitelnost, konfigurovatelnost	2	2	2	2	**2**
Individuální generování – použitelnost, konfigurovatelnost	2	2	2	1	**1,75**
Vytváření metod	2	3	2	2	**2,25**
Vytváření akcí	2	1	2	2	**1,75**
Celkové hodnocení části pro generování zdrojového kódu	2	2	2	2	**2**
Celkové hodnocení pluginu	2	2	2	2	**2**

8.1 Hodnocení výsledků v tabulce

První hodnocenou částí je *práce s komponentou při návrhu GUI*. Zde byla nejčastější kritika směřovaná na horší grafické zpracování komponent. Jelikož jsou všechny prvky návrhu kresleny a nejsou použity žádné obrázky, byla tato kritika očekávána. Ostatní hodnocení této části bylo poměrně kladné.

Další testovanou částí byl *správce panelů*. Většina testerů uvedla, že mají problém s orientací ve stromové struktuře panelů. Tento problém nastával zejména v případě složitějšího návrhu GUI.

Třetím úkolem bylo otestování *správce menu*. Horší hodnocení bylo odůvodněno především tím, že zde není umožněno přesouvat již jednou vytvořené komponenty. Při návrhu bylo chybně předpokládáno, že uživatel nebude chtít jednou vytvořené menu dále upravovat.

Další testovanou částí byl *správce toolbaru*. Zde bylo nejčastěji vytýkáno to, že je umožněno vkládat toolbary jen do dvou stran v okně. Při návrhu pluginu byla tato možnost zvolena z toho důvodu, že se jedná o nejčastěji používaná místa pro nástrojové lišty.

Poslední hodnocenou částí z oblasti návrhu, bylo *chování komponent při změnách layout manageru*. Každý tester uvedl jako výtku to, že se komponenty nerozloží úplně přesně tak, jako při následném překladu zdrojového kódu. Tato výtka byla předem očekávána. Aktuální pozice a rozměry komponent jsou vypočteny pomocí pluginu. Z toho plyne, že podoba okna po překladu zdrojového kódu a okna s vytvořeným návrhem nikdy nebude přesná.

Při testování generátoru zdrojového kódu byl nejprve hodnocen *návrh pomocí šablony* a následně bylo hodnoceno *individuální rozmístění komponent*. V obou případech byla uvedena jako nečastější výtka to, že se při vytvoření většího množství akcí, vícekrát vkládají stejně pojmenované třídy pro obsluhu akcí. Tento problém může vyřešit sám uživatel již při vytvoření akce. Pokud má jistotu, že nově vytvořenou třídu nepotřebuje, může jí ihned smazat.

Další testovanou částí bylo *vytváření metod*. Nejčastějším problémem uvedeným v této části, byl menší výběr moditifikátorů u parametrů metody. Jelikož není dodělána kontrola přeložitelnosti zdrojového kódu, nebyla tato část při implementaci příliš řešena.

Poslední testovanou částí je *vytváření akcí*. Zde bylo jako nejčastější problém uvedeno to, že je možné při individuálním generování zdrojového kódu přidávat akce jen po jedné komponentě. Tento postup byl zvolen z předpokladu, že si uživatel raději navolí obsluhu pro každý prvek zvlášť.

8.2 Konkrétní příklad

Na tomto jednoduchém příkladu bude demonstrováno generování zdrojového kódu za použití šablony. Vytvoření šablony je popsáno v uživatelské dokumentaci (viz Příloha A, kapitola A.4.1). Níže uvedená šablona byla použita pro generování ukázkového příkladu. Barvy jednotlivých částí šablony odpovídají barvám ve zdrojovém kódu, který je uveden v Příloze B.

Okno, které vznikne po přeložení výše zmíněného zdrojového kódu a okno z návrhu GUI je možno porovnat na Obrázku 21 (okno z návrhu GUI je v horní části obrázku).

Další porovnání navrženého a vygenerovaného okna pro složitější případ je uveden v Příloze C.

Ukázka vytvořené šablony:

```
//vytvoření hlavní třídy
CLASS  myClass
     //Vytvoření konstruktoru třídy
    CONSTRUCTOR public  myClass()
          //vytvoření JFrame
        CREATE  Frame
          //vytvoření všech JButtonů
        CREATE  Buttons
          //vytvoření všech JCheckboxů
        CREATE  CheckBoxes
          //vytvoření všech ostatních komponent
        CREATE  Others
          //volání konstruktoru pojmenované vnitřní třídy
        CREATE CLASS WITH ACTION  class_name action_name
          //nastavení všech ostatních prvků
        SET  Others
          //nastavení JFrame
        SET  Frame
          //nastavení všech JButtonů (s akcí)
        SET WITH ACTION Buttons with action action_name
          //nastavení všech JCheckBoxů (s akcí)
        SET WITH ACTION CheckBoxes with action action_name
          //sestavení GUI
        ADD ALL CREATED GUI COMPONENTS
          //volání pack()
        PACK  Frame
     //metoda Main
    METHOD  Main
          //volání konstuktoru
        CALL CONSTRUCTOR myClass()
//vytvoření pojmenované vnitřní třídy
CLASS WITH ACTION FOR COMPONENT   class_name
```

8.3 Porovnání navrženého a vygenerovaného okna

Budeme porovnávat navržené a vygenerované okno, které je zobrazeno na Obrázku 21. Z obrázku je patrné, že je v návrhu použit `BorderLayout`, který má v každé své části vložen `JPanel`. V dalších kapitolách budeme porovnávat jednotlivé části oken vzájemně mezi sebou.

Na první pohled vidíme, že si rozměry obou porovnávaných oken přesně neodpovídají. Tento fakt musíme při porovnávání zohlednit.

8.3.1 Porovnání severních částí oken

Jako layout manager pro severní `JPanel` je nastaven `FlowLayout` (levé zarovnání). Do tohoto panelu je vložen `JComboBox`, který má nastaven preferovaný rozměr.

Při porovnání obou oken je patrné, že rozměry a pozice vloženého `JComboBoxu` jsou v obou oknech přibližně stejné. Barva a rozměry `JPanelu` také souhlasí.

8.3.2 Porovnání západních částí oken

Jako layout manager pro západní `JPanel` je nastaven `GridLayout` (počet řádků: 2, počet sloupců: 0). Do tohoto panelu jsou vloženy dvě tlačítka (`JButton`).

Při porovnání obou oken je vidět, že vložené `JButtony` jsou do řádek umístěny stejným způsobem a jejich rozměry si také odpovídají. Barvy tlačítek rovněž souhlasí.

8.3.3 Porovnání centrálních částí oken

Jako layout manager pro centrální `JPanel` je nastaven `FlowLayout` (centrální zarovnání, vertikální mezera: 15, horizontální mezera: 15). Do tohoto panelu je vložen `JCheckBox` a `JRadioButton`.

Při porovnání obou oken je vidět, že komponenty jsou v panelech umístěny stejným způsobem. Po vzájemném porovnání dvou komponent stejného typu zjistíme, že i jejich rozměry jsou přibližně stejné. Vertikální a horizontální mezery v obou oknech také odpovídají. Barva a rozměry `JPanelu` rovněž souhlasí.

8.3.4 Porovnání východních částí oken

Jako layout manager pro východní `JPanel` je nastaven `BoxLayout` (`Y_AXIS`). Do tohoto panelu jsou vloženy dva `JLabely`. Oba mají nastaveno levé zarovnání.

Při porovnání obou oken je vidět, že se výška obou komponent lehce odlišuje, šířky si vzájemně odpovídají. Umístění komponent v panelu je stejné, barva a rozměry `JPanelu` také souhlasí.

8.3.5 Porovnání jižních částí oken

Jižní panel neobsahuje žádné další komponenty. Nebudeme tedy uvádět jeho další nastavení. Při porovnání obou oken je vidět, že si výška, šířka a barva panelu odpovídají.

8.3.6 Zhodnocení

Po porovnání jednotlivých částí oken můžeme říci, že se chování layout managerů použitých v návrhu velice podobá skutečnosti. Jediný drobný rozdíl vidíme ve výškách komponent při použití `BoxLayoutu`. Panely a komponenty jsou vkládány a nastavovány správně. Při porovnání grafického vzhledu komponent jsou vidět pochopitelné rozdíly. Celkově můžeme říci, že jsou si obě okna velice podobná.

Obrázek 21: Porovnání navrženého a vygenerovaného okna

9 Závěr

Úkolem této diplomové práce bylo navrhnout, implementovat a dále otestovat plugin, který má umožnit tvorbu GUI s konfigurovatelným generováním zdrojového kódu.

Pro úspěšné dokončení práce bylo nejprve nutné lépe se seznámit s vývojovým prostředím Eclipse a problematikou jeho pluginů. Dále bylo potřeba prozkoumat existující nástroje, které umožňují vytvořit vlastní návrh GUI. Teprve poté mohla začít implementace nového pluginu.

Při návrhu pluginu bylo podstatné navrhnout datovou vrstvu tak, aby bylo umožněno její další efektivní využití. Jako inspirace pro tvorbu prezenční vrstvy sloužilo vývojové prostředí Netbeans. Aplikační vrstva tohoto pluginu byla navržena tak, aby byla zajištěna přehlednost a jednoznačnost ze strany programátora.

V průběhu vývoje docházelo k nezbytnému průběžnému testování jednotlivých částí. Po dokončení implementace bylo nutné celý plugin řádně otestovat. Testování probíhalo dle různých scénářů pro menší i větší návrhy GUI.

Jistým nedostatkem této práce je to, že vznikající zdrojový kód není nijak validován. Tato funkčnost nebyla součástí původního zadání. Její vytvoření by bylo netriviální a zabralo by poměrně velké množství času. Je však námětem pro další rozšíření práce.

Na závěr lze konstatovat, že práce zcela splňuje zadání. Podařilo se efektivně navrhnout, implementovat a poté otestovat nový plugin. Ten je možné využívat ve více verzích Eclipse, které splňují požadavky uvedené v uživatelské příručce.

Seznam zkratek

API - *Application Programming Interface*

Označuje v informatice rozhraní pro programování aplikací. Jde o sbírku procedur, funkcí, tříd či protokolů nějaké knihovny (ale třeba i jiného programu nebo jádra operačního systému), které může programátor využívat.

AWT -*Abstract Windows Toolkit*

Knihovna grafických uživatelských prvků pro platformu Java.

EPL - *Eclipse Public License*

Open source licence pro software nadace Eclipse Foundation.

GUI - *Graphical User Interface*

Uživatelské rozhraní, které umožňuje ovládat počítač pomocí interaktivních grafických ovládacích prvků.

HTML - *HyperText Markup Language*

Značkovací jazyk pro hypertext. Je hlavním z jazyků pro vytváření stránek v systému World Wide Web, který umožňuje publikaci dokumentů na Internetu.

IBM - *International Business Machines Corporation*

Přední světová společnost v oboru informačních technologií. Mezi hlavní činnosti společnosti patří v současnosti výroba a prodej počítačového software a hardware a desítky služeb.

IDE - *Integrated Development Environment*

Software usnadňující práci programátorů, většinou zaměřen na jeden konkrétní programovací jazyk.

JVM – *Java Virtual Machine*

Sada počítačových programů a datových struktur, která využívá modul virtuálního stroje ke spuštění dalších počítačových programů a skriptů vytvořených v jazyce Java.

OSGi - *Open Services Gateway initiative*

Specifikace dynamického modulárního systému pro programovací jazyk Java. *OSGi* umožňuje instalaci a odebírání modulů za běhu, definuje životní cyklus modulu a nabízí infrastrukturu pro spolupráci modulů skrze služby.

SWT - *Standard Widget Toolkit*

Knihovna grafických uživatelských prvků pro platformu Java.

Citovaná literatura

[1] Eclipse - The Eclipse Foundation open source community website.
About the Eclipse Foundation. [Online]
URL: http://www.eclipse.org/org/
[citováno: 20. 4 2013.]

[2] Eclipse Downloads.
Eclipse - The Eclipse Foundation open source community website. [Online]
URL: http://www.eclipse.org/downloads/
[citováno: 25. 4 2013.]

[3] Platform architecture. *Eclipse documentation - Current Release.* [Online]
URL:http://help.eclipse.org/juno/index.jsp?topic=%2Forg.eclipse.platform.doc.isv%2Fguide%2Farch.htm
[citováno: 18. 4 2013.]

[4] Equinox Framework.
Eclipse - The Eclipse Foundation open source community website. [Online]
URL: http://www.eclipse.org/equinox/framework/
[citováno: 21. 4 2013.]

[5] Editor, pohled a perspektiva.
Eclipse názory. [Online]
URL: http://eclipse.nazory.cz/01_Workbench/03.htm
[citováno: 20. 4 2013.]

[6] E. Clayberg, D. Rubel,**.** *Eclipse Plug-ins, Third Edition.*
Boston, Massachusetts : Pearson Education, Inc., 2009.
ISBN 978-0-321-55346-1.

[7] Eclipse Plugin Development TUTORIAL.
Eclipse Plugin Development. [Online]
URL: http://www.eclipsepluginsite.com/index.html
[citováno: 20. 4 2013.]

[8] Swing.
kore.fi.muni.cz. [Online]
URL: http://kore.fi.muni.cz/wiki/index.php/Swing
[citováno: 20. 4 2013.]

[9] How to Make Frames (Main Windows).
The Java Tutorials. [Online]]
URL: http://docs.oracle.com/javase/tutorial/uiswing/components/frame.html
[citováno: 11. 4 2013.]

[10] How to Use Buttons, Check Boxes, and Radio Buttons.
The Java Tutorials. [Online]
URL: http://docs.oracle.com/javase/tutorial/uiswing/components/button.html
[citováno: 11. 4 2013.]

[11] How to Use Combo Boxes.
The Java Tutorials. [Online]
URL: http://docs.oracle.com/javase/tutorial/uiswing/components/combobox.html
[citováno: 11. 4 2013.]

[12] How to Use Labels.
The Java Tutorials. [Online]
URL: http://docs.oracle.com/javase/tutorial/uiswing/components/label.html
[citováno: 11. 4 2013.]

[13] How to Use Lists.
The Java Tutorials. [Online]
URL: http://docs.oracle.com/javase/tutorial/uiswing/components/list.html
[citováno: 11. 4 2013.]

[14] How to Use Panels.
The Java Tutorials. [Online]
URL: http://docs.oracle.com/javase/tutorial/uiswing/components/panel.html
[citováno: 11. 4 2013.]

[15] How to Use Scroll Panes.
The Java Tutorials. [Online]
URL: http://docs.oracle.com/javase/tutorial/uiswing/components/scrollpane.html
[citováno: 11. 4 2013.]

[16] How to Use Text Areas.
The Java Tutorials. [Online]
URL: http://docs.oracle.com/javase/tutorial/uiswing/components/textarea.html
[citováno: 11. 4 2013.]

[17] How to Use Text Fields.
The Java Tutorials. [Online]
URL: http://docs.oracle.com/javase/tutorial/uiswing/components/textfield.html
[citováno: 11. 4 2013.]

[18] Správce rozvržení (Layout Manager).
ing. Vojtěch Hordějčuk. [Online]
URL: http://voho.cz/wiki/informatika/jazyk/java/layout-manager/
[citováno: 25. 4 2013.]

[19] Herout, Pavel. *Java grafické uživatelské prosteředí a čeština.*
České Budějovice : Koop, 2007.
ISBN 978-80-7232-328-9

[20] How to Use BoxLayout.
The Java Tutorias. [Online]
URL: http://docs.oracle.com/javase/tutorial/uiswing/layout/box.html
[citováno: 21. 4 2013.]

[21] How to Use SpringLayout.
The Java Tutorials. [Online]
URL: http://docs.oracle.com/javase/tutorial/uiswing/layout/spring.html
[citováno: 1. 5 2013.]

[22] Introduction to Event Listeners.
The Java Tutorials. [Online]
URL: http://docs.oracle.com/javase/tutorial/uiswing/events/intro.html
[citováno: 21. 4 2013.]

[23] Lesson: Using Swing Components.
The Java Tutorials. [Online]
URL: http://docs.oracle.com/javase/tutorial/uiswing/components/
[citováno: 22. 4 2013.]

[24] Java Desktop application: SWT vs. Swing.
Stack Overflow. [Online]
URL: http://stackoverflow.com/questions/2306190/java-desktop-application-swt-vs-swing
[citováno: 28. 3 2013.]

Seznam obrázků

Seznam tabulek

Přílohy dokumentu

Příloha A Instalační a uživatelská příručka

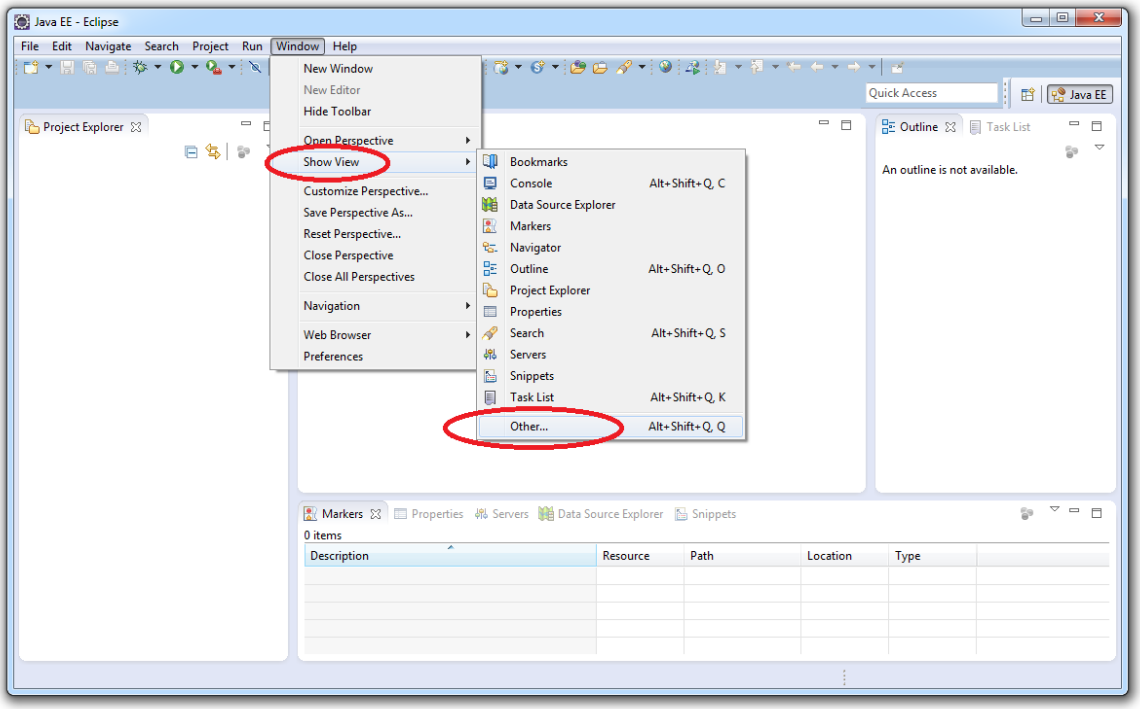
A.1 Požadavky

- nainstalované JRE 7
- Vývojové prostředí Eclipse Juno Service Release 2 a novější

A.2 Instalace pluginu

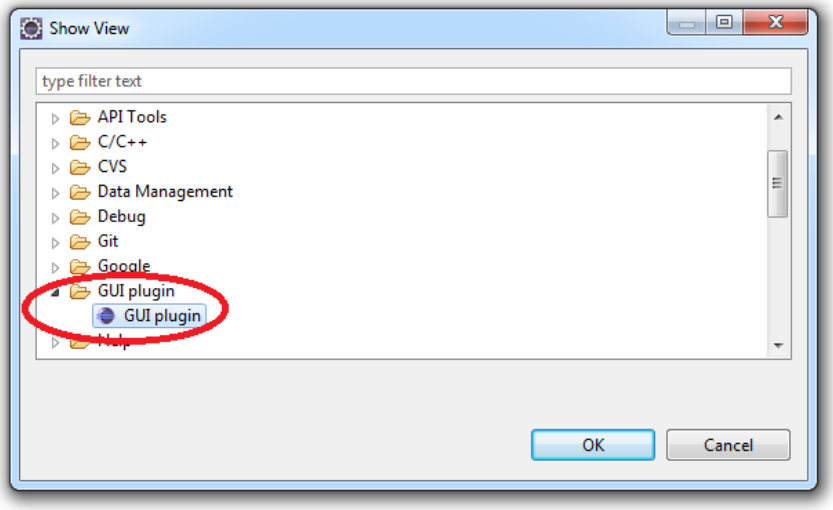
Soubor *GUIplugin1.0.jar*, který se nachází na přiloženém DVD ve složce *plugin*, stačí nakopírovat do složky *dropins* v instalaci Eclipse. Pokud je Eclipse spuštěný, je nutné ho restartovat.

Po spuštění vývojového prostředí Eclipse, se automaticky načte i vložený plugin. Pro jeho využívání je nutné v hlavním menu Eclipse zvolit *Window – Show View – Other…* (viz Obrázek A.1)



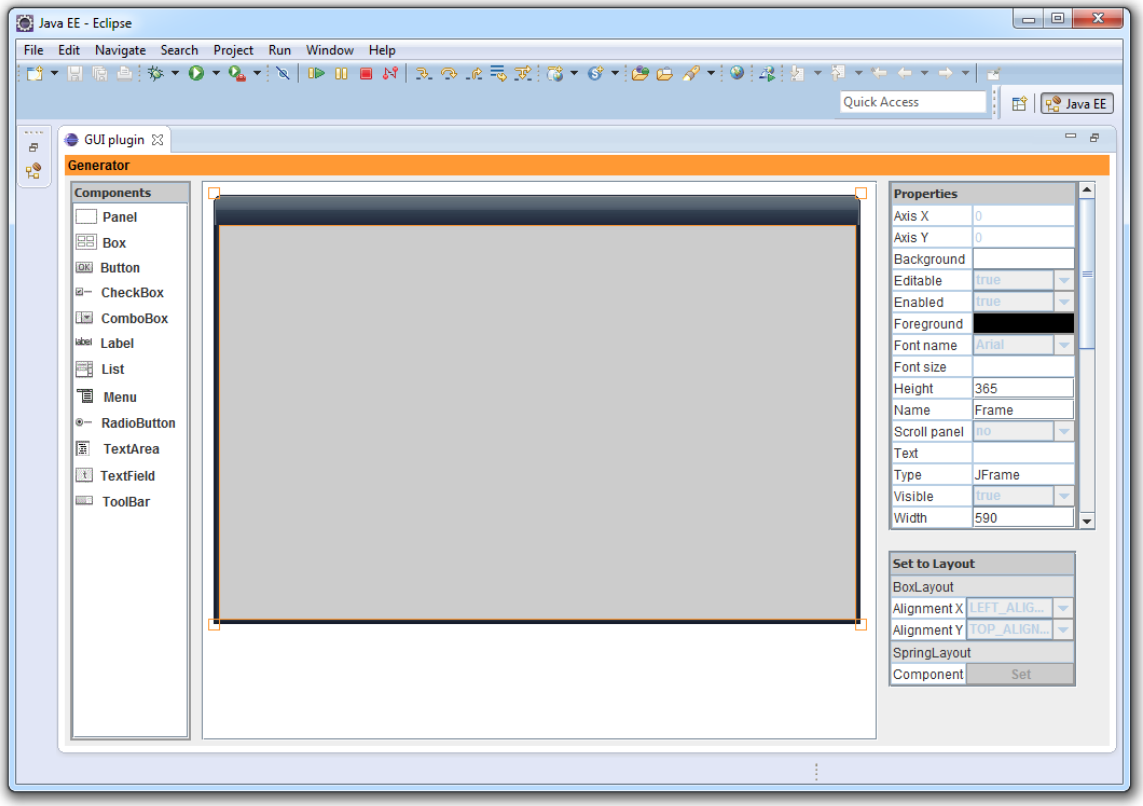
Obrázek A.1: Vložení pluginu - 1. Krok

Zobrazí se okno, ve kterém zvolíme složku *GUI plugin* a následně položku *GUI plugin* (viz Obrázek A.2).



Obrázek A.2: Vložení pluginu - 2. Krok

Po stisknutí tlačítka *OK* se zobrazí okno pluginu (viz Obrázek A.3) a můžeme začít tvořit své vlastní návrhy.



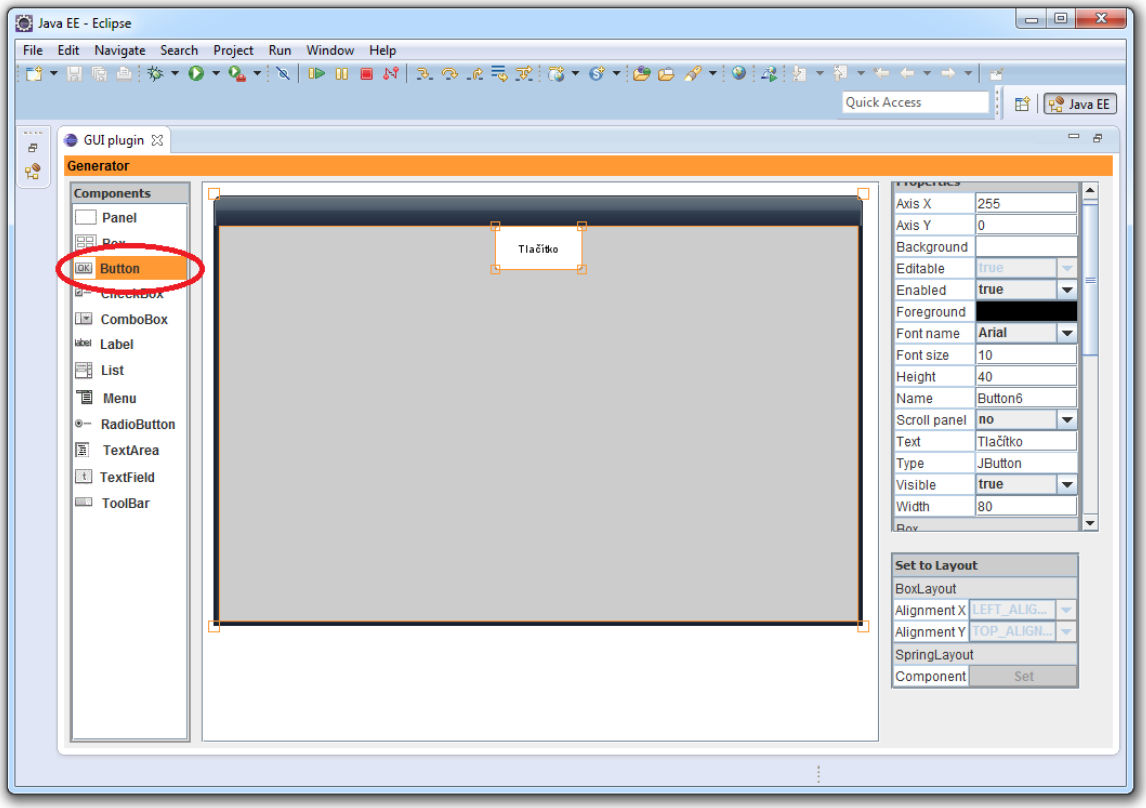
Obrázek A.3: Vložení pluginu - 3. Krok

A.3 Vytváření návrhu GUI

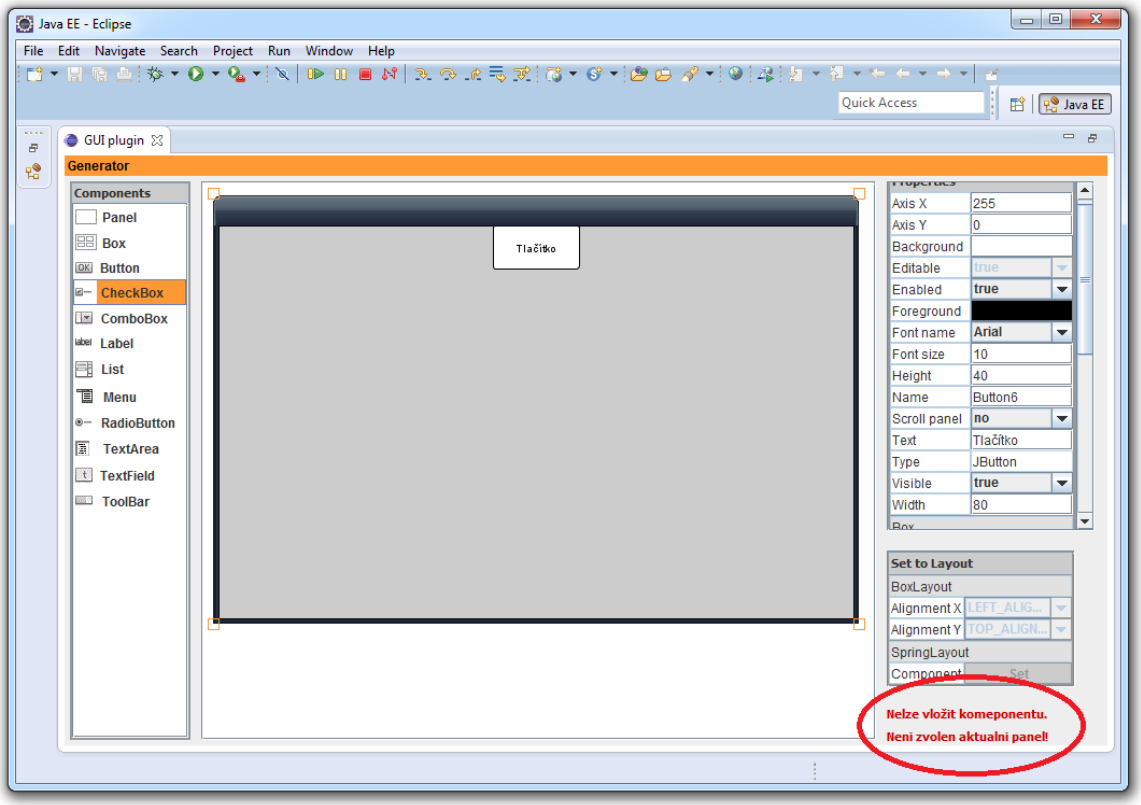
A.3.1 Vložení a smazání komponenty

Pro *vložení komponenty* je nutné nejprve kliknout levým tlačítkem myši na panel, do kterého chceme komponentu vložit. Následně je možné kliknout na libovolnou komponentu z nabídky v levé části okna (nabídka s názvem *Components*) a tím jí vložit do návrhu (viz Obrázek A.4). Pokud nebude zvolen žádný panel pro vložení komponenty nebo bude vkládána komponenta, kterou není možné aktuálně využít, bude zobrazeno upozornění (viz Obrázek A.5).

Smazání komponenty je umožněno pouze tak, že umístíme kurzor myši na prvek, který chceme odstranit (prvek bude zvýrazněn) a stiskneme ***pravé*** *tlačítko myši*. Takto je možné smazat všechny komponenty mimo hlavního okna, do kterého vkládáme komponenty a jeho panelů. Odstranění panelů je popsáno v kapitole A.3.4.



Obrázek A.4: Vložení komponenty



Obrázek A.5: Ukázka chybové hlášky při vkládání komponenty

A.3.2 Nastavení vlastností komponenty

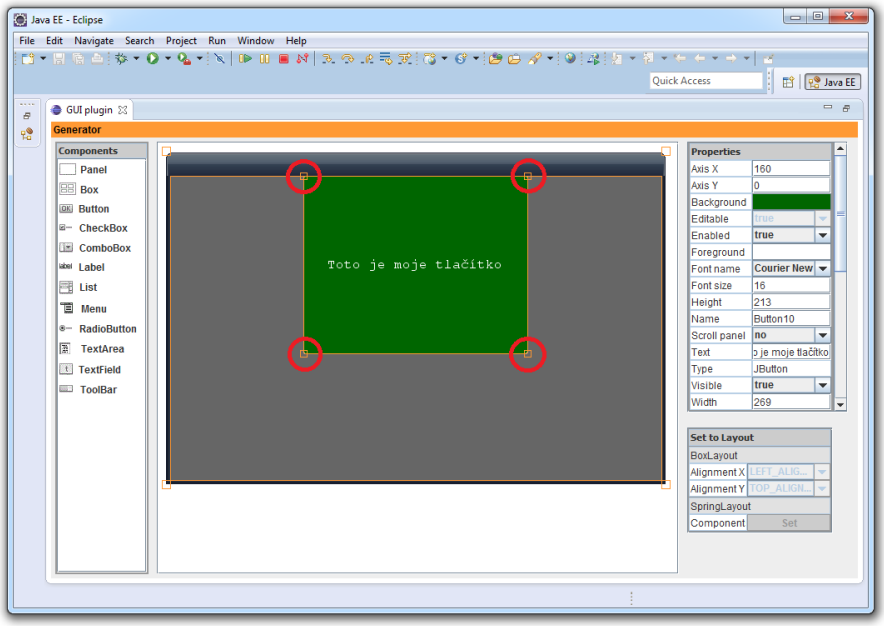
Po označení komponenty (klik levým tlačítkem myši na zvolenou komponentu) jsou v pravé části okna s názvem *Properties* vyplněny její vlastností. Význam vlastností je popsán v Tabulce A.1. Položky, které je možné aktuálně měnit, mají povolenu editaci. V případě zadání nevalidních dat je zobrazeno upozornění.

Změna rozměrů komponenty je umožněna také pomocí myši. Umístěním kurzoru myši do objektu komponenty, dojde k jejímu zvýraznění. Zároveň jsou zobrazeny i záchytné body, které umožňují změnit její rozměry. Tyto body jsou označeny na Obrázku A.6. Stačí tedy umístit kurzor myši do záchytného bodu, držet stisknuté levé tlačítko myši a tahem změnit rozměry komponenty.

Upozornění: Nebude-li ukazatel myši v záchytném rohu ale bude uvnitř komponenty, dojde místo změny rozměru k jejímu přesunu.

Tabulka A.1: Význam jednotlivých vlastností komponent

Název vlastnosti	Popis
Axis X	Souřadnice na ose X, na které je umístěn levý horní roh komponenty v okně návrhu. Nultý bod je umístěn v levém horním rohu.
Axis Y	Souřadnice na ose Y, na které je umístěn levý horní roh komponenty v okně návrhu. Nultý bod je umístěn v levém horním rohu.
Alignment X	Určuje zarovnání komponenty na ose X, při použití BoxLayoutu.
Alignment Y	Určuje zarovnání komponenty na ose Y, při použití BoxLayoutu.
Background	Barva pozadí.
Box type	Nastavení typu použitého boxu.
ButtonGroup	Umožňuje nastavit skupinu, do které bude prvek patřit.
Columns	Počet sloupců.
Editable	Nastavuje možnost editace komponenty.
Enabled	Nastavuje možnost povolení komponenty.
Foreground	Barva popředí.
Font name	Výběr druhu písma.
Font size	Nastavení velikosti písma.
Height	Nastavení výšky komponenty.
Checked	Zaškrtnutí komponenty.
Items	Umožňuje naplnit komponentu prvky.
Name	Název komponenty.
Rows	Počet řádek.
Scroll panel	Nastavuje možnost přiřazení scroll panelu ke komponentě.
Text	Text zobrazený v komponentě.
Visible	Určuje, zda bude komponenta viditelná.
Width	Nastavení šířky komponenty.



Obrázek A.6: Záchytné body pro změnu velikosti komponenty.

A.3.3 Změna umístění komponenty

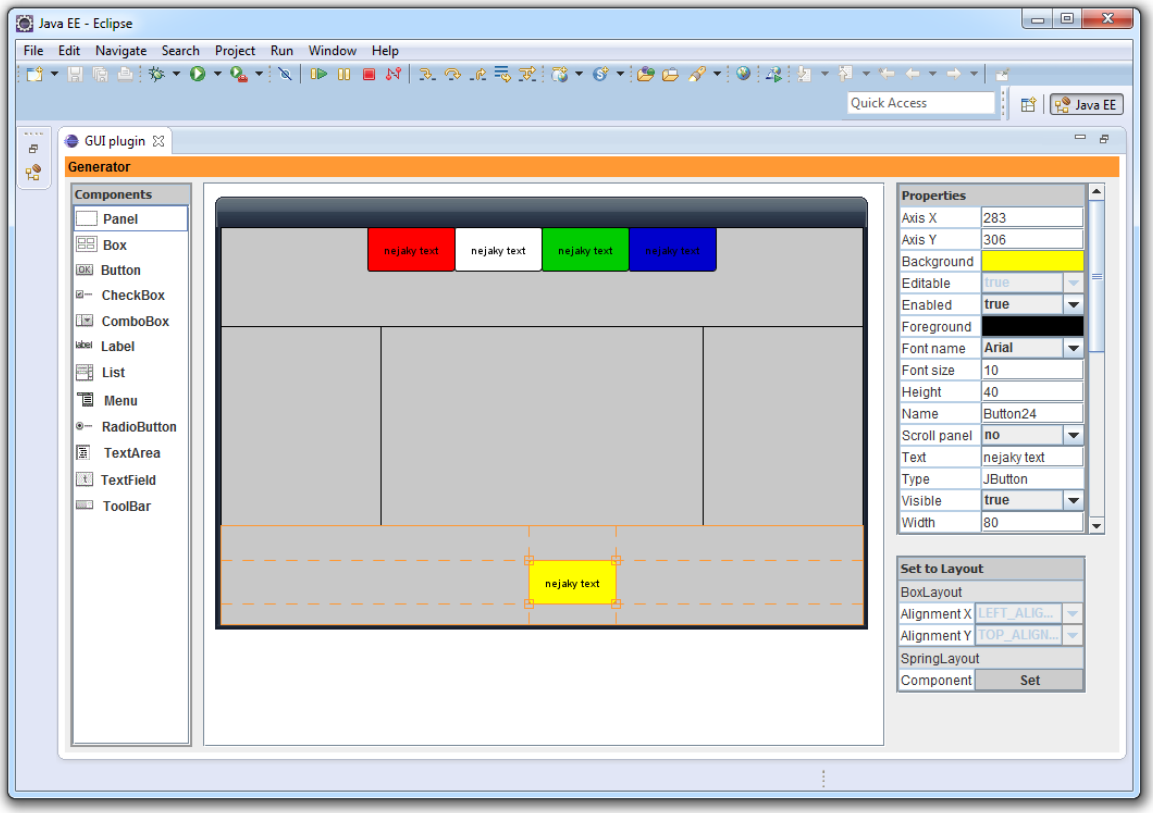
V rámci *jednoho panelu* je možné měnit pozice prvků buď prohozením dvou komponent, nebo prostým přesunem komponenty na zvolenou pozici. Vše je závislé na zvoleném layout manageru.

Prohození dvou komponent lze udělat tak, že najedeme ukazatelem myši na zvolenou komponentu, kterou budeme chtít přesunout. Stiskneme levé tlačítko myši a přesuneme ukazatel nad cílovou komponentu, kterou chceme nahradit. Tlačítko nyní uvolníme. Přesun komponenty je zobrazen na Obrázku A.8. Zde přesouváme zelené tlačítko na místo bílého tlačítka. Počáteční stav je zobrazen na Obrázku A.7.

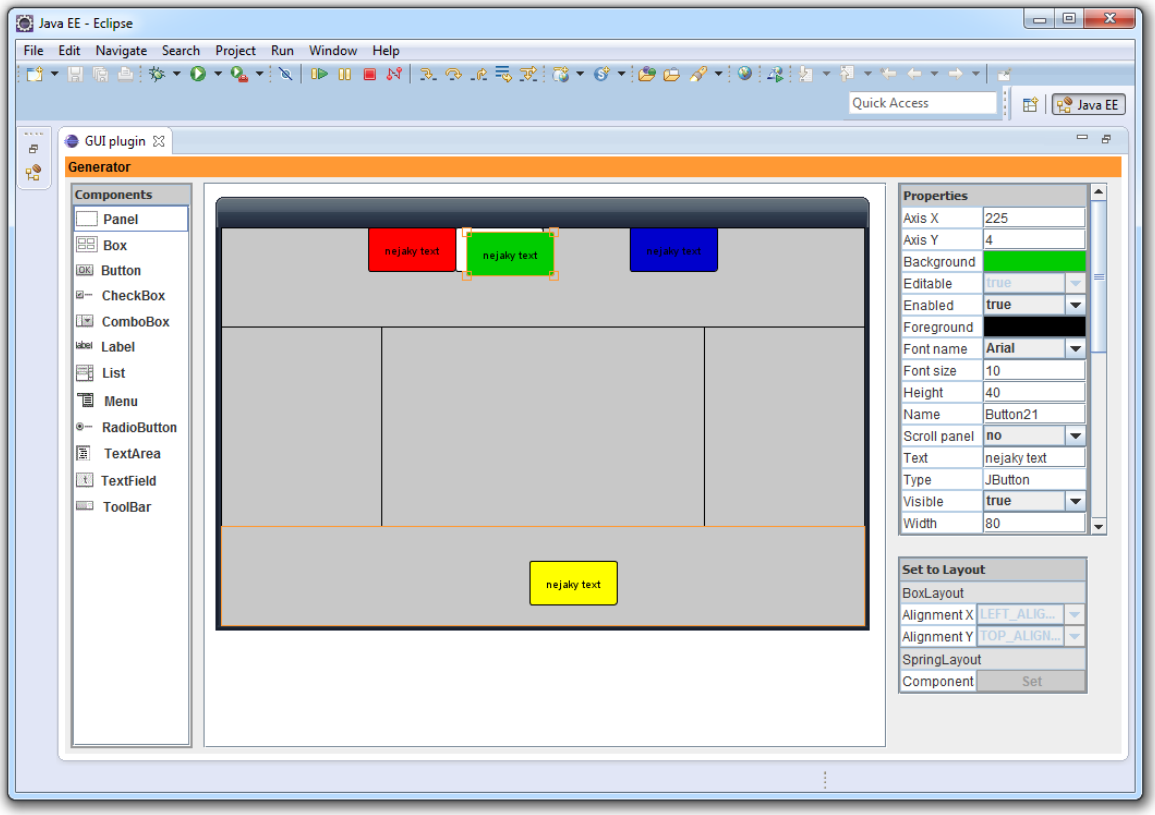
Přesun komponenty na námi zvolenou pozici lze udělat tak, že umístíme ukazatel myši na zvolenou komponentu, kterou budeme chtít přesunout. Stiskneme levé tlačítko myši a přesuneme ukazatel na námi zvolené místo. Při přesunu se zobrazí orientační linky komponenty viditelné na Obrázku A.7 (žluté tlačítko). Tento přesun je možné využívat, pouze pokud je v panelu nastaven `SpringLayout`.

V rámci *více panelů* lze komponenty pouze přesouvat, nikoliv prohazovat.

Upozornění: Komponenta s názvem Box, umožňuje pouze přesun celé skupiny komponent. Je zakázán přesun komponenty z Panelu do Boxu. Prohození pozic komponent uvnitř Boxu je povoleno.



Obrázek A.7: Přesun komponent - počáteční stav

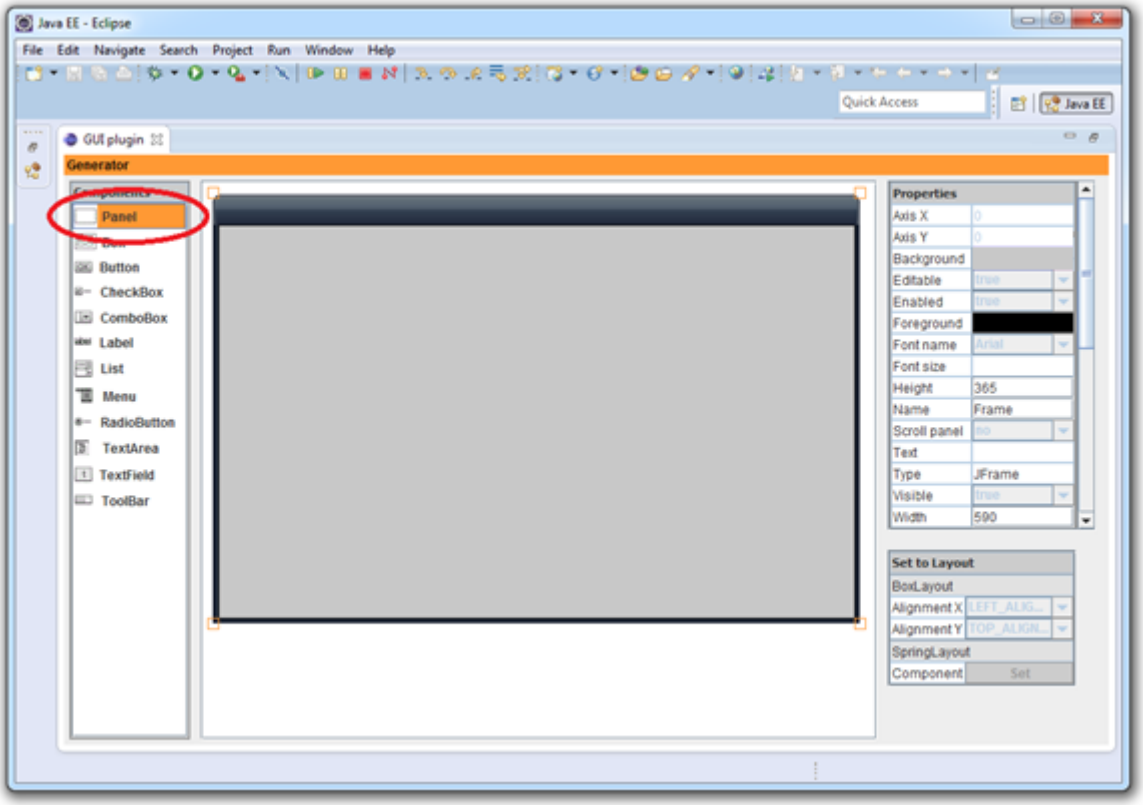


Obrázek A.8: Přesun komponent - prohození dvou komponent

A.3.4 Správce panelů

Správce panelů umožňuje vkládat, editovat a mazat `JPanely` ve vznikajícím návrhu nového okna. Dále je zde umožněno nastavit jednotlivým panelům vlastní layout managery.

Správce panelů se nachází v levé části okna, nabídka s názvem *Components*, položka *Panel* (viz Obrázek A.9).

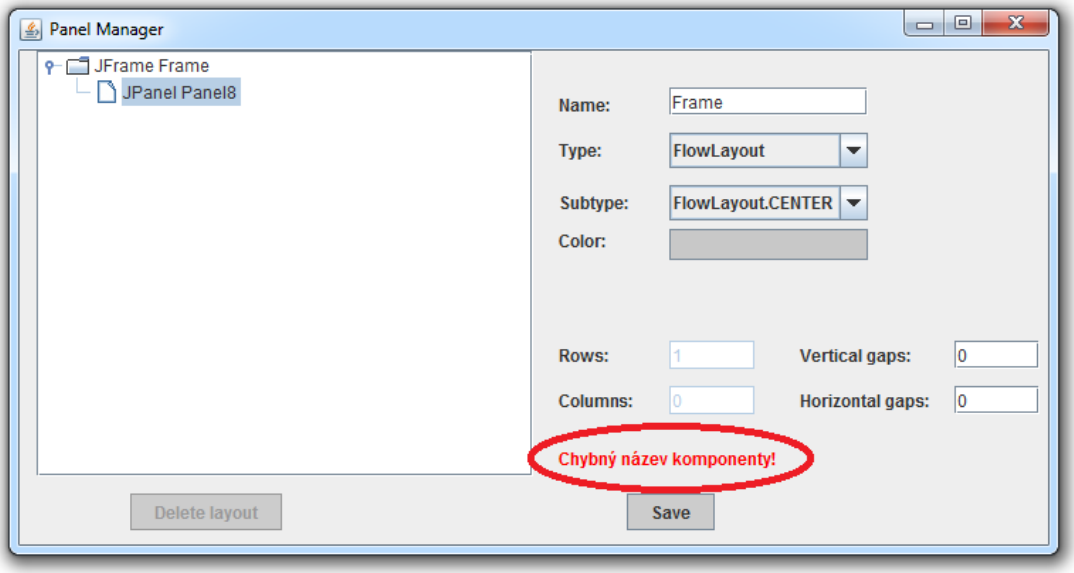


Obrázek A.9: Správce panelů - umístění správce v nabídce

Po kliknutí na položku je zobrazeno okno správce panelů. Pro jednotlivé layout managery je možné nastavit další vlastnosti. Nastavené hodnoty jsou validovány. V případě špatně zadané hodnoty je uživatel informován (viz Obrázek A.10). Pokud nejsou tyto vlastnosti nastaveny, budou použity defaultní. Panel, jehož layout manager chceme editovat, označíme ve stromové struktuře (levá část okna).

Významy vlastností:

- Name – název komponenty
- Type – typ layout manageru
- Subtype – upřesnění typu layout manageru
- Rows – počet řádek do kterých se budou vkládat komponenty
- Columns – počet sloupců do kterých se budou vkládat komponenty
- Vertical gaps – velikosti vertikálních mezer mezi komponentami
- Horizontal gaps – velikosti horizontálních mezer mezi komponentami

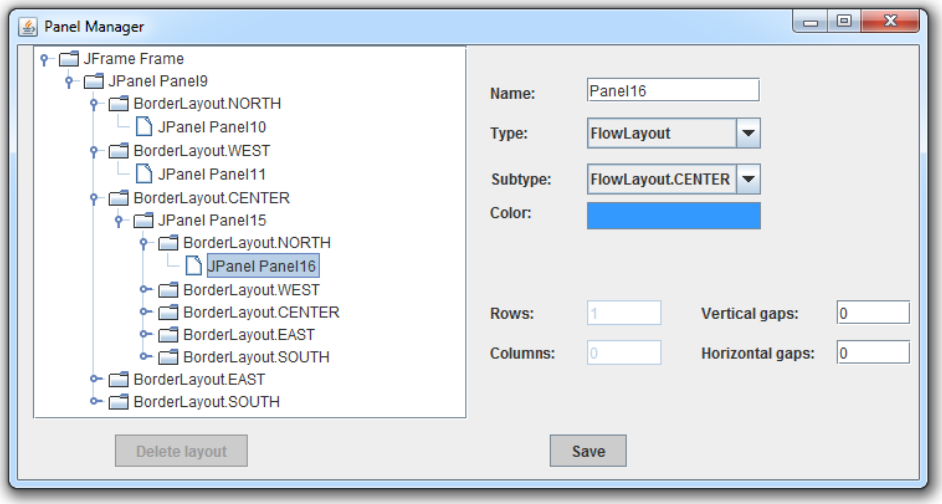


Obrázek A.10: Správce panelů - validace vstupních dat

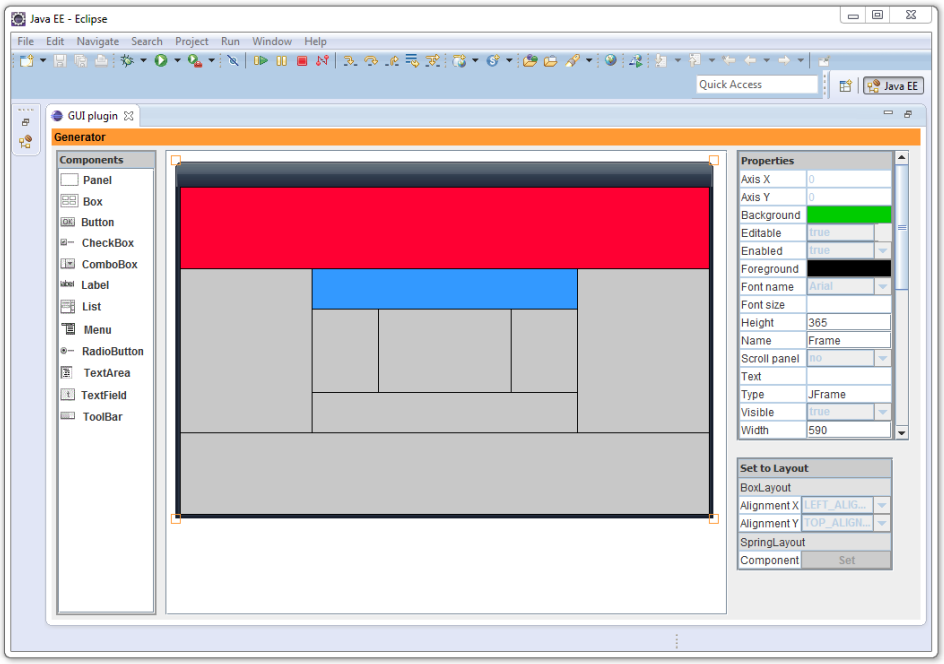
Odstranění prvku provádíme tak, že v levé části nejprve zvolíme panel, který má být vymazán a stiskneme tlačítko *Delete layout*. Pokud tlačítko nelze stisknout, znamená to, že prvek není možné odstranit.

Vhodným použitím layout managerů je možné rozdělit celý návrh a v každé části použít jiný způsob rozmísťování komponent. Ukázka použití správce návrhu je na Obrázku A.11. Zde je do `BorderLayoutu` vložen druhý `BorderLayout`. Výsledný vzhled okna při takto použitém složení layout managerů je zobrazen na Obrázku A.12.

Upozornění: Při použití `BorderLayoutu` dojde k odstranění původního panelu a to včetně komponent, které jsou v něm umístěny.



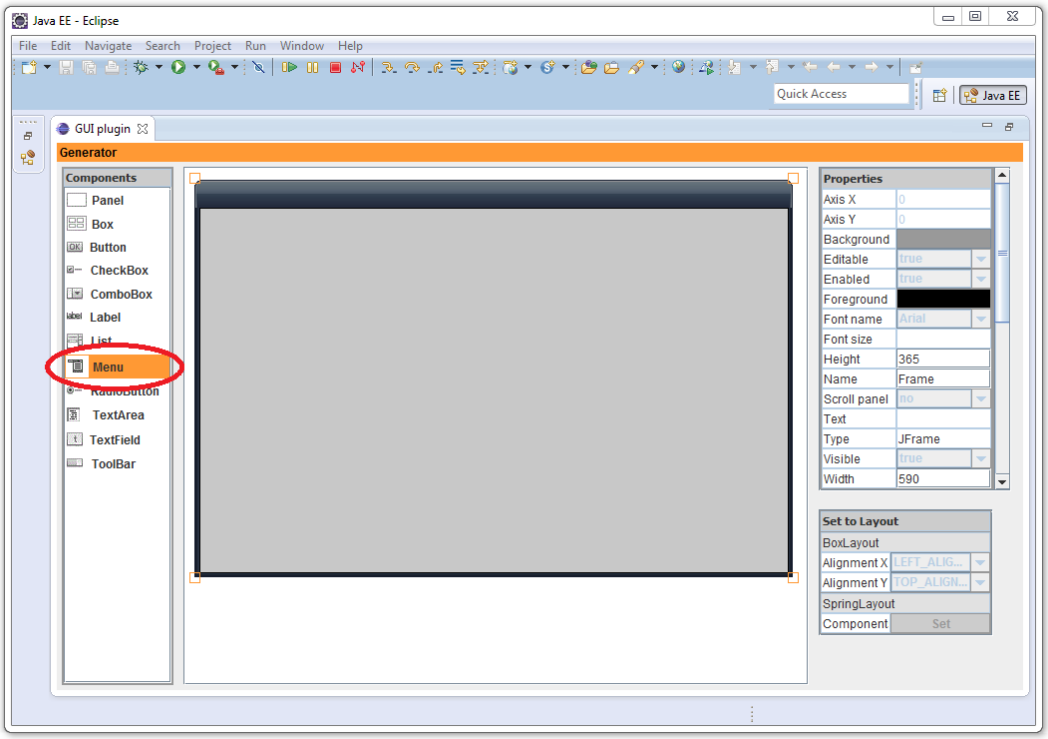
Obrázek A.11: Správce panelů - Ukázka sestavení panelů



Obrázek A.12: Správce panelů - Reálná podoba návrhu

A.3.5 Vytvoření menu

Správce menu se nachází v levé části okna, nabídka s názvem *Components*, položka *Menu* (viz Obrázek A.13).



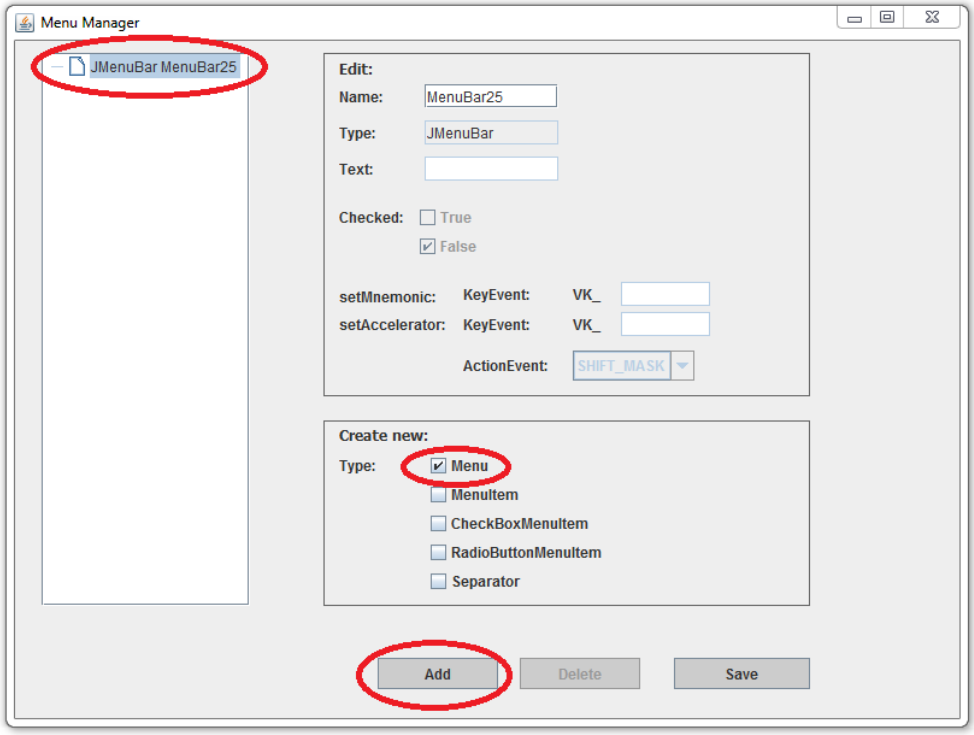
Obrázek A.13: Správce menu - umístění správce v nabídce

Po kliknutí na položku je zobrazeno okno správce menu. Zde je možné libovolně navrhnout celé menu. Jeho aktuální podoba je vidět v levé části okna.

Před *vložením komponenty* do menu je nutné nejprve zvolit v levé části prvek, do kterého bude nová komponenta vložena. Dále musíme v sekci *Create new* vybrat druh nového prvku a vše potvrdit stisknutím tlačítka *Add* (viz Obrázek A.14).

Odstranění prvku provádíme obdobně. V levé části nejprve zvolíme prvek, který má být vymazán a stiskneme tlačítko *Delete*. Pokud tlačítko nelze stisknout, znamená to, že prvek není možné odstranit.

Upozornění: Vkládání prvků do menu není validováno. Je tedy možné vkládat prvky naprosto libovolně. Takto může vzniknout nevalidní kód!



Obrázek A.14: Správce menu - přidání komponenty do menu

Pro jednotlivé prvky menu je možné nastavit další vlastnosti. Ty jsou validovány. V případě špatně zadané hodnoty je uživatel informován. Pokud nejsou tyto vlastnosti nastaveny, budou použity defaultní. Prvek, který chceme editovat, označíme ve stromové struktuře (levá části okna).

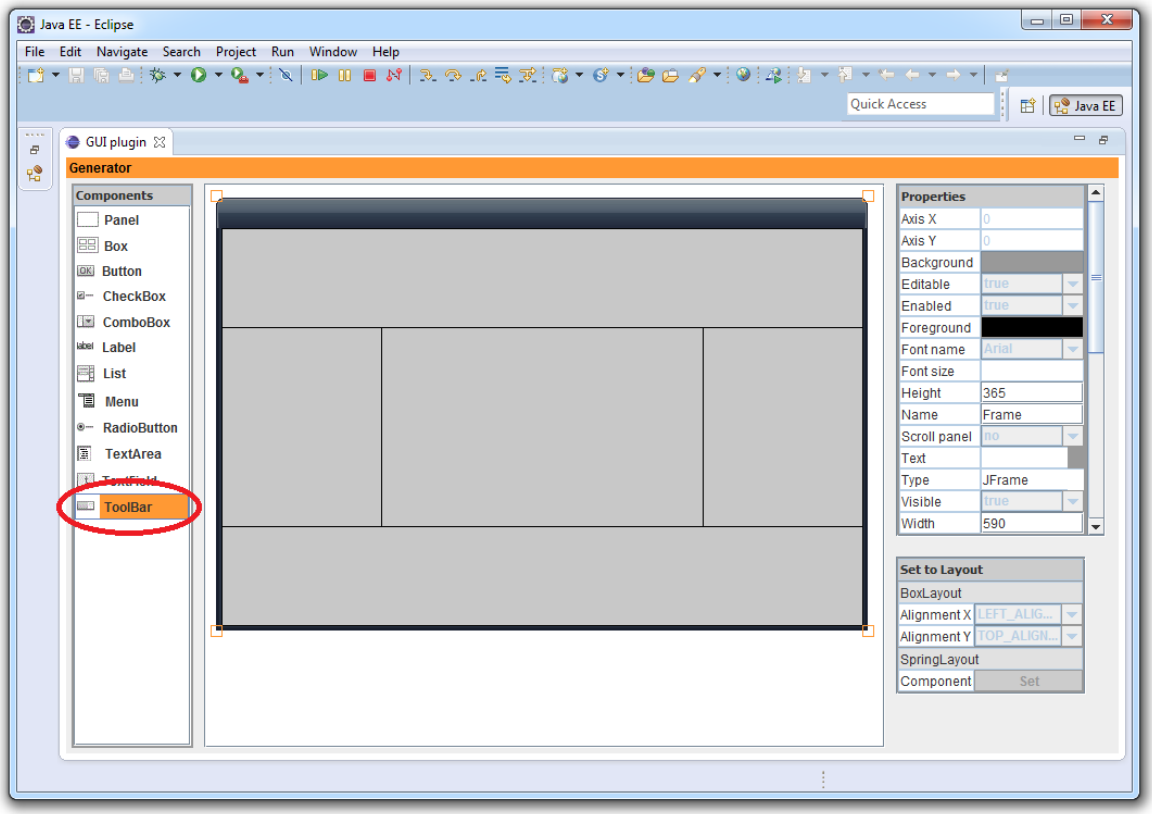
Významy vlastností:

- Name – název komponenty
- Text – text popisující položku menu
- Checked – označuje, zda má být položka zaškrtnuta
- setAccelerator – kombinace kláves pro provedení akce (např. Alt + A)
    - ActionEvent – první spouštěcí klávesa
    - KeyEvent – druhá spouštěcí klávesa např. A (hodnota není validována)
- setMnemonic - možnost aktivovat položku menu.
    - KeyEvent – aktivační klávesa

A.3.6 Správce toolbarů (nástrojové lišty)

Umožňuje vkládat, editovat a mazat `JToolBary`. Správce je možné využívat, pouze pokud je na hlavním panelu nastaven `BorderLayout`.

Správce panelů se nachází v levé části okna, nabídka s názvem *Components*, položka *ToolBar* (viz Obrázek A.15).



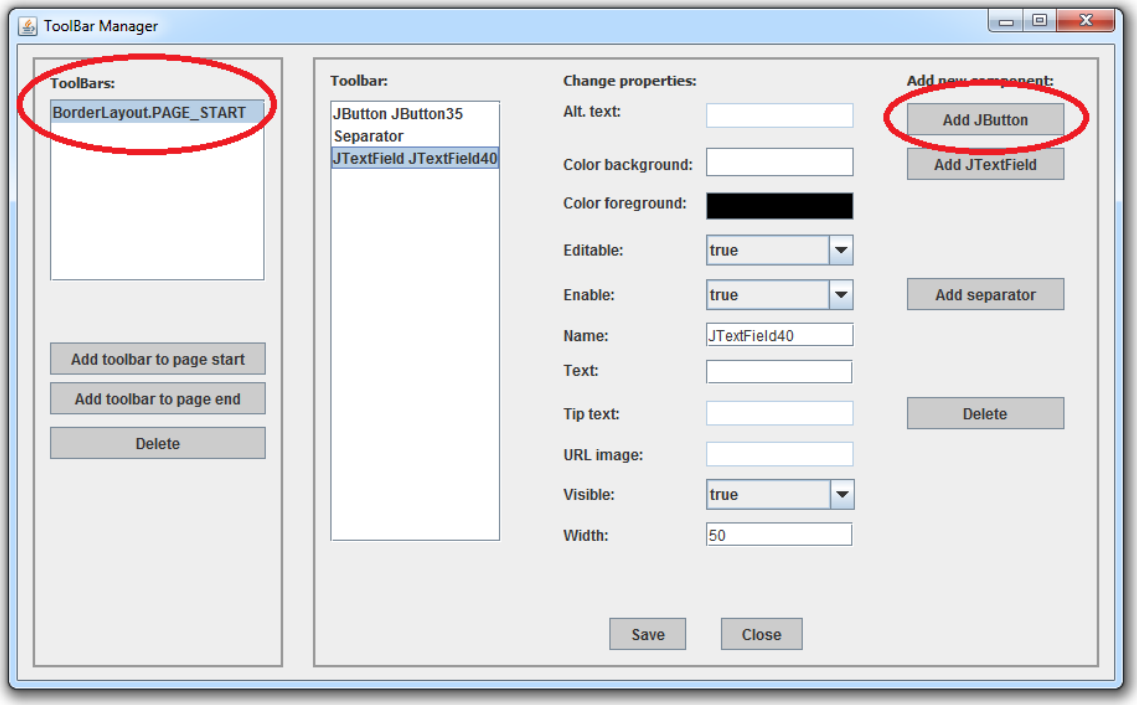
Obrázek A.15: Správce toolbarů - umístění správce v nabídce

Po kliknutí na položku je zobrazeno okno správce toolbarů. Zde je možné libovolně navrhnout nové toolbary.

Před *vložením komponenty* do toolbaru je nutné nejprve zvolit v sekci *Toolbars* prvek, do kterého bude nová komponenta vložena. Pokud je aktuální nabídka prázdná, klikneme na tlačítko *Add toolbar to page start* nebo na tlačítko *Add toolbar to page end*. Z názvu tlačítek je patrné, že určují pozici v panelu, do které bude toolbar umístěn. Toolbar na začátku i na konci stránky může být pouze jeden.

Dále musíme v pravé části okna kliknout na tlačítko pro přidání zvoleného prvku (např. *Add JButton*). Nyní je přidán prvek do toolbaru (viz Obrázek A.16).

Odstranění prvku provádíme tak, že v levé části nejprve zvolíme prvek v seznamu, který má být vymazán a stiskneme tlačítko *Delete*. Takto se smaže celý toolbar. Obdobně mažeme i konkrétní prvky v toolbaru.



Obrázek A.16: Správce toolbaru - přidání komponenty

Pořadí prvků uvnitř toolbaru je možné měnit. Stačí označit prvek v seznamu komponent *Toolbar*, stisknout levé tlačítko myši a pohybem kurzoru určit jeho nové umístění.

Pro jednotlivé prvky toolbaru je možné *nastavit další vlastnosti*. Ty jsou validovány. V případě špatně zadané hodnoty je uživatel informován. Pokud nejsou tyto vlastnosti nastaveny, budou použity defaultní. Prvek, který chceme editovat, označíme v libovolném seznamu.

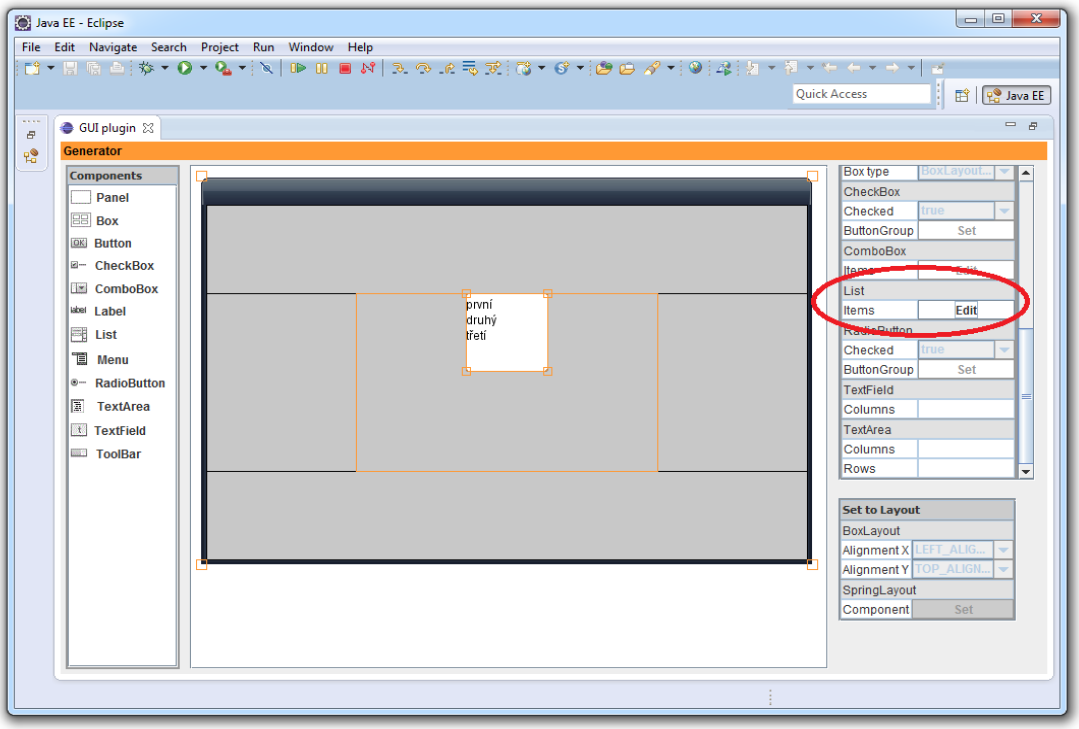
Významy vlastností:

- Alt. text – text bude zobrazen v komponentě, pokud nebude nalezen obrázek
- Color background – barva pozadí komponenty
- Color foreground – barva popředí komponenty
- Editable – určuje, zda je komponenta editovatelná.
- Enable – určuje, zda je komponenta aktivní
- Name – název komponenty
- Text – text zobrazený v komponentě
- Tip text – text bude zobrazen jako popisek kurzoru myši při umístění kurzoru na komponentu
- URL image – URL adresa obrázku, který bude v komponentě zobrazen (adresa není validována)
- Visible – určuje, zda je komponenta viditelná
- Width – určuje šířku komponenty (počet sloupců)

A.3.7 Naplnění obsahu komponenty

Tato část je určena pouze pro komponenty `JComboBox` a `JList`. Těm může být nastaven obsah.

Označíme tedy komponentu typu ComboBox nebo List. V pravé nabídce *Properties* nalezneme její sekci a u položky *Items* klikneme na tlačítko *Edit* (viz Obrázek A. 17).



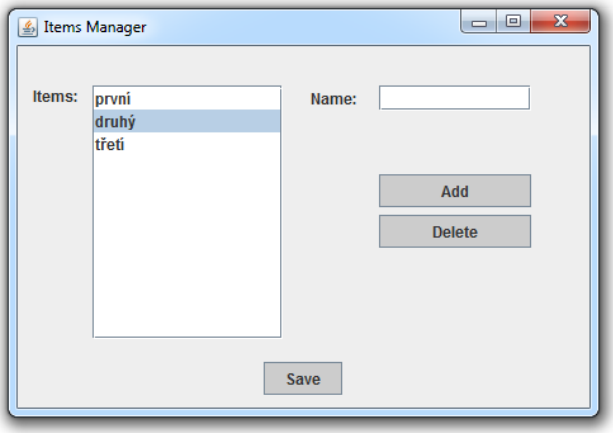
Obrázek A. 17: Správce naplnění obsahu komponenty - umístění správce

Po kliknutí na tlačítko *Edit* je zobrazeno okno správce naplnění obsahu (viz Obrázek A.18). Zde je možné vytvořit seznam prvků, který bude daná komponenta obsahovat.

Pro *vložení položky* stačí vyplnit její název a stisknout tlačítko *Add*.

Odstranění prvku provádíme tak, že v seznamu všech prvků zvolíme prvek, který má být vymazán a stiskneme tlačítko *Delete*.

Pořadí prvků uvnitř seznamu je možné měnit. Stačí označit prvek v seznamu komponent, stisknout levé tlačítko myši a pohybem kurzoru myši určit jeho nové umístění.



Obrázek A.18: Správce naplnění obsahu – náhled

A.3.8 Vložení komponenty do skupiny

Tato část je určena pouze pro komponenty `JCheckBox` a `JRadioButton`. Ty mohou tvořit skupiny.

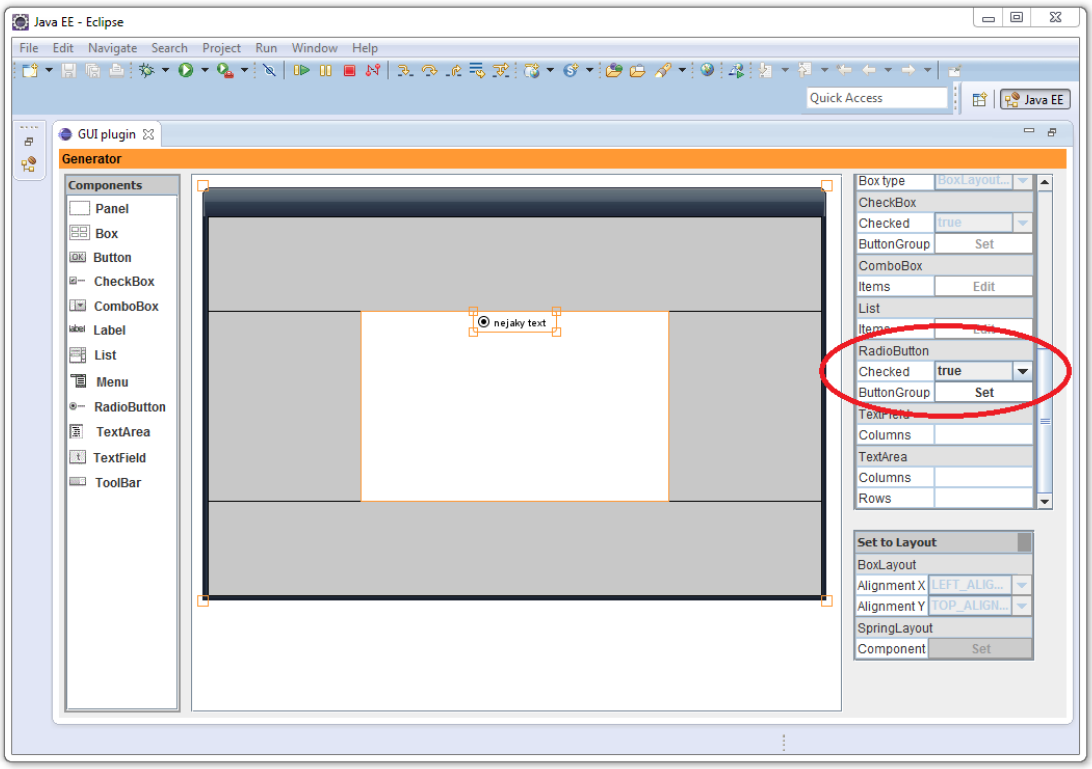
Označíme tedy komponentu typu CheckBox nebo RadioButton. V pravé nabídce *Properties* nalezneme její sekci a u položky *ButtonGroup* klikneme na tlačítko *Set* (viz Obrázek A.19).

Po kliknutí na tlačítko *Set* je zobrazeno okno správce skupin (viz Obrázek A.20). Zde je možné určit skupinu, do které bude komponenta patřit.

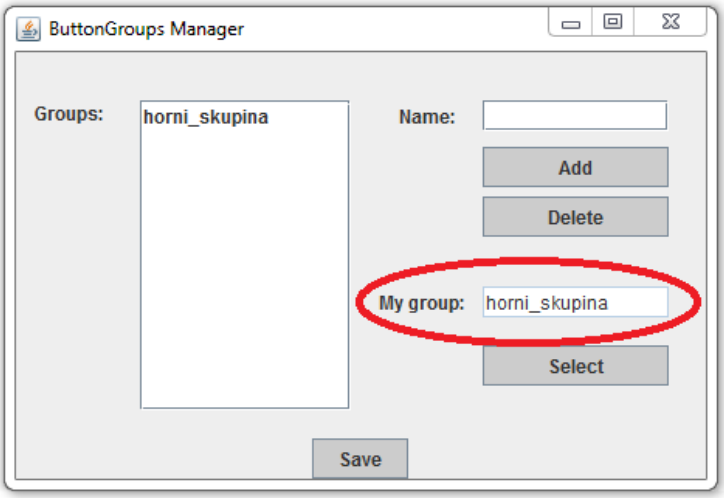
Pro *vytvoření skupiny* stačí vyplnit její název a stisknout tlačítko *Add*.

Odstranění skupiny provádíme tak, že v seznamu všech prvků zvolíme prvek, který má být vymazán a stiskneme tlačítko *Delete*.

Prvek do skupiny zařadíme tak, že označíme jednu skupinu v seznamu existujících skupin a klikneme na tlačítko *Select*. Název skupiny, do které byla komponenta vložena, se zobrazí vedle popisky *My Group* (viz Obrázek A.20). Komponenta může být členem pouze jedné skupiny.



Obrázek A.19: Správce skupin

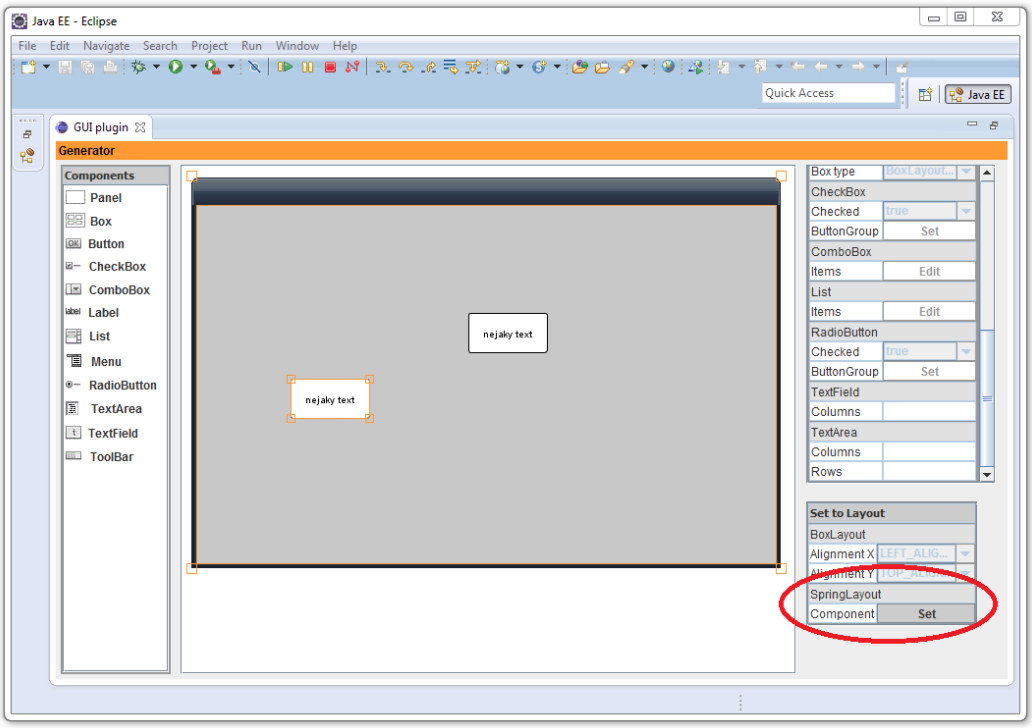


Obrázek A.20: Správce skupin - zobrazení skupiny

A.3.9 Nastavení pozice komponenty při použití SpringLayoutu

Pro použití této funkce musí být na panelu jako layout manager nastaven `SpringLayout`.

Kliknutím na levé tlačítko označíme komponentu, jejíž pozici chceme nastavit. V pravé nabídce *Set to Layout* nalezneme sekci *SpringLayout* a klikneme na tlačítko *Set* (viz Obrázek 19).



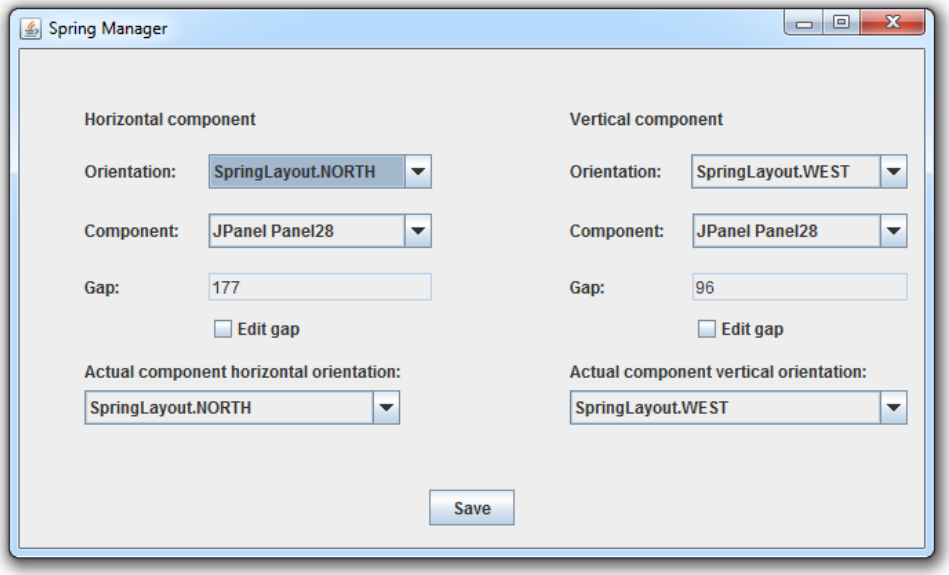
Obrázek A. 21: Správce spring layoutu - umístění

Po kliknutí na tlačítko *Set* je zobrazeno okno správce skupin (viz Obrázek A.22). Zde je možné nastavit konkrétní pozici prvku.

Pozice komponenty musí být určena jak *horizontálním* směrem, tak i *vertikálním* směrem. Nastavení komponenty popíšeme na horizontální části. Vertikální nastavení probíhá stejným způsobem.

Nejprve zvolíme hodnotu položky *Orientation*. Tím je myšlena hrana referenční komponenty, od které bude počítaná vzdálenost. Dále vybereme referenční komponentu z nabídky *Component*. Hodnota *Gap* udává mezeru mezi zvolenými hranami obou prvků. Cifra je předem vyplněna v závislosti na aktuálním stavu hodnot. Zaškrtnutím políčka *Edit gap* můžeme velikost mezery měnit a tím posunout komponentu. Poslední hodnotou, kterou nastavíme je položka *Actual component horizontal orientation*. Ta určuje hranu aktuální komponenty, od které se bude nová vzdálenost počítat.

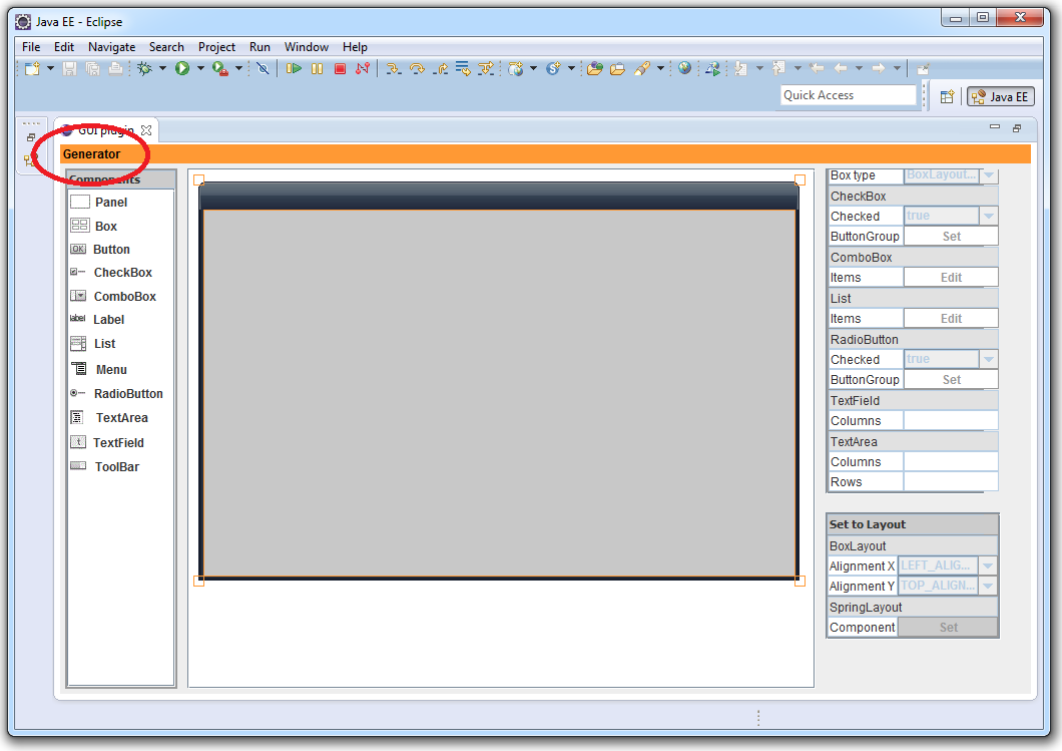
Poznámka: Polohu komponenty můžeme nastavit také tahem myši (viz kapitola A.3.3).



Obrázek A.22: Správce skupin

A.4 Generování zdrojového kódu

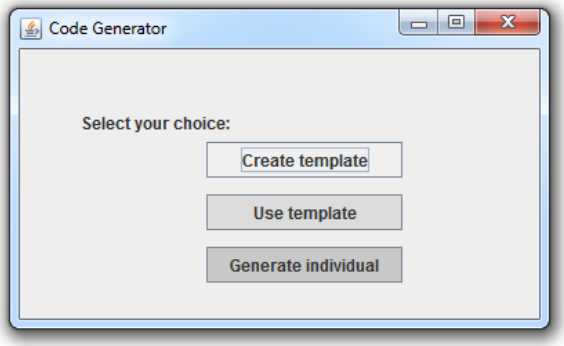
Úvodní okno generátoru zdrojového kódu se spustí kliknutím na tlačítko *Generátor*. To je umístěno v levém horním rohu návrhářského okna (viz Obrázek A.23)



Obrázek A.23: Spuštění generátoru zdrojového kódu

Po kliknutí na tlačítko *Generátor* se zobrazí úvodní okno generátoru zdrojového kódu (viz Obrázek A.24). Zde si můžeme vybrat z několika možností.

- *Create template* – vytvoření šablony pro generování zdrojového kódu
- *Use template* – na základě již existující šablony bude vygenerován zdrojový kód
- *Generate individual* – individuální návrh zdrojového kódu



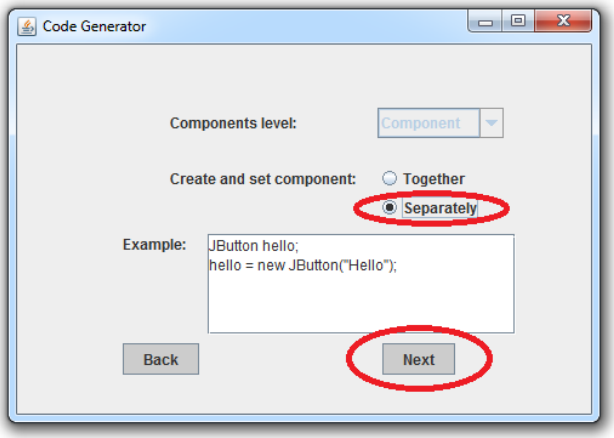
Obrázek A.24: Úvodní okno generátoru zdrojového kódu

A.4.1 Vytvoření šablony

V úvodním okně generátoru zdrojového kódu (viz Obrázek A.24) klikneme na tlačítko *Create template*. Zobrazí se nové okno pro volbu způsobu vytváření komponent.

- *Together* – komponenta bude v jednom místě zdrojového kódu vytvořena i nastavena
- *Separately* – vytvoření a nastavení komponenty ve zdrojovém kódu může probíhat odděleně

Zvolíme například možnost *Separately* a pokračujeme dále kliknutím na tlačítko *Next* (viz Obrázek A.25).



Obrázek A.25: Výběr způsobu vytváření komponent

Nyní se zobrazí hlavní okno generátoru zdrojového kódu pro návrh šablony (viz Obrázek A.26).

Okno je tvořeno pomocí pěti seznamů komponent.

- *Source code* – tento seznam prvků představuje zdrojový kód. Přesně tak, jak jsou jednotlivé prvky v seznamu seřazeny, budou i vygenerovány ve zdrojovém kódu.
- *Create & Set components* – zde jsou uloženy části pro vytvoření a nastavení prvků jednotlivých skupin.
- *Add & Call component* – v tomto seznamu jsou položky, které umožňují vložení komponent do panelů, menu a toolbaru. Zároveň jsou zde umístěny i položky pro volání metod a konstrukorů.
- *Methods* – obsahuje položky, které vytváří metody a konstruktory.
- *Actions* – obsahuje položky, které jsou nutné pro nastavení akcí.

Mezi všemi seznamy (i uvnitř nich) je umožněn přesun prvku (nebo označené skupiny prvků). Stačí označit ukazatelem myši zvolený prvek, stisknout levé tlačítko myši a prvek přesunout do libovolného okna a na libovolnou pozici.

V hlavním okně můžeme libovolně *mazat* zvolené položky. Stačí kliknutím označit položky a následně stisknout tlačítko *Delete*. Je však nutné upozornit na to, že jednou smazané *položky nelze vrátit zpět*!

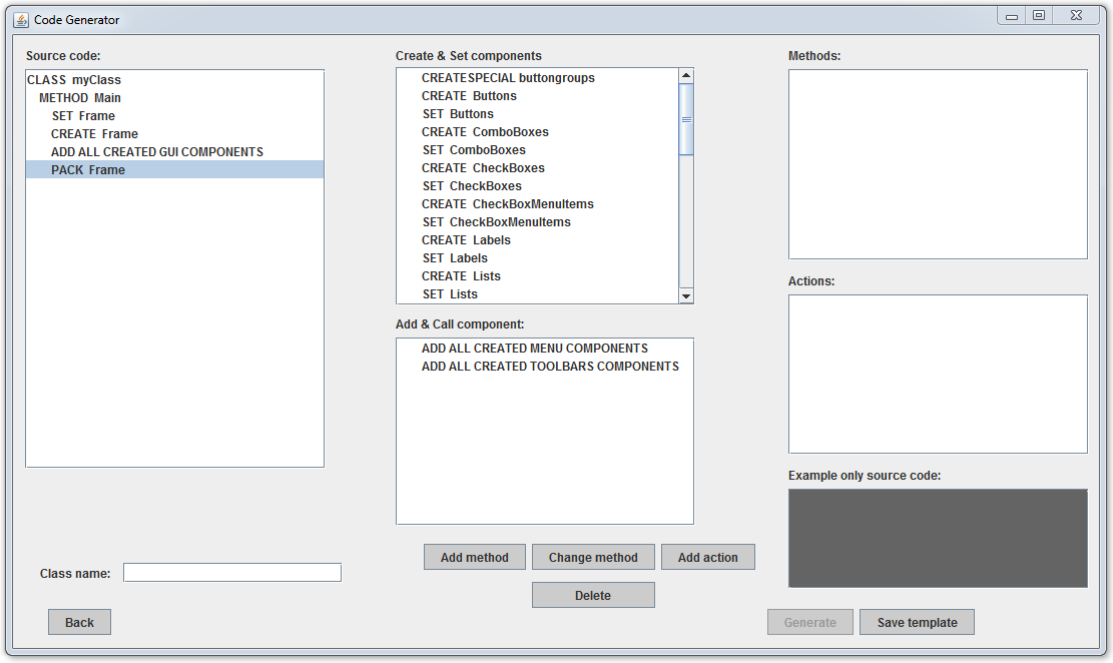
Význam položek v seznamech je vysvětlen v kapitole A.4.6.

Jak tedy *sestavit šablonu* pro generování?

1) Vytvoříme vlastní akce, metody, atd.
2) Označíme jednotlivé prvky, které chceme vložit do výsledné šablony. Stiskneme levé tlačítko myši a prvky přesuneme na zvolenou pozici v seznamu s názvem *Source code*. Pokud chceme využít pro obsluhu akcí jednu společnou třídu, vložíme jí pouze jednou.
3) Vyplníme položku *Class name*, která udává název hlavní třídy. Ta bude při každém generování podle šablony vytvořena.
4) Pro uložení šablony klikneme na tlačítko *Save template*
5) V nově otevřeném okně zvolíme cílovou složku pro umístění nové šablony a libovolně vyplníme název souboru nové šablony.
6) Stiskneme tlačítko *OK*

Poznámka: Není nutné využít všechny nabízené prvky.

Důležité upozornění: Pořadí prvků ve vytvářené šabloně není nijak validováno! Uživatel sám musí uvážit, zda jím navržený kód bude přeložitelný.



Obrázek A.26: Hlavní okno pro návrh šablony

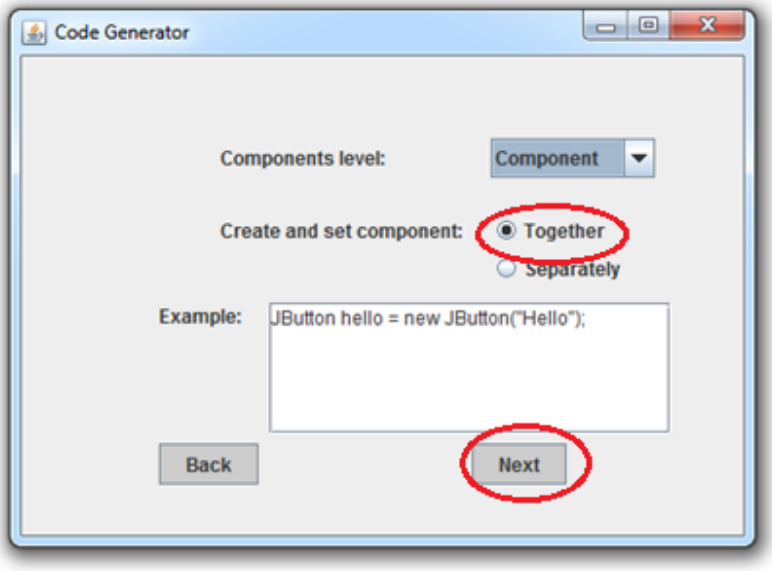
A.4.2 Generování za pomoci existující šablony

Pro generování zdrojového kódu za pomoci již existující šablony, klikneme v úvodním okně generátoru na tlačítko *Use template* (viz Obrázek A.24).

Po kliknutí bude zobrazeno první dialogové okno, kde nejdříve zvolíme šablonu, podle které bude vytvořen zdrojový kód. V následujícím dialogovém okně máme určit název výstupního souboru. Název je přednastaven a *není* doporučeno ho měnit.

A.4.3 Individuální generování zdrojového kódu

Pro generování zdrojového kódu za pomoci individuálního návrhu, klikneme v úvodním okně generátoru na tlačítko *Generate individual* (viz Obrázek A.24). Zobrazí se nám nové okno (viz Obrázek A.27), ve kterém zvolíme způsob vytváření komponenty (*Create and set component*) a úroveň do které budeme chtít rozhodovat o umístění prvků (*Components level*).



Obrázek A.27: Výběr způsobu vytváření komponent- individuální návrh

Components level:

Jako příklad zvolíme úroveň *Panel*. To znamená, že určíme místa, kde se budou vytvářet, nastavovat a přidávat panely. Komponenty umístěné na panelu se budou vytvářet, nastavovat a přidávat současně s ním.

Create and set komponent:

- *Together* – komponenta bude v jednom místě zdrojového kódu vytvořena i nastavena
- *Separately* – vytvoření a nastavení komponenty ve zdrojovém kódu může probíhat odděleně

V našem případě zvolíme možnost *Together* a pokračujeme dále kliknutím na tlačítko *Next* (viz Obrázek A.27).

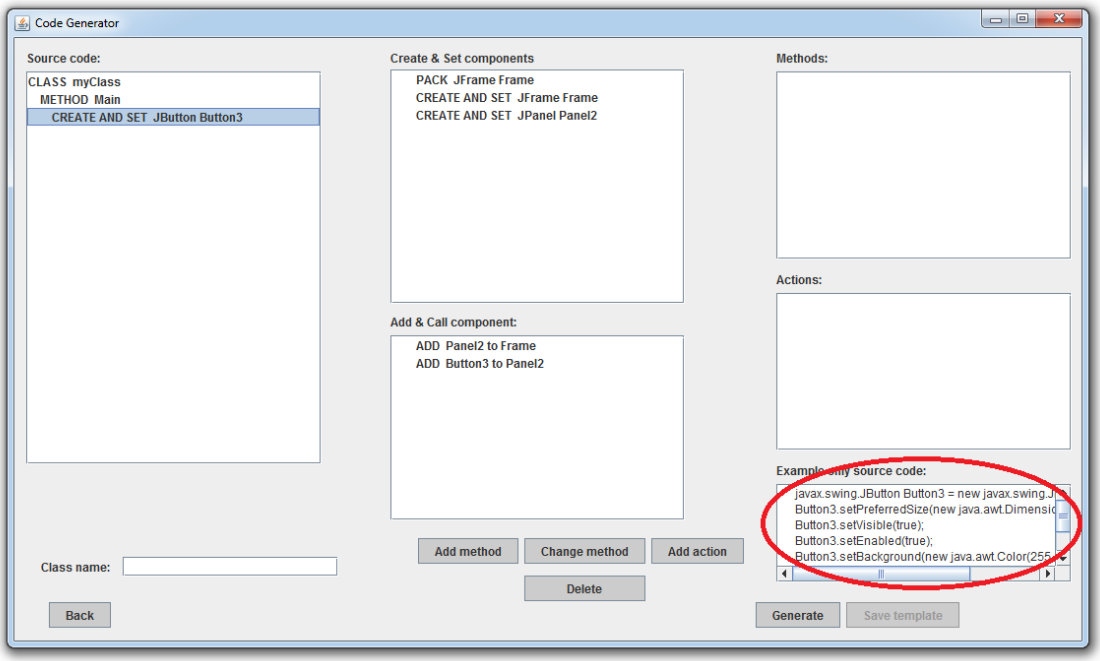
Nyní se zobrazí hlavní okno generátoru zdrojového kódu pro individuální návrh (viz Obrázek A.28).

Okno je tvořeno pomocí pěti seznamů komponent.

- *Source code* – tento seznam prvků představuje zdrojový kód. Přesně tak, jak jsou jednotlivé prvky v seznamu seřazeny, budou i vygenerovány ve zdrojovém kódu.
- *Create & Set components* – zde jsou uloženy části pro vytvoření a nastavení jednotlivých komponent.
- *Add & Call component* – v tomto seznamu jsou položky, které umožňují vložení jednotlivých komponent do panelů, menu a toolbaru. Zároveň jsou zde umístěny i položky pro volání metod a konstrukorů.

- *Methods* – obsahuje položky, které vytváří metody a konstruktory
- *Actions* – obsahuje položky, které jsou nutné pro nastavení akcí.

Mezi všemi seznamy (i uvnitř nich) je umožněn přesun prvku (nebo označené skupiny prvků). Stačí označit ukazatelem myši zvolený prvek, stisknout levé tlačítko myši a prvek přesunout do libovolného okna a na libovolnou pozici. Při individuálním návrhu je uživateli zobrazen i náhled zdrojového kódu. Zobrazuje se pouze pro jednu označenou položku v seznamu *Source code* (viz Obrázek A.28).



Obrázek A.28: Hlavní okno generátoru pro individuální návrh

V hlavním okně můžeme libovolně *mazat* zvolené položky. Stačí kliknutím myši označit položky a následně stisknout tlačítko *Delete*. Je však nutné upozornit na to, že jednou smazané *položky nelze vrátit zpět*.

Význam položek v seznamech je vysvětlen v kapitole A.4.6.

Jak tedy sestavit šablonu pro generování?

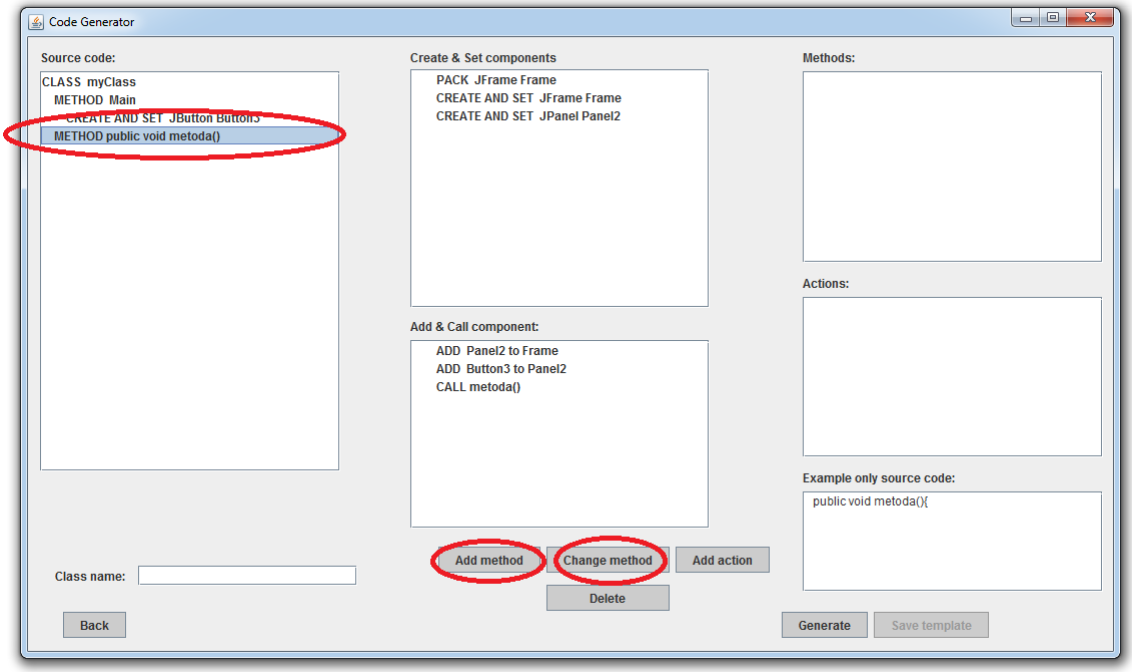
1) Vytvoříme vlastní akce, metody, atd.
2) Označíme jednotlivé prvky, které chceme vložit do výsledné šablony. Stiskneme levé tlačítko myši a prvky přesuneme na zvolenou pozici v seznamu s názvem *Source code*. Pokud chceme využít pro obsluhu akcí jednu společnou třídu, vložíme jí pouze jednou.
3) Vyplníme položku *Class name*, která udává název hlavní třídy. Ta bude při každém generování podle šablony vytvořena.
4) Pro vygenerování zdrojového kódu klikneme na tlačítko *Genrate*.
5) V nově otevřeném okně zvolíme cílovou složku pro umístění souboru se zdrojovým kódem, vyplníme název souboru a stiskneme tlačítko *OK*.

Poznámka: Není nutné využít všechny nabízené prvky.

Důležité upozornění: Pořadí prvků ve vytvářeném zdrojovém kódu není nijak validováno! Uživatel sám musí uvážit, zda jím navržený kód bude přeložitelný. Z toho plyne, že musíme vkládat položky do panelu ve správném pořadí, nikoli náhodně!

A.4.4 Vytvoření nebo editace metody

V případě, že chceme editovat již exitující metodu, označíme její název a klikneme na tlačítko *Change method*. Pro vytvoření nové metody klikáme na tlačítko *Add method* (viz Obrázek A.29). Přidání metody je umožněno jak při návrhu šablony tak, i při individuálním generování zdrojového kódu.

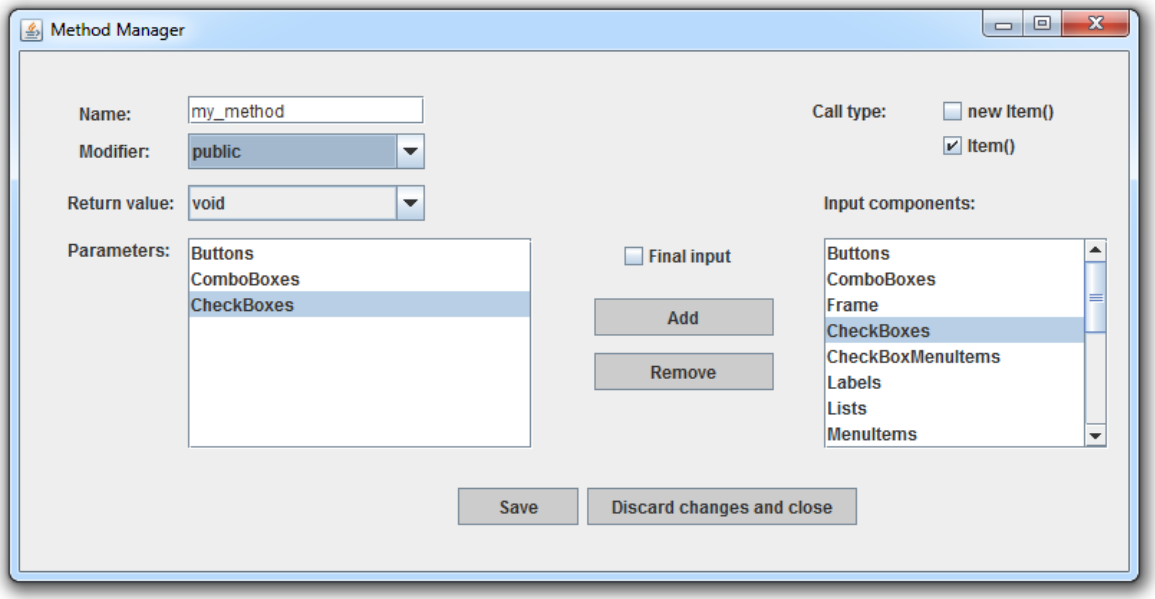


Obrázek A.29: Změna existující metody

Po kliknutí na tlačítko *Change method* nebo *Add method* dojde k zobrazení okna pro návrh metody (viz Obrázek A. 30).

V okně je možno nastavit tyto hodnoty:

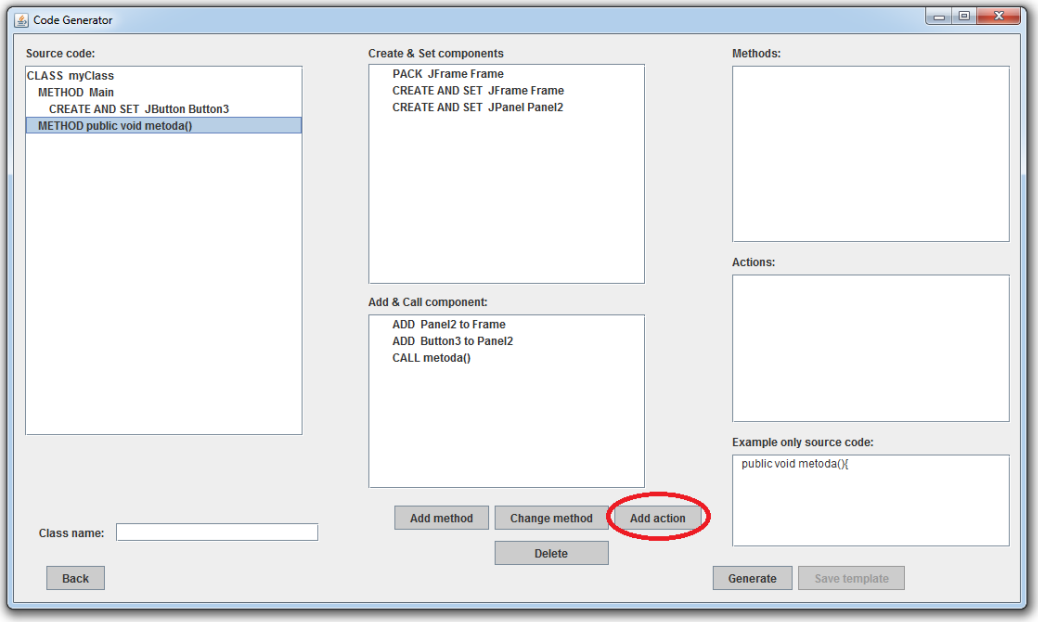
- Name – název metody
- Modifier – označení modifikátoru metody
- Return value – návratová hodnota metody
- Call type – dle zaškrtnuté varianty se buď vytvoří konstruktor anebo metoda
- Final input – zaškrtnutím nastavíme vkládaný parametr jako finální
- Parameters – vstupní parametry metody
    - V seznamu *Input components* zvolíme položku a klikneme na tlačítko *Add*. Nyní dojde k *přidání položky* do seznamu.
    - Obdobně můžeme v seznamu *Parameters* zvolit prvek a kliknout na tlačítko *Remove*. To zajistí *smazání položky*.
    - Pořadí parametrů je možné změnit tak, že označíme prvek, stiskneme levé tlačítko myši a prvek tahem přesuneme na zvolenou pozici.



Obrázek A. 30: Okno pro návrh metody

A.4.5 Vytvoření nové akce

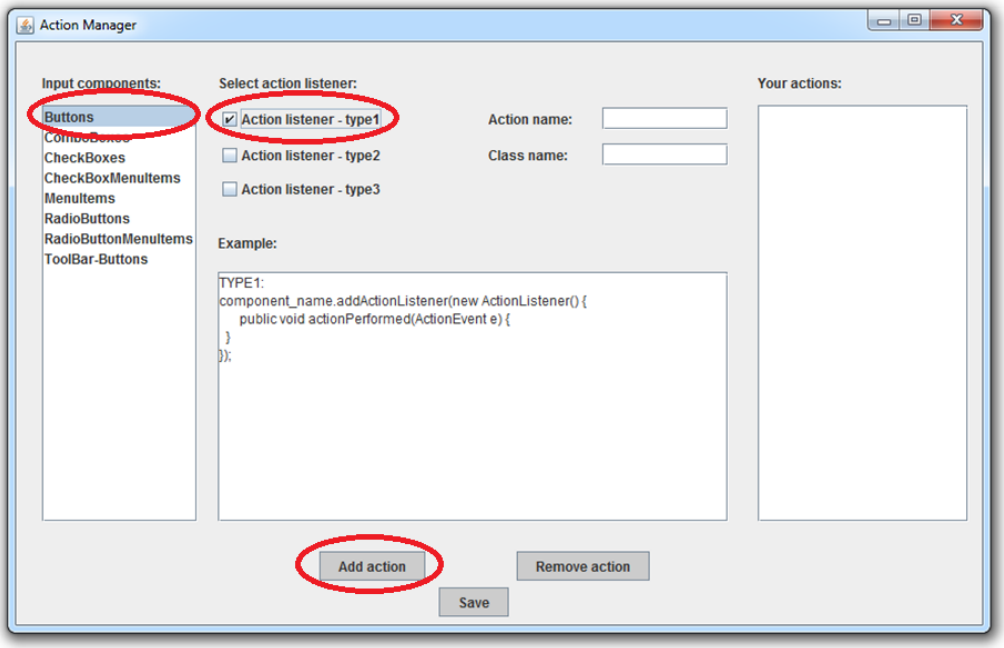
Přidání nové akce je umožněno kliknutím na tlačítko *Add Action* v hlavním okně generátoru (viz Obrázek A.31). Vytvoření nové akce je dostupné jak při návrhu šablony, tak i při individuálním generování zdrojového kódu.



Obrázek A.31: Vytvoření akce - umístění

Po kliknutí na tlačítko *Add Action*, dojde k zobrazení správce akcí (viz Obrázek A.32). Pro přidání akce musíme nejprve zvolit položku v seznamu *Input components*, pro kterou budeme akce generovat. Dále zvolíme *action listener* a případně vyplníme název akce a třídy (v části *Example se* zobrazí náhled zdrojového kódu). Nyní zbývá kliknout na tlačítko *Add action*. V seznamu s názvem *Your actions* budou zobrazeny právě vytvořené akce.

Pro odstranění akce ze seznamu, je nutné položku v seznamu *Your actions* označit a kliknout na tlačítko *Remove action.*



Obrázek A. 32: Okno pro vytvoření nové akce

A.4.6 Význam položek použitých v generátoru

Většinu názvů položek, se kterými se setkáme při práci s generátorem, můžeme rozdělit na dvě části. První část udává, co bude tato položka vytvářet (jejich význam je uveden v Tabulce A.2) a druhá část říká, pro co to bude vytvářet. První část je pro generování individuálním způsobem i pro vytváření šablony společná, druhá část se mění.

V případě vytváření šablony je určena celá skupina prvků (například buttons). Pro všechny tyto prvky se bude vytvářet zdrojový kód.

Při individuálním generování zdrojového kódu je uveden jeden konkrétní prvek (například panel7), pro který se bude vytvářet zdrojový kód.

Tabulka A. 2: Význam položek použitých v generátoru

Název položky	**Význam**
ACTION METHOD	`public void actionPerformed(ActionEvent e) { }`
ADD Panel2 to Frame	`Frame.add(Panel2);`
ADD ACTION LISTENER Button1	`Button1.addActionListener((ActionListener)this);`
ADD ALL CREATED GUI COMPONENTS	Vloží všechny prvky v GUI do jejich předka. Je zaručeno správné pořadí vložení prvků. Odpovídá opakovanému volání ADD.
ADD ALL CREATED MENU COMPONENTS	Vloží všechny prvky z menu do jejich předka. Je zaručeno správné pořadí vložení prvků. Odpovídá opakovanému volání ADD.
ADD ALL CREATED TOOLBARS COMPONENTS	Vloží všechny prvky v toolbaru do jejich předka Je zaručeno správné pořadí vložení prvků. Odpovídá opakovanému volání ADD.
ADD SIMPLE ACTIONLISTENER WITH METOD Button1	`Button1.addActionListener(new ActionListener() { public void actionPerformed(ActionEvent e) {} });`
CALL my_constr()	Zavolání metody nebo konstruktoru s názvem *my_ monstr* `new my_constr ();`
CLASS myClass	`public class myClass{`
CLASS WITH ACTION FOR COMPONENT ma_trida	`public class my_trida extends AbstractAction { public my_trida() { } public void actionPerformed(ActionEvent e){ } }`
CONSTRUCTOR public muj_nazev()	`public muj_nazev(){`

CREATE Button1	`JButton Button1;`
CREATE AND SET Button1	`JButton Button1 = new JButton();` `Button1.setText("Save");`
CREATE CLASS WITH ACTION nazev_tridy ma_akce	`nazev_tridy  ma_akce = new nazev_tridy();`
CREATESPECIAL buttongroups	Vytvoření všech prvků ve skupině ButtonGroups `ButtonGroup group = new ButtonGroup();`
METHOD Main	**`public static void`** `main(String[] args) {`
METHOD public void metoda (JFrame Frame1)	`public void metoda(JFrame Frame1){`
PACK Frame	`frame.pack();`
SET Button1	`Button1 = new JBButton();` `Button1.setText("Save");`
SET WITH ACTION Button1 with action ma_akce	`Button1 = new JBButton(ma_akce)` `Button1.setText("Save");`

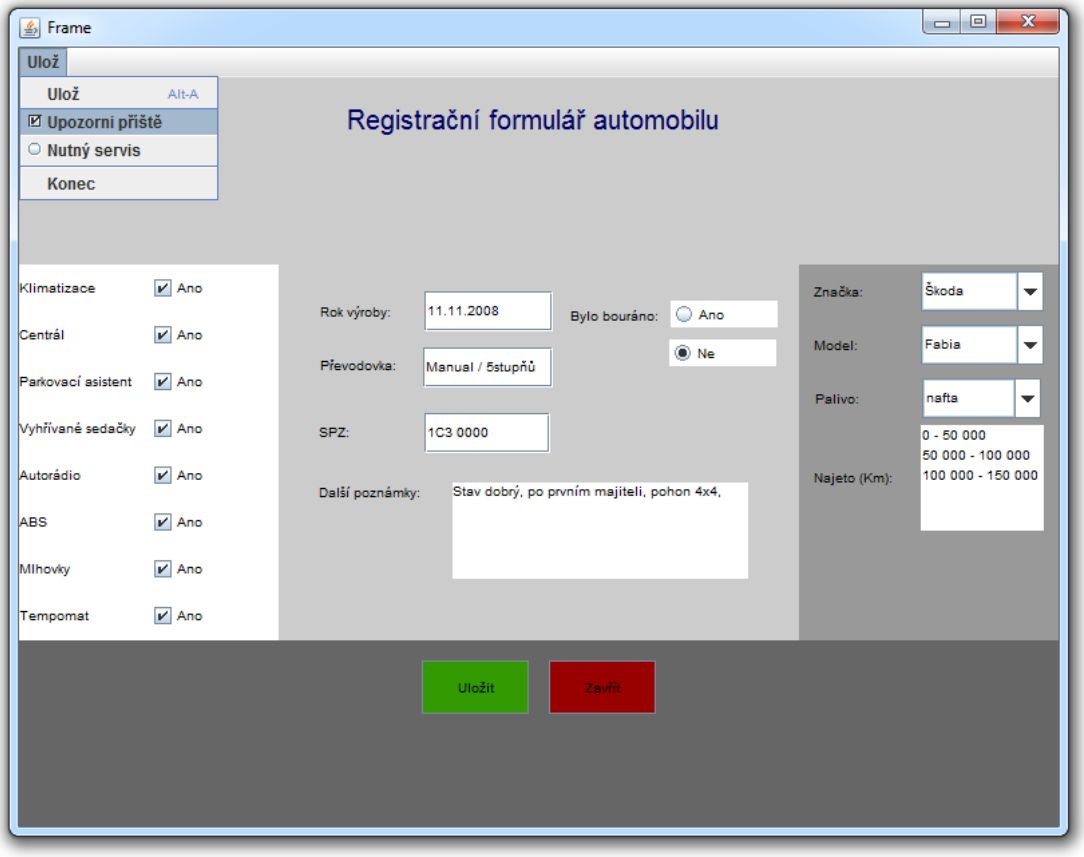
Příloha B Ukázka výstupního zdrojového kódu

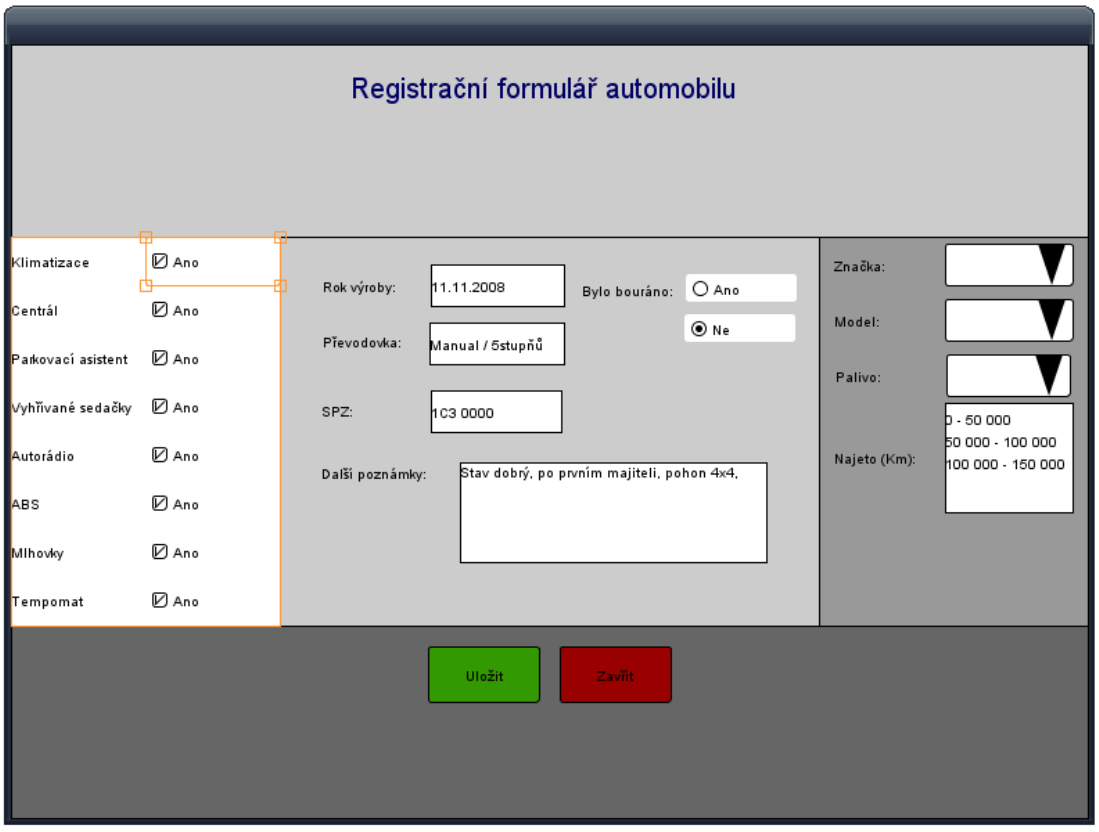
```
public class myClass {

         public  myClass(){

      javax.swing.JFrame Frame;

      javax.swing.JButton Button4;
      javax.swing.JButton Button3;

      javax.swing.JCheckBox Checkbox1;

      javax.swing.JPanel Panel9;
      javax.swing.JPanel Panel10;
      javax.swing.JComboBox ComboBox18;
      javax.swing.JPanel Panel11;
      javax.swing.JPanel Panel12;
      javax.swing.JRadioButton RadioButton1;
      javax.swing.JPanel Panel13;
      javax.swing.JLabel Label1;
      javax.swing.JLabel Label2;
      javax.swing.JPanel Panel14;

      class_name action_name = new class_name();

      Panel9 = new javax.swing.JPanel( new java.awt.BorderLayout());
      Panel10 = new javax.swing.JPanel( new
      java.awt.FlowLayout(java.awt.FlowLayout.LEFT, 0,0));
      Panel10.setBackground(new java.awt.Color(255, 153, 0));
      Panel10.setPreferredSize(new java.awt.Dimension(590,91));
      ComboBox18 = new javax.swing.JComboBox();
      ComboBox18.setPreferredSize(new java.awt.Dimension(265,35));
      ComboBox18.setVisible(true);
      ComboBox18.setEnabled(true);
      ComboBox18.setBackground(new java.awt.Color(255, 255, 255));
      ComboBox18.setForeground(new java.awt.Color(0, 0, 0));
      ComboBox18.setFont(new
      java.awt.Font("Arial",java.awt.Font.PLAIN,10));
      Panel11 = new javax.swing.JPanel(new
      java.awt.GridLayout(2,0,0,0));
      Panel11.setBackground(new java.awt.Color(204, 0, 153));
      Panel11.setPreferredSize(new java.awt.Dimension(147,183));
      Panel12 = new javax.swing.JPanel( new
      java.awt.FlowLayout(java.awt.FlowLayout.CENTER, 15,15));
      Panel12.setBackground(new java.awt.Color(255, 255, 255));
      Panel12.setPreferredSize(new java.awt.Dimension(296,183));
      RadioButton1 = new javax.swing.JRadioButton("RadioButton1");
      RadioButton1.setPreferredSize(new java.awt.Dimension(91,70));
      RadioButton1.setVisible(true);
      RadioButton1.setEnabled(true);
      RadioButton1.setBackground(new java.awt.Color(255, 255, 255));
      RadioButton1.setForeground(new java.awt.Color(0, 0, 0));
      RadioButton1.setFont(new
      java.awt.Font("Arial",java.awt.Font.PLAIN,10));
      RadioButton1.setSelected(true);
      Panel13 = new javax.swing.JPanel();
      Panel13.setLayout(new javax.swing.BoxLayout(Panel13,
      javax.swing.BoxLayout.Y_AXIS));
      Panel13.setBackground(new java.awt.Color(102, 255, 255));
```

```
Panel13.setPreferredSize(new java.awt.Dimension(147,183));
Label1 = new javax.swing.JLabel("Label1");
Label1.setPreferredSize(new java.awt.Dimension(43,21));
Label1.setVisible(true);
Label1.setEnabled(true);
Label1.setBackground(new java.awt.Color(255, 255, 255));
Label1.setForeground(new java.awt.Color(0, 0, 0));
Label1.setFont(new
java.awt.Font("Arial",java.awt.Font.PLAIN,10));
Label1.setAlignmentX(java.awt.Component.LEFT_ALIGNMENT);
Label2 = new javax.swing.JLabel("Label2");
Label2.setPreferredSize(new java.awt.Dimension(43,21));
Label2.setVisible(true);
Label2.setEnabled(true);
Label2.setBackground(new java.awt.Color(255, 255, 255));
Label2.setForeground(new java.awt.Color(0, 0, 0));
Label2.setFont(new
java.awt.Font("Arial",java.awt.Font.PLAIN,10));
Label2.setAlignmentX(java.awt.Component.LEFT_ALIGNMENT);
Panel14 = new javax.swing.JPanel( new
java.awt.FlowLayout(java.awt.FlowLayout.RIGHT, 0,0));
Panel14.setBackground(new java.awt.Color(51, 204, 0));
Panel14.setPreferredSize(new java.awt.Dimension(590,91));

Frame = new javax.swing.JFrame("Frame");
Frame.setDefaultCloseOperation
(javax.swing.JFrame.EXIT_ON_CLOSE);
Frame.setVisible(true);
Frame.setSize(new java.awt.Dimension(600, 400));

Button4 = new javax.swing.JButton(action_name);
Button4.setPreferredSize(new java.awt.Dimension(147,91));
Button4.setVisible(true);
Button4.setEnabled(true);
Button4.setBackground(new java.awt.Color(153, 153, 0));
Button4.setForeground(new java.awt.Color(0, 0, 0));
Button4.setFont(new
java.awt.Font("Arial",java.awt.Font.PLAIN,10));
Button4.setText("tlacitko2");
Button3 = new javax.swing.JButton(action_name);
Button3.setPreferredSize(new java.awt.Dimension(147,91));
Button3.setVisible(true);
Button3.setEnabled(true);
Button3.setBackground(new java.awt.Color(255, 255, 255));
Button3.setForeground(new java.awt.Color(0, 0, 0));
Button3.setFont(new
java.awt.Font("Arial",java.awt.Font.PLAIN,10));
Button3.setText("tlacitko1");

Checkbox1 = new javax.swing.JCheckBox(action_name);
Checkbox1.setPreferredSize(new java.awt.Dimension(80,20));
Checkbox1.setVisible(true);
Checkbox1.setEnabled(true);
Checkbox1.setBackground(new java.awt.Color(255, 255, 255));
Checkbox1.setForeground(new java.awt.Color(0, 0, 0));
Checkbox1.setFont(new
java.awt.Font("Arial",java.awt.Font.PLAIN,10));
Checkbox1.setSelected(true);
Checkbox1.setText("CheckBox1");
```

```
        Frame.add(Panel9);
        Panel9.add(Panel10, java.awt.BorderLayout.NORTH);
        Panel10.add(ComboBox18);
        Panel9.add(Panel11, java.awt.BorderLayout.WEST);
        Panel11.add(Button4);
        Panel11.add(Button3);
        Panel9.add(Panel12, java.awt.BorderLayout.CENTER);
        Panel12.add(Checkbox1);
        Panel12.add(RadioButton1);
        Panel9.add(Panel13, java.awt.BorderLayout.EAST);
        Panel13.add(Label1);
        Panel13.add(Label2);
        Panel9.add(Panel14, java.awt.BorderLayout.SOUTH);

        Frame.pack();

        }
    public static void main(String[] args) {

            new myClass();

        }
}class class_name extends javax.swing.AbstractAction {
    public  class_name() {
    }
    @Override
    public void actionPerformed(java.awt.event.ActionEvent e) {
    }
}
```

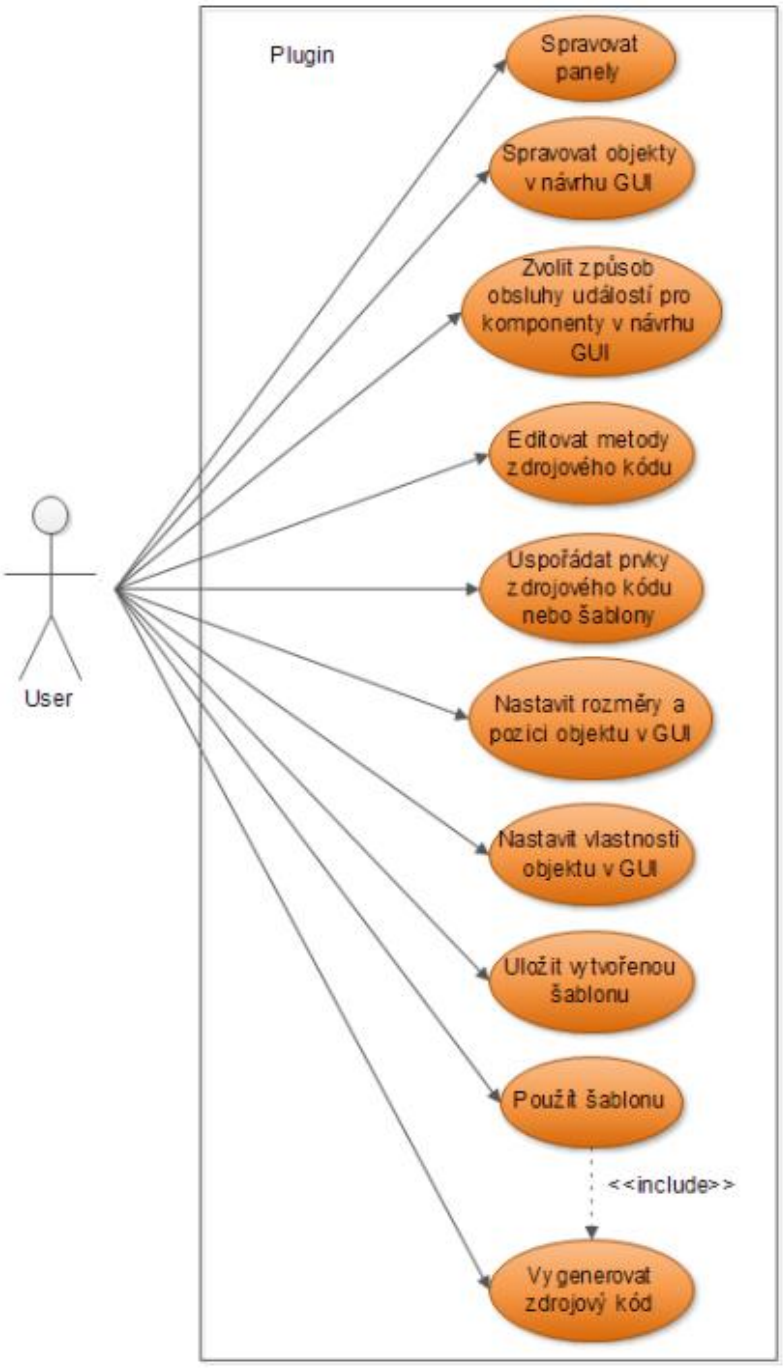
Příloha C Ukázka vygenerovaného GUI



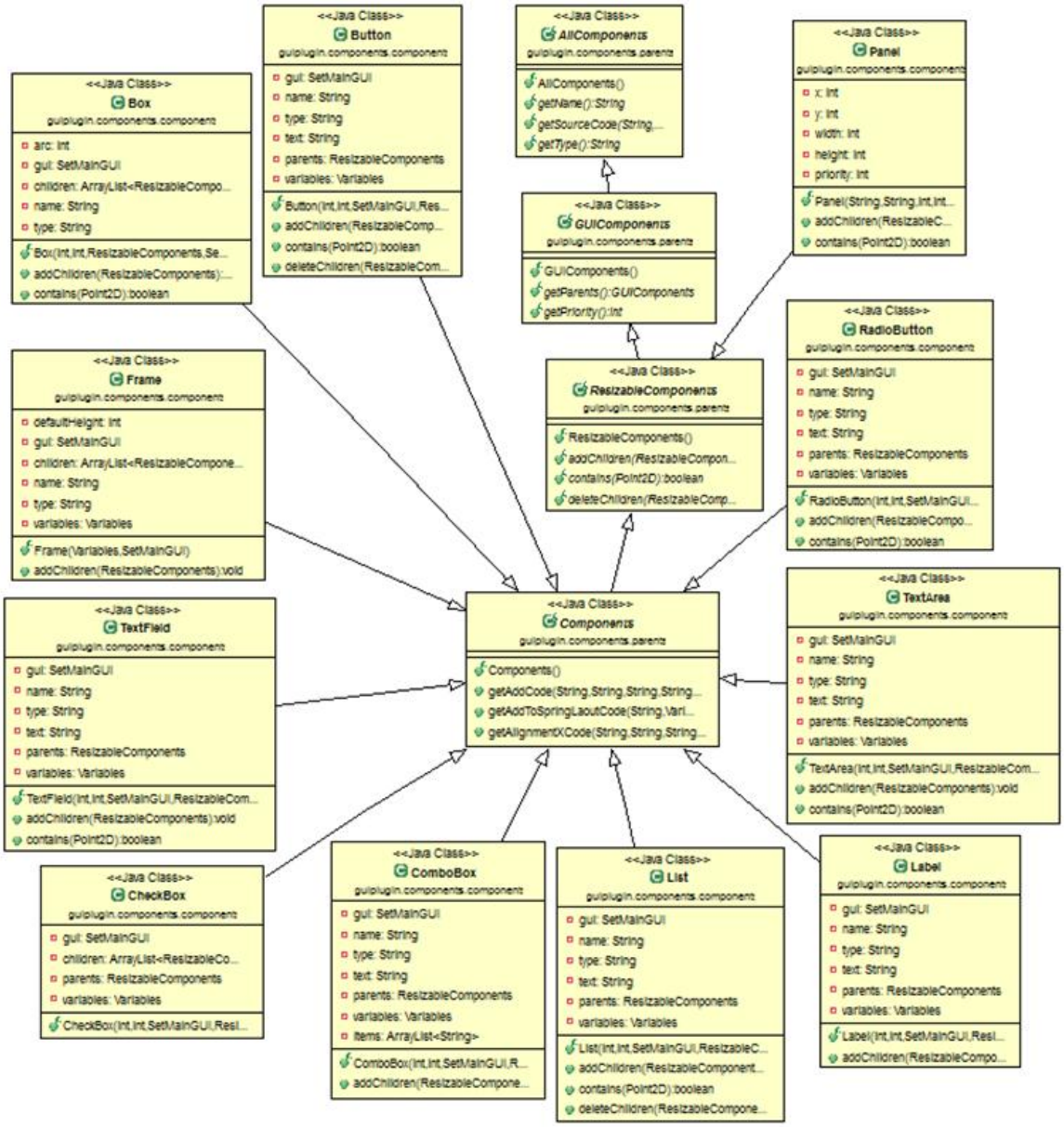


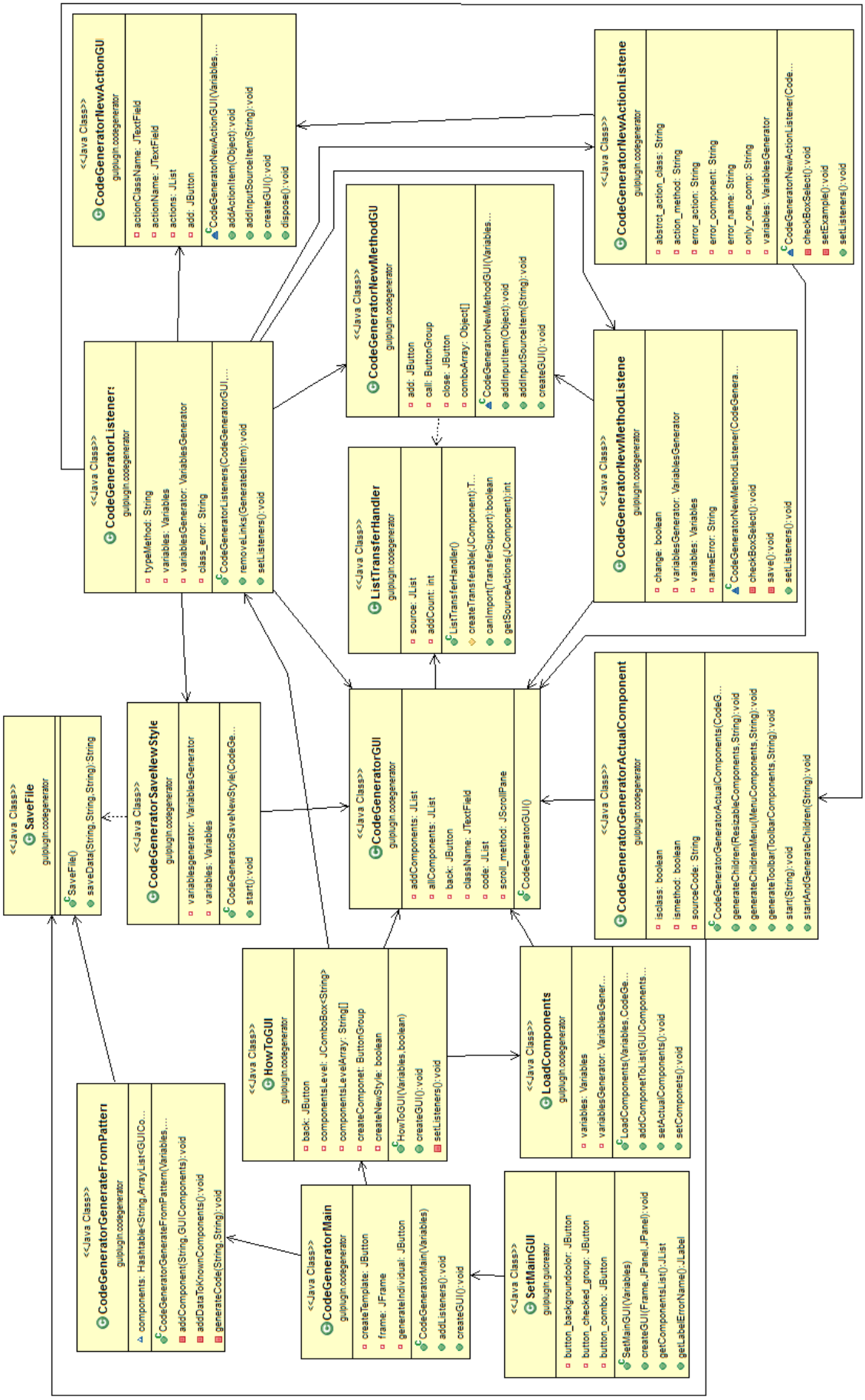
Příloha D UML diagramy

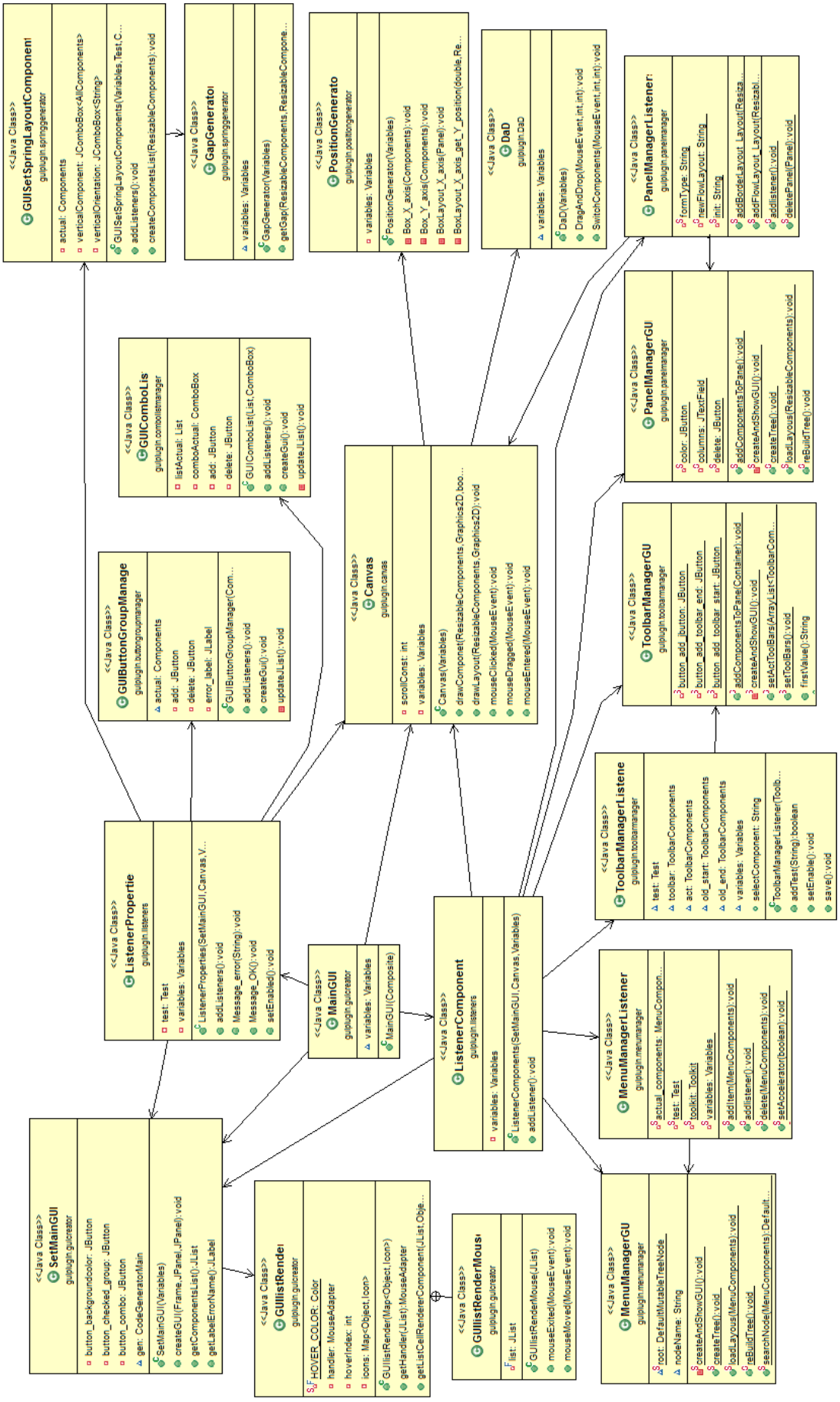
D.1 Diagram případů užití

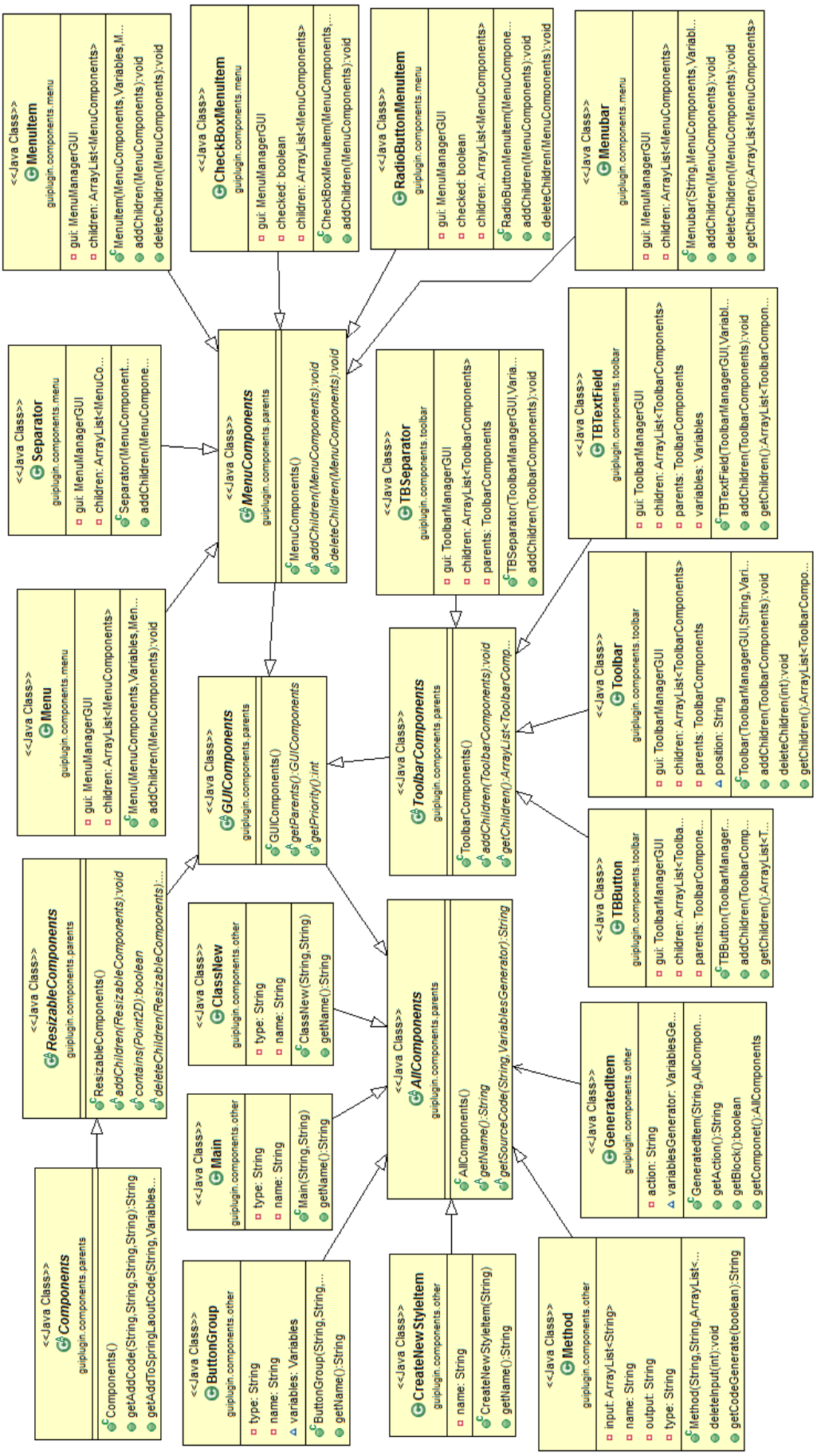


D.2 Diagramy tříd




<<Java Class>>
CodeGeneratorNewActionGUI
gulplugin.codegenerator
actionClassName: JTextField
actionName: JTextField
actions: JList
add: JButton
CodeGeneratorNewActionGUI(Variables,...
addActionItem(Object):void
addInputSourceItem(String):void
createGUI():void
dispose():void
<<Java Class>>
CodeGeneratorNewActionListene
gulplugin.codegenerator
abstract_action_class: String
action_method: String
error_action: String
error_component: String
error_name: String
only_one_comp: String
variables: VariablesGenerator
CodeGeneratorNewActionListener(Code...
checkBoxSelect():void
setExample():void
setListeners():void
<<Java Class>>
CodeGeneratorListener
gulplugin.codegenerator
typeMethod: String
variables: Variables
variablesGenerator: VariablesGenerator
class_error: String
CodeGeneratorListeners(CodeGeneratorGUI,...
removeLinks(GeneratedItem):void
setListeners():void
<<Java Class>>
CodeGeneratorNewMethodGU
gulplugin.codegenerator
add: JButton
call: ButtonGroup
close: JButton
comboArray: Object[]
CodeGeneratorNewMethodGUI(Variables...
addInputItem(Object):void
addInputSourceItem(String):void
createGUI():void
<<Java Class>>
CodeGeneratorNewMethodListene
gulplugin.codegenerator
change: boolean
variablesGenerator: VariablesGenerator
variables: Variables
nameError: String
CodeGeneratorNewMethodListener(CodeGenera...
checkBoxSelect():void
save():void
setListeners():void
<<Java Class>>
ListTransferHandler
gulplugin.codegenerator
source: JList
addCount: int
ListTransferHandler()
createTransferable(JComponent):T...
canImport(TransferSupport):boolean
getSourceActions(JComponent):int
<<Java Class>>
SaveFile
gulplugin.codegenerator
SaveFile()
saveData(String,String,String):String
<<Java Class>>
CodeGeneratorSaveNewStyle
gulplugin.codegenerator
variablesgenerator: VariablesGenerator
variables: Variables
CodeGeneratorSaveNewStyle(CodeGe...
start():void
<<Java Class>>
CodeGeneratorGUI
gulplugin.codegenerator
addComponents: JList
allComponents: JList
back: JButton
className: JTextField
code: JList
scroll_method: JScrollPane
CodeGeneratorGUI()
<<Java Class>>
CodeGeneratorGeneratorActualComponent
gulplugin.codegenerator
isclass: boolean
ismethod: boolean
sourceCode: String
CodeGeneratorGeneratorActualComponents(CodeG...
generateChildren(ResizableComponents,String):void
generateChildrenMenu(MenuComponents,String):void
generateToolbar(ToolbarComponents,String):void
start(String):void
startAndGenerateChildren(String):void
<<Java Class>>
CodeGeneratorGenerateFromPattern
gulplugin.codegenerator
components: Hashtable<String,ArrayList<GUICo...
CodeGeneratorGenerateFromPattern(Variables,...
addComponent(String,GUIComponents):void
addDataToKnownComponents():void
generateCode(String,String):void
<<Java Class>>
HowToGUI
gulplugin.codegenerator
back: JButton
componentsLevel: JComboBox<String>
componentsLevelArray: String[]
createComponet: ButtonGroup
createNewStyle: boolean
HowToGUI(Variables,boolean)
createGUI():void
setListeners():void
<<Java Class>>
LoadComponents
gulplugin.codegenerator
variables: Variables
variablesGenerator: VariablesGener...
LoadComponents(Variables,CodeGe...
addComponetToList(GUIComponents...
setActualComponents():void
setComponets():void
<<Java Class>>
CodeGeneratorMain
gulplugin.codegenerator
createTemplate: JButton
frame: JFrame
generateIndividual: JButton
CodeGeneratorMain(Variables)
addListeners():void
createGUI():void
<<Java Class>>
SetMainGUI
gulplugin.guicreator
button_backgroundcolor: JButton
button_checked_group: JButton
button_combo: JButton
SetMainGUI(Variables)
createGUI(Frame,JPanel,JPanel):void
getComponentsList():JList
getLabelErrorName():JLabel


GUISetSpringLayoutComponent
GapGenerator
PositionGenerato
DaD
PanelManagerListener
GUIComboLis
PanelManagerGUI
Canvas
ToolbarManagerGU
ListenerPropertie
MainGUI
ToolbarManagerListene
ListenerComponent
MenuManagerListener
SetMainGUI
GUIListRende
GUIListRenderMous
MenuManagerGU


<<Java Class>>
MenuItem
guiplugin.components.menu
gui: MenuManagerGUI
children: ArrayList<MenuComponents>
MenuItem(MenuComponents,Variables,M...
addChildren(MenuComponents):void
deleteChildren(MenuComponents):void
<<Java Class>>
CheckBoxMenuItem
guiplugin.components.menu
gui: MenuManagerGUI
checked: boolean
children: ArrayList<MenuComponents>
CheckBoxMenuItem(MenuComponents,...
addChildren(MenuComponents):void
<<Java Class>>
RadioButtonMenuItem
guiplugin.components.menu
gui: MenuManagerGUI
checked: boolean
children: ArrayList<MenuComponents>
RadioButtonMenuItem(MenuCompone...
addChildren(MenuComponents):void
deleteChildren(MenuComponents):void
<<Java Class>>
Menubar
guiplugin.components.menu
gui: MenuManagerGUI
children: ArrayList<MenuComponents>
Menubar(String,MenuComponents,Variabl...
addChildren(MenuComponents):void
deleteChildren(MenuComponents):void
getChildren():ArrayList<MenuComponents>
<<Java Class>>
Separator
guiplugin.components.menu
gui: MenuManagerGUI
children: ArrayList<MenuCo...
Separator(MenuComponent...
addChildren(MenuCompone...
<<Java Class>>
MenuComponents
guiplugin.components.parents
MenuComponents()
addChildren(MenuComponents):void
deleteChildren(MenuComponents):void
<<Java Class>>
TBSeparator
guiplugin.components.toolbar
gui: ToolbarManagerGUI
children: ArrayList<ToolbarComponents>
parents: ToolbarComponents
TBSeparator(ToolbarManagerGUI,Varia...
addChildren(ToolbarComponents):void
<<Java Class>>
TBTextField
guiplugin.components.toolbar
gui: ToolbarManagerGUI
children: ArrayList<ToolbarComponents>
parents: ToolbarComponents
variables: Variables
TBTextField(ToolbarManagerGUI,Variabl...
addChildren(ToolbarComponents):void
getChildren():ArrayList<ToolbarCompon...
<<Java Class>>
Menu
guiplugin.components.menu
gui: MenuManagerGUI
children: ArrayList<MenuComponents>
Menu(MenuComponents,Variables,Men...
addChildren(MenuComponents):void
<<Java Class>>
GUIComponents
guiplugin.components.parents
GUIComponents()
getParents():GUIComponents
getPriority():int
<<Java Class>>
ToolbarComponents
guiplugin.components.parents
ToolbarComponents()
addChildren(ToolbarComponents):void
getChildren():ArrayList<ToolbarComp...
<<Java Class>>
Toolbar
guiplugin.components.toolbar
gui: ToolbarManagerGUI
children: ArrayList<ToolbarComponents>
parents: ToolbarComponents
position: String
Toolbar(ToolbarManagerGUI,String,Vari...
addChildren(ToolbarComponents):void
deleteChildren(int):void
getChildren():ArrayList<ToolbarCompo...
<<Java Class>>
ResizableComponents
guiplugin.components.parents
ResizableComponents()
addChildren(ResizableComponents):void
contains(Point2D):boolean
deleteChildren(ResizableComponents):...
<<Java Class>>
ClassNew
guiplugin.components.other
type: String
name: String
ClassNew(String,String)
getName():String
<<Java Class>>
AllComponents
guiplugin.components.parents
AllComponents()
getName():String
getSourceCode(String,VariablesGenerator):String
<<Java Class>>
TBButton
guiplugin.components.toolbar
gui: ToolbarManagerGUI
children: ArrayList<Toolba...
parents: ToolbarCompone...
TBButton(ToolbarManager...
addChildren(ToolbarComp...
getChildren():ArrayList<T...
<<Java Class>>
Components
guiplugin.components.parents
Components()
getAddCode(String,String,String,String):String
getAddToSpringLaoutCode(String,Variables...
<<Java Class>>
Main
guiplugin.components.other
type: String
name: String
Main(String,String)
getName():String
<<Java Class>>
GeneratedItem
guiplugin.components.other
action: String
variablesGenerator: VariablesGe...
GeneratedItem(String,AllCompon...
getAction():String
getBlock():boolean
getComponet():AllComponents
<<Java Class>>
ButtonGroup
guiplugin.components.other
type: String
name: String
variables: Variables
ButtonGroup(String,String,...
getName():String
<<Java Class>>
CreateNewStyleItem
guiplugin.components.other
name: String
CreateNewStyleItem(String)
getName():String
<<Java Class>>
Method
guiplugin.components.other
input: ArrayList<String>
name: String
output: String
type: String
Method(String,String,ArrayList<...
deleteInput(int):void
getCodeGenerate(boolean):String

Příloha E Obsah přiloženého DVD

Přiložené DVD obsahuje instalaci Eclipse IDE (verze Eclipse Juno Service Release 2 pro Windows 64bit) s importovaným pluginem, vlastní plugin a písemnou část bakalářské práce. Struktura přiloženého CD vypadá takto:

- **src** – složka obsahuje všechny zdrojové soubory.
- **plugin** – složka obsahuje vytvořený plugin.
- **eclipse** – složka s instalací Eclipse.
- **diplomová_prace.pdf** – dokument s diplomovou prací.