Vysoká škola ekonomická v Praze

Fakulta informatiky a statistiky

Katedra informačních technologií

Student	: Tatsiana Pusiankova
Vedoucí bakalářské práce	: Ing. Alena Buchalcevová, Ph.D.
Oponent bakalářské práce	: RNDr. Vladimír Tichý

TÉMA BAKALÁŘSKÉ PRÁCE

NetBeans GUI Builder

Prohlášení

Prohlašuji, že jsem bakalářskou práci zpracoval(a) samostatně a že jsem uvedl(a) všechny použité prameny a literaturu, ze kterých jsem čerpal(a).

V Praze dne .....

podpis

Abstrakt:

Cílem této práce je přiblížit čtenáři možnosti tvorby a úpravy grafického rozhraní k aplikaci vyvíjené v programovacím prostředí jazyka Java NetBeans a to za pomocí nástroje NetBeans IDE GUI Builder. Ve výsledku by pak měl být i čtenář, který přijde do styku s tímto nástrojem poprvé, schopen tvorby vlastního grafického prostředí.
Práce obsahuje teoretický popis nástroje NetBeans IDE GUI Builder, včetně jeho nejdůležitějších vlastností a charakteristik, a také řadu praktických návodů na užití, které pomohou čtenářovi ve vytvoření svého prvního grafického uživatelského rozhraní.
Po seznámení se samotným prostředím tohoto nástroje jsou čtenáři přiblíženy jeho jednotlivé prvky, komponenty, jejich význam a využití. Dále jsou již popsány konkrétní komponenty jednotlivých prvků, jejich význam, funkce a u stěžejních komponent pro názornost i jejich praktický příklad. Práce také zahrnuje popis základních postupů při využití nástroje NetBeans IDE GUI Builder a proto může sloužit jako návod k použití pro začínajícího uživatele

Annotation:

This work aims at making readers familiar with the powerful tool NetBeans IDE GUI Builder and helping them make their first steps to creation of their own graphical user interface in the Java programming language.
The work includes theoretical description of NetBeans IDE GUI Builder, its most important characteristics and peculiarities and also a set of practical instructions that will help readers in creation of their first GUI.
The readers will be introduced to the environment of this tool in general and to its main elements in particular. The work also contains description of standalone components and their functions and use. The description of the complicated components also includes instructions of their application in practice. The work also includes the description of the main steps during use of NetBeans IDE GUI Builder and can be used as a user manual for a beginning user.

Obsah

1. Úvod

Bakalářská práce na téma **NetBeans GUI Builder** je věnována velice zajímavé a důležité součásti platformy NetBeans IDE – nástroji GUI Builder, což je speciální nástroj, určený pro pohodlnou a snadnou tvorbu grafického uživatelského rozhraní v jazyce Java. Popisovaný nástroj je založen na grafickém vzhledu a není určen pro přímou tvorbu zdrojového kódu, což usnadňuje jeho snadné, pohodlné a velice efektivní užití.

Důvodem výběru tohoto tématu pro zpracování v rámci bakalářské práce byl jednak osobní zájem autorky o tento velice silný a pohodlný nástroj snadného programování (po absolvování předmětu Základy softwarového inženýrství), a jednak absence úplného strukturovaného návodu k tomuto rozhraní nejen v češtině, ale i v angličtině. Dostupné informace o nástroji NetBeans IDE GUI Builder jsou rozptýleny na více místech, což komplikuje jejich použití a občas i vyhledávání uživatelem. Tato práce proto dává tyto informace dohromady a je tak určena pro uživatele, kteří začínají pracovat s nástrojem NetBeans IDE GUI Builder a nechtějí zkoumat velké množství elektronických zdrojů již pro pochopení základních postupů.

Cílem této práce je poskytnout čtenáři teoretické informace o vlastnostech a možnostech rozhraní NetBeans IDE GUI Builder a také návod na jeho praktické užití při programování jednoduchých grafických aplikací. Tato práce může sloužit i jako pomůcka ke studiu předmětu Základy softwarového inženýrství v části tvorby grafického uživatelského rozhraní. Poskytuje čtenáři možnost nejen se naučit provádět určité operace, ale také pochopit principy a souvislosti, aplikované při práci s komponentami tvořené aplikace v rámci nástroje GUI Builder, čímž nakonec umožňuje intuitivní využití i těch vlastností a funkcí, které nejsou explicitně v této práci uvedeny. Práce je určena jak k obecnému seznámení s nástrojem NetBeans IDE GUI Builder, tak i pro „konzultační“ účely při vzniku potíží s použitím tohoto nástroje.

Tato práce nepokrývá veškeré možnosti nástroje NetBeans IDE GUI Builder vzhledem k jeho rozsáhlosti a rozmanitosti, např. zde jsou zcela vynechány popisy použití komponent Java Persistence a práce s databází, tato oblast již patří k pokročilé úrovni využití NetBeans IDE a i právě proto není zahrnuta ani do obsahu předmětu Základy softwarového inženýrství.

Předpokladem práce je, že má čtenář alespoň základní znalosti jazyka Java a také jeho principů. Nicméně pochopení práce nevyžaduje od čtenářů nikterak hluboké znalosti programování, jelikož NetBeans IDE GUI Builder poskytuje možnost vyhnout se „manuálnímu“ programování, což určitě ocení dost velká skupina čtenářů.

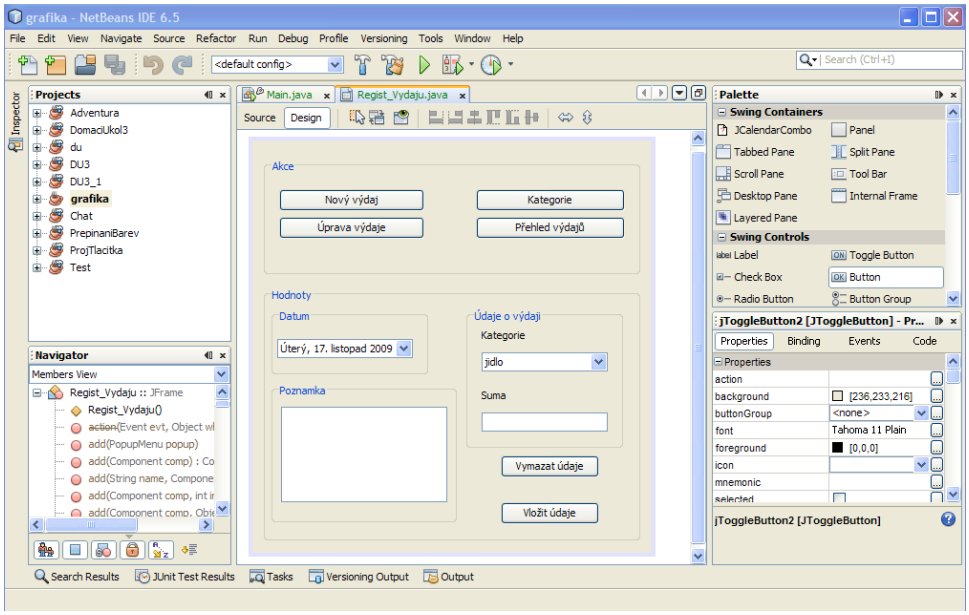
Práce je vytvořena ve formě popisu-návodu na použití a skládá se z teoretické a praktické části. Teoretická část zahrnuje druhou a třetí kapitoly, praktické aplikaci je věnovaná kapitola čtvrtá. Druha kapitola obsahuje obecný popis nejdůležitějších vlastností a konceptu nástroje NetBeans IDE GUI Builder. Třetí kapitola zahrnuje popis hlavního okna tohoto nástroje, možností poskytovaných tímto oknem, a také podrobný popis komponent, používaných při tvorbě grafického uživatelského rozhraní (včetně návodů na práci s neviditelnými komponentami). Čtvrtá kapitola popisuje proces tvorby grafického uživatelského rozhraní pomocí nástroje NetBeans IDE GUI Builder: vytvoření kontejnerů GUI, vkládání komponent na formulář, vkládání obrázku na plátno, výběr komponent ve formuláři, zarovnání komponent pomocí vodících čar, ukazatelů ukotvení a rozměrových ukazatelů. Dále čtvrtá kapitola zahrnuje popis vlastností komponent a jejich aplikace, způsoby změny vlastností komponent, uvázání komponent grafického rozhraní a kódu, spojení komponent, a také rozmístění a spuštění aplikace. Pro názornost většina instrukcí a návodů disponuje obrázky, které pomáhají pochopit určitý krok návodu.

Při vypracování aplikace autorka použila verzi NetBeans IDE 6.5.

2. Seznámení s NetBeans GUI Builder

NetBeans GUI Builder (dříve **Matisse**) je součástí platformy Java, která umožňuje pohodlné vytvoření grafického uživatelského rozhraní na vysoké úrovni bez důrazu na psaní kódu. Za pomoci nástroje NetBeans GUI Builder se lze snadno vyhnout nejen manuálnímu programování, ale i složitým postupům při aplikaci komplikovaného Swing API, obtížím spojeným se změnou velikosti a zarovnáním komponent, často nepředvídatelnému chování rozhraní na různých platformách OS, a také použití složitých a rozmanitých správců rozvržení, značně komplikujících tvorbu rozhraní. Veškeré tyto problémy budou pomocí nástroje NetBeans GUI Builder vyřešeny automaticky. Uživatel[1] také není nucen naučit se přímo pracovat se správcem rozvržení, což ve výsledku ušetří mnoho času a neodradí předem uživatele od tvorby GUI rozhraní. Během návrhu jej totiž GUI Builder překládá do funkčního kódu a případně také mění tento kód podle změny návrhu. Tento proces není pro uživatele viditelný a uživatel jej ani nemůže nikterak ovlivnit, tím pádem se uživatel může zcela soustředit na návrh aplikace a její funkčnost. Uživatelovi pak zbývá jen zvolit potřebné komponenty z předem připravené nabídky, umístit je na požadovaná místa na obrazovce, spojit je s kódem, který bude řídit chování každé komponenty a užívat si výsledek své práce.

Na Obr. 1 je představen obecný screenshot NetBeans IDE 6.5.



Obr. 1 NetBeans IDE 6.5

[1] Zde a dále v textu se pod pojmem „uživatel“ rozumí uživatel NetBeans IDE

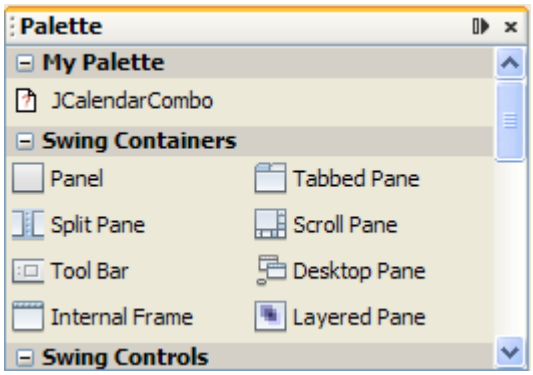
Nejdůležitější vlastnosti NetBeans GUI Builder:

- **Možnost snadného vytvoření profesionálního Swing GUI**[2] - Swing GUI se vytváří jednoduše umístěním (přetahováním) potřebných komponent z palety na plátno (do oblasti návrhu do potřebného místa, přičemž programováni odstupu, zarovnání a ukotvení v rámci rozvržení se koná automaticky. Funkcionalita komponent formuláře se zde do velké míry zabezpečuje skrze rozhraní IDE, ačkoliv existuje i možnost editace kódu - buďto přímo manuálně, nebo za použití speciálních okének a polí, rovněž poskytovaných v rámci NetBeans GUI Builder

- **Přizpůsobitelnost a intuitivní ovládání** – základní ovládání nástroje NetBeans GUI Builder je velmi pohodlné, postupy práce s komponentami jsou velmi intuitivní, ačkoliv pokročilejší práce již vyžaduje od uživatele předcházející přípravu a studium dokumentace nebo nápovědy.

- **Široké možnosti použití standardních a vlastních komponent GUI –** NetBeans GUI Builder nabízí uživatelovi veškeré běžně používané standardní elementy palet Swing, AWT, Beans[3] a komponent Java Persistence, a krom toho také možnost tvorby a instalace vlastních komponent za pomoci **Palette Manager** (zobrazuje se kliknutím pravého tlačítka myši v jakékoliv části palety a volbou odpovídající možnosti). Uživatel si může také vytvořit vlastní kategorie komponent, a po vytvoření (instalaci) komponenty ji může používat stejně jako standardní komponenty, které jsou součástí palety. Vlastní komponenty mohou být dodané z libovolného zdroje ve formě externí JAR složky, z již vytvořeného projektu NetBeans IDE, nebo z knihovny, nainstalované v IDE[4]. Na Obr. 2 je zobrazena vlastní komponenta JCalendarCombo v nově vytvořené kategorii My Palette[5] .

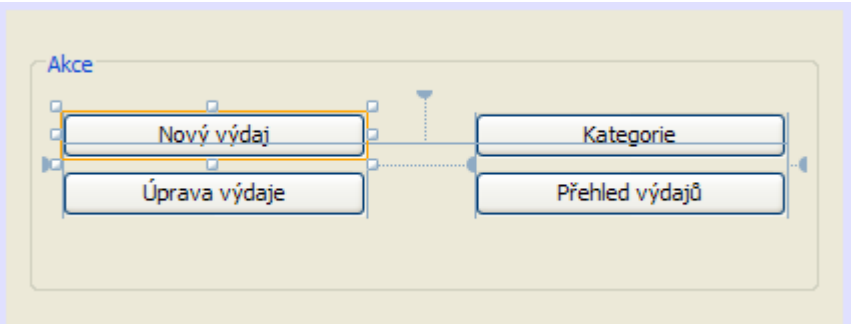
[2] SWING a AWT – obě palety jsou součásti NetBeans GUI Builder
[3] Zvláštní programové komponenty, které mohou být vytvořeny programátorem pro použití ve složitých komponentách, appletů, servletů, atd.
[4] Dodání jak vlastní komponenty, tak i vlastní kategorie je velice snadné: pro dodání vlastní kategorie použijte možnost **Create New Category**, pro dodáni vlastní komponenty – **Palette Manager->Add from JAR/Add from Library/Add from Project**.
[5] Použitá komponenta JCalendarCombo byla získána na stránce http://flib.sourceforge.net/JCalendar/doc/index.html



Obr. 2 Vlastní komponenta ve zvláštní kategorii My palette[6]

- **Volný režim návrhu –** koncepce **Free Design** je tvořena kombinací správce rozvržení GroupLayout a samotného nástroje NetBeans IDE GUI Builder. Použitím tohoto režimu může uživatel volně měnit pozice komponent použitých v rámci plátna bez ohledu na rozvržení – nízkoúrovňové aspekty velikostí, umístění, zarovnání a ukotvení komponent jsou řešeny automaticky pomoci správce rozvržení GroupLayout. Toto rozvržení nevyužívá žádné absolutní pozice, místo toho rozhraní udržuje ustanovené relace mezi komponenty. Jestli-že uživatel mění umístění komponent v náhledovém poli NetBeans, aplikace navrhuje kotvy, čáry zarovnání a nápovědy ke změně velikosti, popřípadě pro přemisťované komponenty navrhne aplikace automaticky polohu s cílem ustanovení výhodnější relace. Tvorba a změna zdrojového kódu podle akce uživatele probíhá automaticky na pozadí a není pro uživatele viditelná.

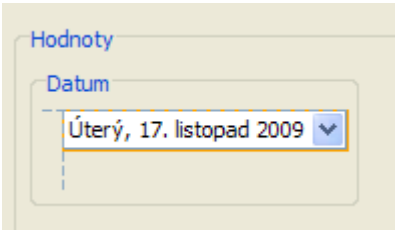
  Režim **Free Design** je standardně aktivní při vytvoření nové formy nebo panelu, používaný správce rozvržení může byt snadno změněn v okně Inspektoru (podrobněji podrobnější informace naleznete v kapitole 3.1) – komponenta -> **Set Layout**. Na Obr. 3 jsou zobrazeny kotvy, nabídnuté systémem pro umístění komponent



Obr. 3 Kotvy pro umístěni komponent

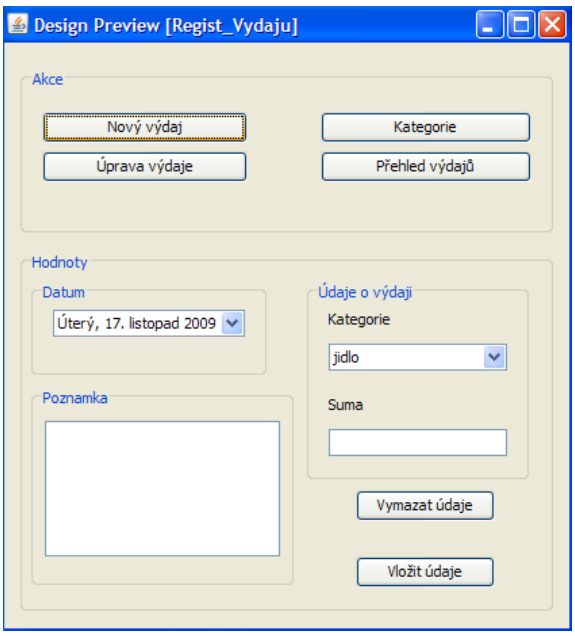
- **Nezávislost na platformě –** rozhraní, vytvořené v NetBeans IDE GUI Builder korektně funguje nezávislé na cílové platformě, nehledě na to, že každá platforma (ať již Mac OS X, Open Solaris nebo Microsoft Windows) má vlastní požadavky na umístění a zarovnání komponent.

  Poznámka: aby rozhraní bylo skutečně nezávislé na platformě, musí se uživatel při jeho tvorbě jednak vyhýbat použití ¨natvrdo¨ zakódovaných pozic a velikostí, a jednak dodržovat systémem navrhované zarovnání, pozice, kotvy, velikost a také náhledové nápovědy. Na Obr. 4 je zobrazená nápověda pro zarovnání komponenty.



Obr. 4 Nápověda pro zarovnání komponenty

- **Náhledy návrhu –** při tvorbě rozhraní může NetBeans IDE GUI Builder zobrazit návrh přesně tak, jak se bude zobrazovat při jeho skutečném použití, to jest přesný náhled tvořené aplikace, která však není funkční – touto funkcí nelze simulovat funkcionalitu aplikace.



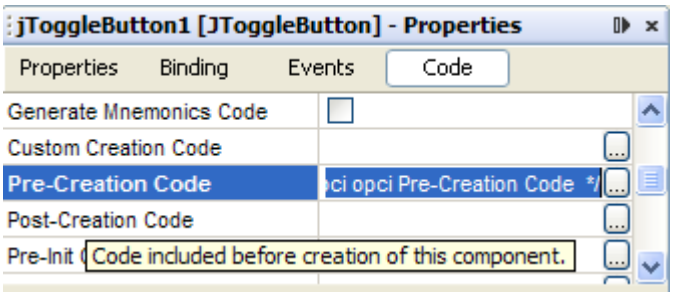
Obr. 5 Náhled návrhu

- **Generování kódu –** Uživatel si může podle své potřeby zobrazovat zdrojový kód, který se automaticky generuje dle toho, co uživatel právě tvoří na plátně (zobrazuje se v rámci metody initComponents()). Automaticky generovaný zdrojový kód, hodnoty komponent a také hlavičky správců událostí (ne jejich těla) není možné měnit, jsou chráněny proti náhodné a potenciálně chybné změně. Na Obr. 6 je zobrazena část kódu, která byla automaticky vygenerována nástrojem NetBeans IDE GUI Builder v metodě initComponents().

```
28      @SuppressWarnings("unchecked")
29      // <editor-fold defaultstate="collapsed" desc="Generated Code">
30      private void initComponents() {
31
32          jPanel1 = new javax.swing.JPanel();
33          jToggleButton1 = new javax.swing.JToggleButton();
34          jToggleButton2 = new javax.swing.JToggleButton();
35          jToggleButton3 = new javax.swing.JToggleButton();
36          jToggleButton4 = new javax.swing.JToggleButton();
37          jPanel2 = new javax.swing.JPanel();
38          jPanel3 = new javax.swing.JPanel();
39          jCalendarCombo1 = new org.freixas.jcalendar.JCalendarCombo();
40          jPanel4 = new javax.swing.JPanel();
41          jComboBox1 = new javax.swing.JComboBox();
```

Obr. 6 Automaticky vygenerovaný kód

Zároveň však GUI Builder poskytuje možnost bezpečné změny chráněného kódu: ke každé komponentě GUI může být dodán vlastní kód. Tento kód bude automaticky dodán ke kódu v metodě initComponents() a tím pádem se téměř úplně vylučuje možnost chybné změny chráněného kódu – viz Obr. 7 a **Chyba! Nenalezen zdroj odkazů.**. – podrobnější informace naleznete v kapitole 4.4.4.5.



Obr. 7 Možnosti pro změnu chráněného kódu

```
30      private void initComponents() {
31
32          jPanel1 = new javax.swing.JPanel();
33          /*
34          * kod, dodaný pomoci opci Pre-Creation Code
35          */
36          jToggleButton1 = new javax.swing.JToggleButton();
37          jToggleButton2 = new javax.swing.JToggleButton();
```

Obr. 8 Výsledek změny chráněného kódu

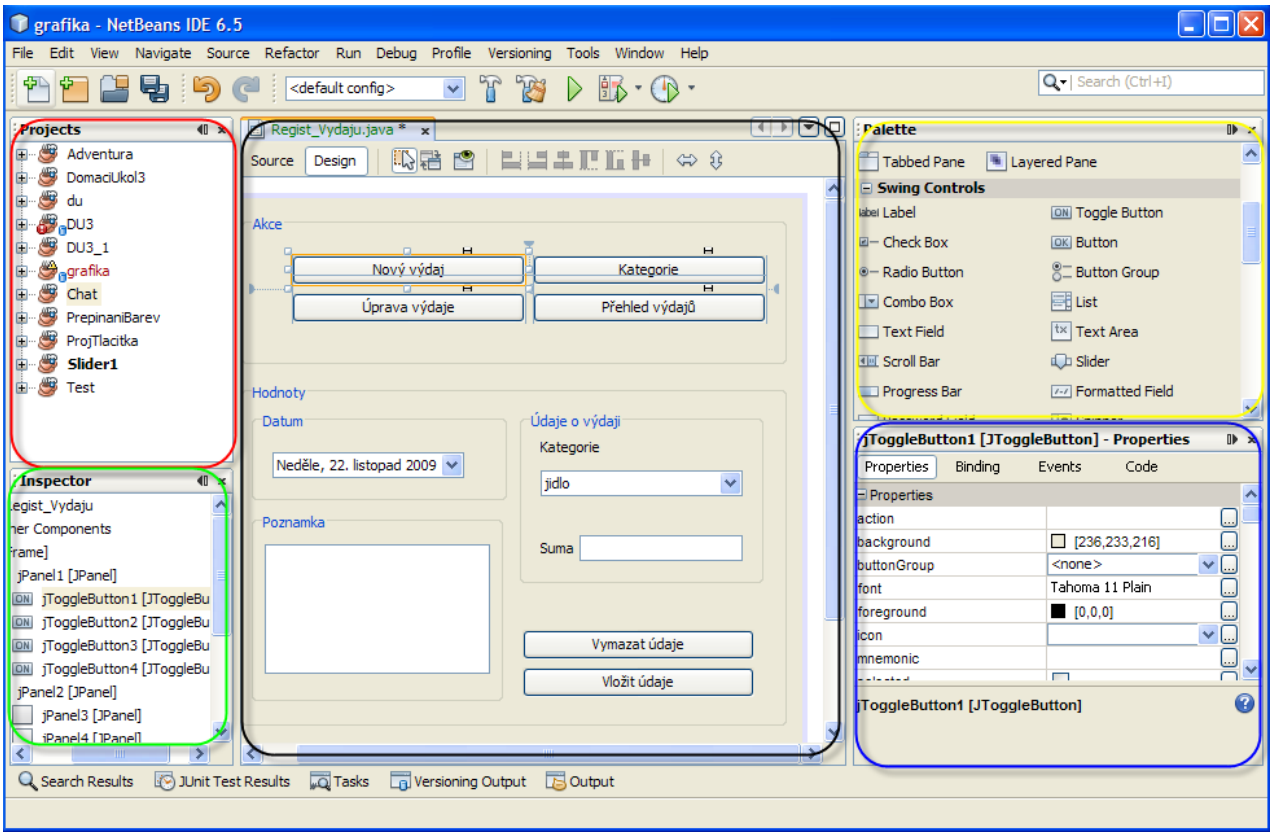
- **Zvláštnosti generovaného kódu:**
  Kód na šedém pozadí je chráněný a needitovatelný. Protože se nepoužívají importy, je každá třída plně kvalifikována (např. javax.swing.JButton namísto JButton).
  Každá sekce může být pro přehlednost rozbalena i zabalena znaky „+“ a „-“.
  Proměnné instance jsou deklarovány na konci třídy a ne na začátku.
  Konstruktér, vytvářející metodu initComponents() (která inicializuje kód), je editovatelný.

- **Internacionalizace –** NetBeans IDE GUI Builder umožňuje tvorbu internacionalizovaných rozhraní, tj. rozhraní, ve kterém řetězce, barvy a obrázky mohou být přeloženy a přizpůsobeny potřebám a určitým „kulturním očekáváním“ dle země, ve které se rozhraní bude používat bez jeho dodatečného překompilování. Pro více informací o pravidlech, které musí byt dodržovány pro kvalitní intrernacionalizivané rozhraní, a také pro návod k jeho tvorbě viz [1.2]

3. Struktura a komponenty NetBeans GUI Builder

3.1. Struktura NetBeans GUI Builder

NetBeans GUI Builder se skládá z několika oken, která jsou rozdělena dle účelu pro jednotlivé potřeby vývoje aplikací. Na Obr. 9 je zobrazeno hlavní okno nástroje GUI Builder se všemi základními oblastmi, používanými pro tvorbu aplikací.



Obr. 9 Jednotlivé sekce rozhraní NetBeans IDE GUI Builder

Základní oblasti hlavního okna NetBeans GUI Builder:

- **Design area** – oblast návrhu (označena černě) – hlavní okno pro tvorbu a editace formulářů Java GUI, ve kterém se definuje poloha komponent a fakticky probíhá grafický návrh aplikace. Oblast návrhu obsahuje řadu dodatečných tlačítek pro

usnadnění práce s prostředím: tlačítka pro změnu režimu (Tab. 1 Tlačítka změny režimuTab. 1) a tlačítka zarovnání a změny velikosti (Tab. 2).

Tab. 1 Tlačítka změny režimu

Ikona tlačítka	Název tlačítka	Popis tlačítka
	Selection Mode	Umožňuje výběr jednoho nebo několika objektů v oblasti návrhu
	Connection Mode	Umožňuje zadání spojení mezi objekty výběrem objektu, který bude generovat událost a objektu, který bude na tuto událost reagovat – podrobnější informace naleznete v kapitole 4.4.4.4)Connection mode.
	Preview Design	Umožňuje prohlédnutí tvořeného formuláře tak, jak bude skutečně vypadat ve výsledné aplikaci (neumožňuje ověření funkčnosti formuláře)

Tab. 2 Tlačítka zarovnáni a změny velikosti:

Ikona tlačítka	Název tlačítka	Popis tlačítka
	Align left/right in column	Zarovnání levého/pravého okraje dvou nebo několika komponent ve formuláři.
	Center horizontally/ vertically	Vodorovné/Svislé zarovnání dvou nebo několika komponent ve formuláři s použitím přesného centra obou komponent
	Align top/bottom in row	Zarovnání horního/dolního okraje dvou nebo několika komponent ve formuláři.
	Change horizontal resizability	Aktivované nebo deaktivované pro každou komponentu. V prvním případě velikost komponenty bude automaticky změněna ve vodorovném/svislém směru jestliže se ve stejném směru mění velikost rodičovské komponenty JFrame (velikost JFrame se mění posunem myši při kliknutí na pravý dolní roh komponenty).

- **Projects** (označena červeně) – obsahuje strom všech projektů, dostupných v IDE. Po rozbalení jakéhokoliv projektu se zde zobrazuje obsah stromu souborů a tříd. Zdrojový kód třídy (případně grafická forma formuláře) se otevře v oblasti návrhu dvojklikem na názvu třídy.

- **Palette** (označena žlutě) – přizpůsobitelný seznam dostupných komponent JFC/Swing, AWT a JavaBeans. Kategorie komponent a samotné komponenty mohou být dodávané pomocí **Palette Manager** (klik pravým tlačítkem myší na oblasti palety -> **Palette Manager**). Jako zdroj nových komponent může být využit soubor JAR, knihovna již nainstalovaná v IDE, či již existující projekt (využití JAR souboru již hotové aplikace).

- **Inspector** (označena zeleně) – zobrazení všech komponent v aplikaci ve formě hierarchického stromu. Inspector poskytuje vizuální zpětnou vazbu o komponentě, která je právě editována a zároveň umožňuje přehled o celkové organizaci komponent na všech přístupných panelech. V okně **Inspector** se zobrazují jak vizuální, tak i nevizuální komponenty, právě toto okno poskytuje možnost práce s nevizuálními komponenty (podrobnější informace naleznete v kapitole 3.2.3).

- **Property Editor** (označena modře) – seznam vlastností komponenty, která je právě zvolena nejen v oblasti návrhu, ale i v inspektoru, v oknu projektu či souboru. Pro přehlednost jsou vlastnosti komponent rozděleny do čtyř kategorií dle účelu, a to
  základní (Properties) – obecné vlastnosti komponenty
  vazby (Binding) – grafické vazby, například pozadí, nadpis, změna kursoru atd.
  událostí (Events) – interakce komponenty po nějaké události, například po kliknutí
  a **kódu** (Code) – kontrola Java kódu generovaného pro tuto komponentu
  Podrobnější informace o vlastnostech komponent naleznete v kapitole 4.4.

Pro pohodlí uživatele může být každá oblast schována do boční lišty kliknutím na šipku  v pravém horním rohu oblasti. Na boční liště pak bude schovaná oblast zobrazena ve formě tlačítka a rozbalena podle potřeby po klinknutím na něj.
Při potřebě permanentního zobrazení oblasti v hlavním okně jednoduše rozbalte oblast z boční lišty a klikněte na kolečko  v pravém horním rohu oblasti – oblast se opět ukotví

v hlavním okně na původním místě.
Každá oblast může být také úplně vymazána z hlavního okna (kliknutím na křížek  v pravém horním rohu oblasti) a opětovně zobrazena použitím možností **Window** v horním menu aplikace (**Inspector** se zobrazí pomoci **Window -> Navigating**)

3.2. Standardní komponenty NetBeans GUI Builder a jejích použití

3.2.1. Komponenty AWT

AWT je zkratka od Abstract Window ToolKit. Jedná se o GUI knihovnu pro aplikace a applety, zahrnující sadu komponent rozhraní s bohatými modely pro zpracování událostí. K přednostem AWT patří rychlost a jednoduchost použití, přenosnost appletů (většina prohlížečů třídy AWT podporuje, což znamená, že applety AWT mohou fungovat i bez nainstalovaného pluginu Javy). Nedostatky AWT zahrnují určité ohraničeni v závislosti na platformě (některé komponenty jsou příliš specifické a tak vůbec nefungují na některých platformách), malé množství nabízených AWT komponent, a také fakt, že AWT komponenty nepodporují vlastnosti typu piktogram a nápovědy. Kromě toho, většina producentů komponent, jako Borland a Sun zakládají nové vlastní komponenty na komponentách Swing.
Použití komponent AWT je obecně vhodnější při tvorbě menších appletů nebo při vývoji, určeném pro jednu stanovenou platformu.

3.2.2. Komponenty Swing

Komponenty Swing jsou založené na technologii AWT, avšak jsou implementovány výhradně v jazyce Java. V porovnání s komponenty AWT jsou komponenty Swing méně závislé na platformě, dále podporují podstatně širší výběr vlastností, a jejich vývoj v dnešní době je neporovnatelně intenzivnější, než vývoj komponent AWT. Swing toolkit obsahuje široký výběr komponent: od nejednodušších komponent typu tlačítek nebo checkboxů až po složité a sofistikované komponenty typu tabulek. Dokonce zdánlivě jednoduché komponenty, např. textová pole, disponují velice vypracovanou funkcionalitou, jako zadáni formátovaného textu nebo schování pole hesla. Také je možná kastomizace komponent. Z jiné stránky mnoho prohlížečů nepodporuje třídy Swing, tj. musí být používán Java plugin. Komponenty Swing jsou obecně pomalejší a jejich použití je spojeno s častějším vznikem chyb.

Použití komponent Swing je vhodné při tvorbě větších grafických aplikaci a appletů, a také pro tvorbu aplikací určených pro použití na několika různých platformách.
V následujících tabulkách jsou stručně popsané komponenty Swing, nabízené k použití nástrojem NetBeans GUI Builder[7].

Tab. 3 Kontejnery Swing

Název	Použití
Panel	Poskytuje multifunkční kontejnery pro menší komponenty. Standardní panely nepředávají barvy ničemu, kromě jejích vlastního pozadí a nezobrazují hranice. Nicméně hranice a názvy panelu mohou byt snadno dodány (**Properties->Border->TitledBorder->Title**)
Tabbed Pane	Poskytuje několika komponentám možnost sdílení stejného prostoru. Uživatel vybírá komponentu pro prohlížení pomocí záložek.
Scroll Pane	Poskytuje náhled na komponentu, rolovatelný po obrazovce. Používá se pro ušetření místa v případě ohraničeného prostoru pro zobrazení velké komponenty (nebo komponenty, jejíž velikost se muže dynamicky měnit).
Layered Pane	Poskytuje třetí rozměr pro umístěni komponent: hloubku (depth). Při dodáni komponenty na **Layered Pane** je zapotřebí definovat její hloubku jako celé číslo – čím větší bude to číslo, tím ¨blíže¨ bude komponenta k vrchní pozici kontejneru. Jestliže se komponenty překrývají, bližší komponenta je zobrazena nad komponenty s nižší hloubkou.
Split Pane	Zobrazuje dvě komponenty vedle sebe, nebo nad sebou. Velikost částí panelu může být různá (uživatel je definuje pomoci rozdělovače). Obrazovka může být rozdělena na potřebné množství částí pomocí dodání dalších rozdělených panelů (**Split Panes**) do již existujících. ***Poznámka***: komponenty mohou být nejdříve dodány do Scroll Pane a po té do **Split Pane** – tím pádem uživatel získá možnost prohlížení komponent jakékoliv velikosti bez zbytečného plýtvání místem na obrazovce.

[7] Detailní popis komponent AWT je vynechán z rozsahu této práce jednak z důvodu jejich jednoduchosti a intuitivní pochopitelnosti, a jednak kvůli jejich méně rozšířenému použití v moderních aplikacích (v porovnání s komponenty Swing).

Tool Bar	Kontejner, seskupující několik komponent – obvykle tlačítek – poskytuje jednoduchý přístup k funkcím obsaženým v menu.
Internal Frame	Poskytuje možnost zobrazení oken typu **Frame** v rámci jiných oken.
Desktop Pane	Používá se pro tvorbu rozhraní s několika dokumenty nebo virtuálními obrazovkami. Je většinou používán jako kontejner pro vnitřní panely, které se mohou překrývat.

Tab. 4 Řídicí komponenty Swing

Název	Použití
Button	Vytváří jednoduché tlačítko.
Button Group*[8]	Vytváří skupinu tlačítek, při „zapnutí“ jednoho se ostatní „vypínají“.
Toggle Button	Poskytuje tlačítko, které muže měnit svůj tvar – např., barvu při „zmačknutí“ apod.
Radio Button	Poskytuje skupiny tlačítek, ve kterých jen jedno tlačítko muže byt zvoleno.
Check Box	Poskytuje skupiny tlačítek, ve kterých může být zároveň vybráno několik tlačítek.
Label	Umožňuje zobrazení ne-interaktivních obrázků nebo nadpisů.
Combo Box	Umožňuje uživatelovi výběr z několika možností. Může být editovatelný nebo needitovatelný (vlastnost se nastavuje v oblasti **Properties**). Je velice užitečný při potřebě úspory místa.
Text Field	Umožňuje zadání malého množství textu. Po ukončení zadávání (většinou zmačknutím enteru) textové pole spouští událost - **action**.
Text Area	Principielně podobné komponentě **Text Field**, ale se používá při potřebě zadání většího množství textu (několik řádků).
Editor Pane a Text Pane	Používá se při potřebě nejen zadání textu, ale také jeho editaci za použití fontu a stylu.

[8] Komponenty, označené hvězdičkou jsou takzvané nevizuální komponenty, tj. při jejich dodání na plátno nejsou graficky zobrazeny. Práce s takovými komponentami je popsána v kapitole 3.2.3.

Password Field	Poskytuje speciální textové pole pro zadání hesla – z bezpečnostního důvodu jednak nezobrazuje zadávané symboly a jedna uchovává zadané hodnoty ve formě pole znaků a ne jako řetězce.
Scroll Bar	Poskytuje rolovací proužek, který umožňuje pohodlný výběr potřebné hodnoty z řady hodnot.
Slider	Umožňuje snadné zadání číselné hodnoty při uvedené minimální a maximální hodnotě.
Spinner	Je alternativou komponenty **Slider** při potřebě úspory místa, umožňuje výběr z několika hodnot, a také zadání vlastní hodnoty. Na rozdíl od komponenty **Combo Box** spinnery nezobrazují seznam hodnot – je vidět pouze aktuální hodnota, proto se často používají místo komponenty **Combo Box** v případech, ve kterých je množství elementů velké a jejich pořadí je zjevné. **Spinner** používá standardně formátovaný text.
Progress bar	Umožňuje názorné zobrazení množství času, který potřebuje určitý úkol, nebo také kolik procent úkolu již bylo splněno, a kolik ještě zbývá.
Separator	Poskytuje horizontální nebo vertikální rozdělující čáru, nebo prázdné místo. Na rozdíl od ohraničení jsou zvláštními komponentami a jsou zobrazeny v rámci kontejneru a ne na okrajích jiné komponenty (může se také používat v menu).
Table	Umožňuje zobrazení tabulek, jejichž buňky mohou být podle potřeby citovatelné. Neuchovává data, je pouhým jejích zobrazením.
List	Poskytuje seznam elementů, ze kterých uživatel může vybírat ty potřebné.
Formatted Field	Poskytuje možnost definování jakékoliv speciální sady znaků, která může být zadána do textového pole. Formátovač převádí hodnotu pole do zobrazovaného textu a text zpátky do hodnoty pole. Například pomocí formátovače se dá zobrazovat hodnoty datu v lokalizovaném formátu, používat masky pro zadání hodnot v určitém formátu (např. telefonní čísla ve formátu (XX) X-XX-XX-XX-XX).
Tree	Umožňuje zobrazení hierarchických dat.

Tab. 5 Menu Swing

Název	Použití
Menu Bar	Poskytuje uživateli snadný způsob výběru z několika možností – tato volba zobrazuje pouze horní lištu skládající se ze dvou standardních tlačítek menu **File** a **Edit**. Elementy menu lze přidat pomocí komponenty **Menu Item.** Při potřebě dodání dalších tlačítek do horního menu se používá komponenta Menu.
Menu Item	Dodává elementy do jednotlivých menu-tlačítek lišty **Menu Bar**.
Menu Item/Radio button	Dodává do jednotlivých tlačítek lišty **Menu Bar** elementy s radio-tlačítky, která uživatel může dle potřeby volit (přičemž může být zároveň označeno několik variant).
Menu Item/Checkbox	Dodává do jednotlivých menu-tlačítek lišty **Menu Bar** elementy s checkbox-tlačítky, která uživatel může dle potřeby volit.
Menu	Dodává elementy do lišty **Menu Bar.**
Popup Menu*	Spojením na akci (např. kliknutí myši na nějakou z jiných komponent v rámci okna) se vyroluje okno s nabídkou zadaných možností. Pozice tohoto okna je generována dynamicky dle aktuální pozice kursoru.

Tab. 6 Okna Swing

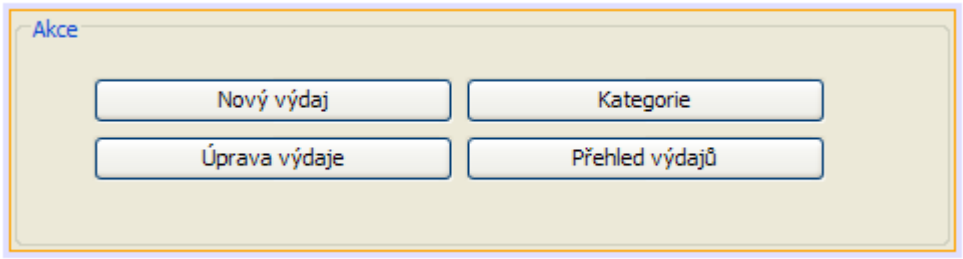
Název	Použití
Dialog*	Poskytuje nezávislé podokno, které je určené k oddělení dočasné zprávy pro uživatele od hlavního okna aplikace (může obsahovat chybové hlášení, varování, apod.)
Frame	Poskytuje hlavní aplikační okno nejvyšší vrstvy.
Color Chooser	Poskytuje paletu pro výběr barev. Tato komponenta se muže byt ujištěná v jakékoliv částí tvořeného rozhraní.

File Chooser	Poskytuje standardní okno pro navigaci ve fialovém systému (s možností volby souboru ze seznamu nebo zadání nazvu souboru nebo složky)
Option Pane	Poskytuje standardní dialogové okno sloužící uživateli pro zobrazení hodnot, nebo libovolných informací.

Tab. 7 Java Persistence

Název	Použití
Entity Manager	Spravuje persistence elementy, elementy které slouží pro uchování dat i po ukončení procesu jejich získání, například zachování výsledných dat z databázového SELECTu.
Query	Slouží pro implementaci dotazu
Query result	Slouží pro interpretaci výsledků dotazu Query

Pro vytvoření větších a složitějších aplikací a formulářů se doporučuje pro jejich jednotlivé části použití zvláštních kontejnerů. Veškeré komponenty, použité v tomto přikladu, nejsou umístěny přímo na hlavní panel, ale na menší panely (s názvy Akce, Hodnoty, Datum, atd., viz Obr. 1). Jednak to dělá návrh přehlednějším, jednak zjednodušuje případnou editaci návrhu a změnu rozmístění komponent v aplikaci. NetBeans IDE GUI Builder poskytuje pohodlnou možnost jednoduché editace samostatných komponent zvlášť místo jejich složité editace v rámci jedné velké plochy. Komponenta může být zobrazena buď samostatně dvojklikem na ní, nebo přes klik pravým tlačítkem myši na panelu a volbu **Design This Container –** viz Obr. 10).



Obr. 10 Samostatné zobrazení komponenty

Do normálního plného zobrazení celého formuláře se lze navrátit opětovným dvojklikem na panelu, nebo skrze klik pravým tlačítkem myši na panelu a volbu **Design Parent -> Top Parent**.

3.2.3. Práce s neviditelnými komponentami

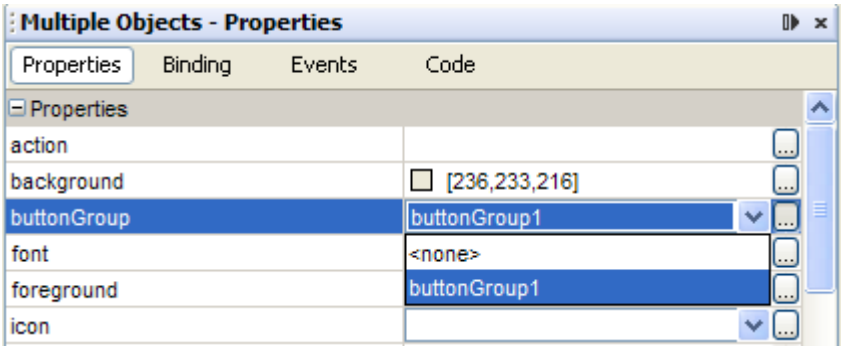
Některé z komponent nejsou viditelné v oblasti návrhu, ale pouze v okně Inspector. Je to dáno zvláštností těchto komponent, které mají buďto povahu čistě datovou bez visuelního projevu (např. objekty Java Persistence), nebo speciální grafický projev (např. Popup Menu). Těmto komponentám se říká neviditelné, nebo nevizuální komponenty. Tyto komponenty slouží pro doplnění funkcionality ostatních viditelných prvků, nevisuelní zpracování dat (Java Persistence) a jiné. To ovšem nutně neznamená, že se tyto objekty v GUI rozhraní nikterak graficky neprojeví. Příkladem pro neviditelnou, avšak ve výsledném rozhraní viditelnou komponentu, je komponenta Popup Menu. Samostatnou kapitolou jsou pak dialogová okna (Windows / Dialog).
Jelikož práce s takovými komponentami může být zvlášť obtížná, je tato subkapitola věnována popisu dodáni funkčnosti těmto komponentám (kromě komponent Java Persistence, které přesahují rámec této práce).

3.2.3.1. Button Group

Komponenta **Button Group v**ytváří skupinu tlačítek, při „zapnutí“ jednoho se ostatní „vypínají“.

Pro dodání několika tlačítek do Button Group:

1. Přetáhněte do oblasti návrhu komponentu **Button Group** – komponenta nebude grafický znázorněná, ale zobrazená v Inspektoru v sekci **Other Components**.
2. Vyberte tlačítka, která budou přidána do Button Group (popis výběru několika komponent je popsán v kapitole 4.2.5).
3. V okně **Properties** vyberte společnou vlastnost vybraných tlačítek **buttonGroup** a definujte **Button Group**, do které budou tlačítka přidány – viz Obr. 11.



Obr. 11 Dodání tlačítek do Button Group

Tlačítka se stanou součásti Button Group s odpovídajícími vlastnostmi.

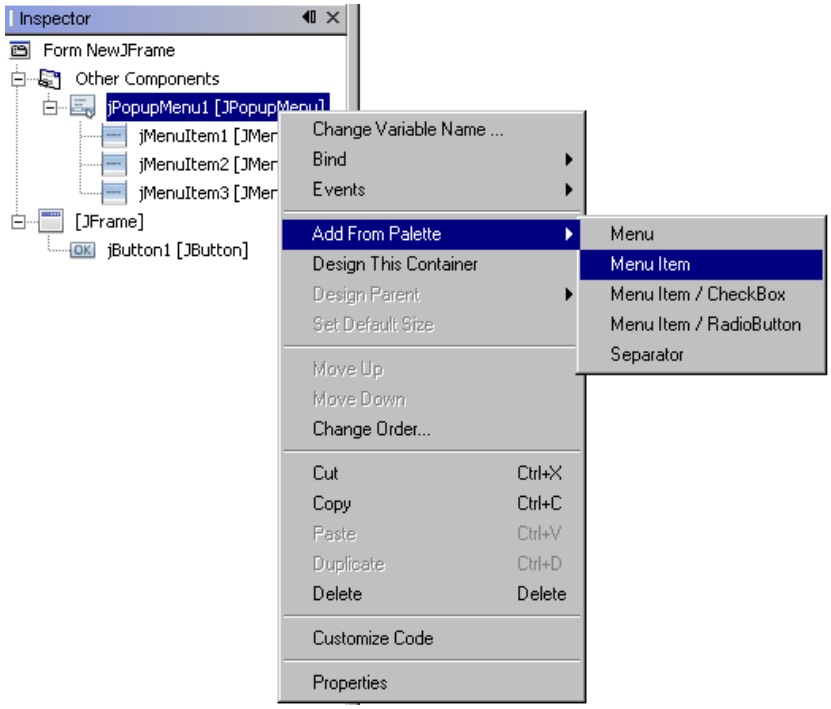
3.2.3.2. Popup menu

Popup menu je menu, které je rozbaleno kliknutím pravého tlačítka myší na nějaké komponentě. Popup menu se často používá v grafických rozhraních, nicméně na plátně v GUI Builder se nezobrazí.

Nejlépe lze práci s touto komponentou vysvětlit na následujícím konkrétním příkladě.

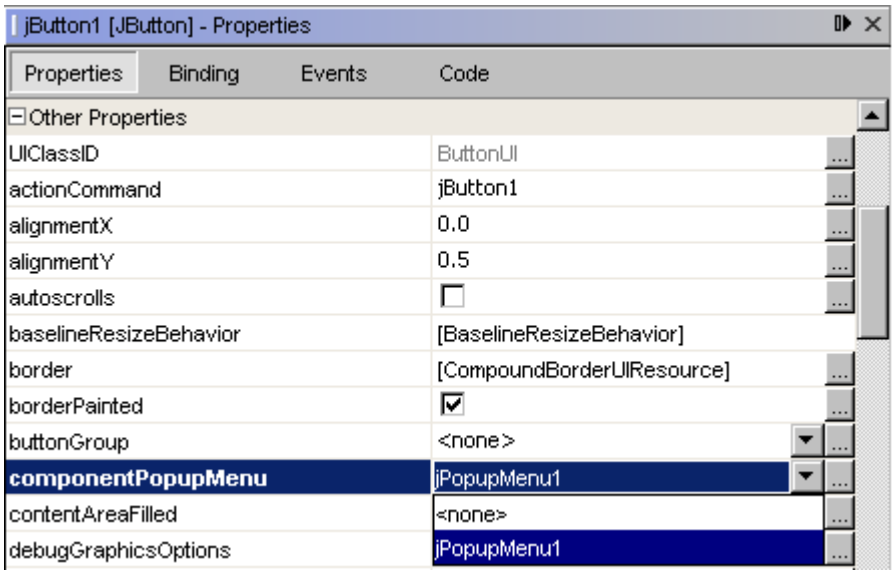
Je-li zapotřebí vytvořit popup menu, které bude zobrazeno po kliknutí pravým tlačítkem myší na obecné tlačítko, postup je následující:

1. Na pole návrhu přetáhněte komponenty **Popup Menu** z **Palette Swing Menus**.
2. Klikněte pravým tlačítkem v poli **Inspector** a zvolte **Add From Palette**. Zde lze vybrat z více možností, standardem je položka Menu Item. (viz Obr. 12)



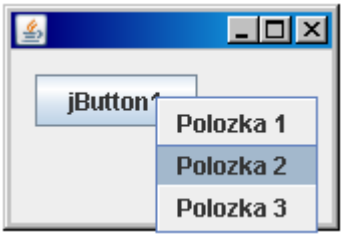
Obr. 12 Dodání elementu do Popup menu

3. Po vytvoření položek popup menu zvolte na plátně komponentu, ke které má být menu asociováno. V panelu **Properties** této komponenty zvolte parametr **componentPopupMenu** a v poli hodnoty zvolte jejím rozbalením již vložené **jPopupMenu1** (viz Obr. 13)



Obr. 13 Asociování Popup menu ke komponentě

Popup menu bude vytvořeno, nicméně při zobrazení náhledu návrhu nebude fungovat. Pro ověření funkčnosti Popup menu (a zároveň i pro zhlédnutí vytvořeného návrhu) je zapotřebí spustit daný projekt **Run->Run File**. Pokud build proběhne v pořádku, lze Popup menu jednoduše otestovat (viz Obr. 14).

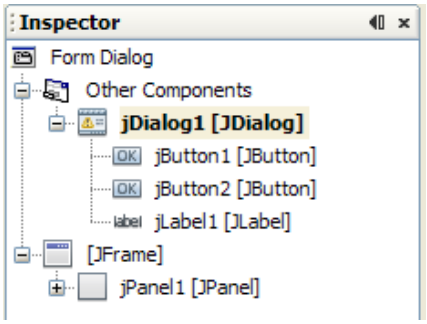


Obr. 14 Výsledek dodání Popup menu ke komponentě Button

3.2.3.3. Dialog

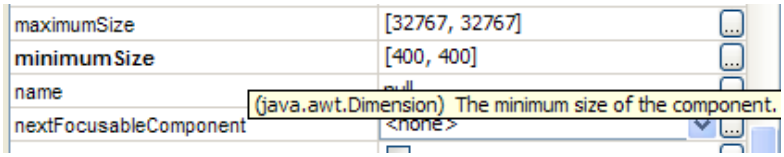
Vzhled různých dialogových oken se může velice lišit, navíc jsou většinou dialogová okna zobrazována jako reakce na určitou akci uživatele – z těchto důvodů musí být tato komponenta navrhována v GUI Builder zvláštním způsobem.
Nejlépe lze práci s touto komponentou vysvětlit opět na následujícím konkrétním příkladě.
Je-li zapotřebí vytvořit dialogové okno, které bude zobrazeno po kliknutí na tlačítko, postup je následující:

1. Na pole návrhu přetáhněte komponentu **Dialog**. Komponenta nebude graficky znázorněna, zobrazí se pouze v okně **Inspector** v sekci **Other Components** (viz Obr. 15)
2. Pomocí možnosti **Add From Palette** dodejte do dialogového okna potřebné komponenty. V popisovaném příkladu to budou dvě tlačítka a nadpis. I když se samotné okno dialogu na plátně nezobrazí, komponenty, které jsou součásti dialogového okna, mohou být zobrazeny, a to buďto dvojklikem na názvu dialogu v okně **Inspector**, nebo pomocí možnosti **Design This Container**.



Obr. 15 Zobrazení neviditelné komponenty Dialog v okně Inspector.

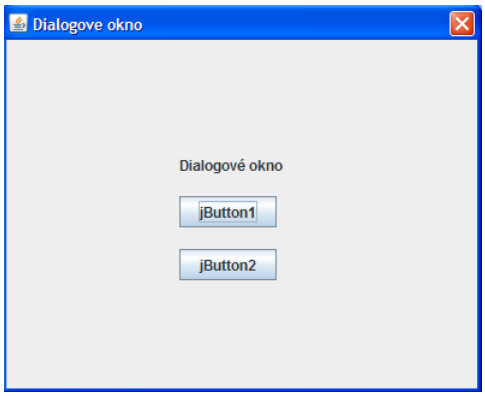
3. Pomocí okna **Properties** nastavte minimální možnou velikost dialogového okna – standartě je tato hodnota je nastavena na (0,0) (viz Obr. 16).



Obr. 16 Nastavení minimální možné velikosti dialogového okna

4. Definujte komponentu, jíž událost způsobí zobrazení dialogového okna a dodejte potřebný kód k metodě události (podrobnější informace naleznete v kapitole 4.4.4.5)

Po provedení buildu projektu a kliknutí na spouštěcí tlačítko se zobrazí dialogové okno.



Obr. 17 Výsledné dialogové okno

4. Vytvoření aplikace s GUI

Veškerý vývoj v IDE se provádí v rámci projektů a z toho důvodu je i pro tvorbu rozhraní potřeba založit projekt, ve kterém se pak budou ukládat veškeré soubory.

Pro založení projektu:

1. Klikněte na **Choose File > New Project** (nebo použijte ikonu ).
2. V okně **Categories** zvolte **Java node**, v okně **Projects** zvolte **Java Application**. Klikněte na **Next**.
3. V poli **Project Name** zvolte název projektu, v poli **Project Location** zadejte umístění projektu.
4. Ponechte políčko **Use Dedicated Folder for Storing Libraries** nevyplněným (nezvoleným).
5. Vyberte políčko **Set as Main Project**.
6. Klikněte na **Finish**.

Po vytvoření projektu je možné začít s tvorbou rozhraní.

4.1. Vytvoření Kontejneru GUI

1. V okně **Projects** klikněte pravým tlačítkem myší na nazvu projektu a zvolte **New > JFrame Form**[9].
2. V poli **Class Name** zadejte název třídy.
3. V poli **Package** zvolte balík, ve kterém bude třída uložena.
4. Klikněte na **Finish**.

[9] Ve skutečnosti se touto volbou dá vybrat neporovnatelně více různých druhů šablon (JApplet, JDialog, Bean Form, atd.), které se spíše liší svým použitím, nikoliv návrhem rozhraní. Z tohoto důvodu pro tuto práci použijeme výhradně **JFrame Form**).

4.2. Vkládání komponent na formulář

Pořadí komponent na formuláři odpovídá pořadí, ve kterém jsou na formulář vloženy. U některých správců rozvržení pořadí komponent určuje také jejich grafické zobrazení na plátně (pořadí se dá měnit přetahováním samotných komponent na plátně nebo v okně Inspektoru). Podstatné je, že se standardně na plátně používá **Free Design**, tj. uživatel může dodat komponentu na jakékoliv místo na plátně.

4.2.1. Vložení komponenty z Palette

1. V okně Palette vyberte potřebnou komponentu kliknutím na její ikonu.
2. Přetáhněte komponentu na požadované místo v oblasti návrhu[10].

Poznámka: jestli na definovaném místě není pro komponentu dostatečný prostor, vedlejší komponenty se posunou.

4.2.2. Vložení několika komponent z Palette

1. V okně Palette vyberte potřebnou komponentu kliknutím na její ikonu
2. Podržte stisknutou klávesu Shift a kliknutím na místech plátna, kde je třeba umístit tuto komponentu, jí následně snadno vložíte.

4.2.3. Vložení komponent pomocí okna Inspector

1. V okně Inspector klikněte pravým tlačítkem myši na kontejneru, kam je zapotřebí vložit komponentu.
2. Z podmenu **Add From Palette** vyberte potřebnou komponentu a vložte jí na požadované místo na plátnu.

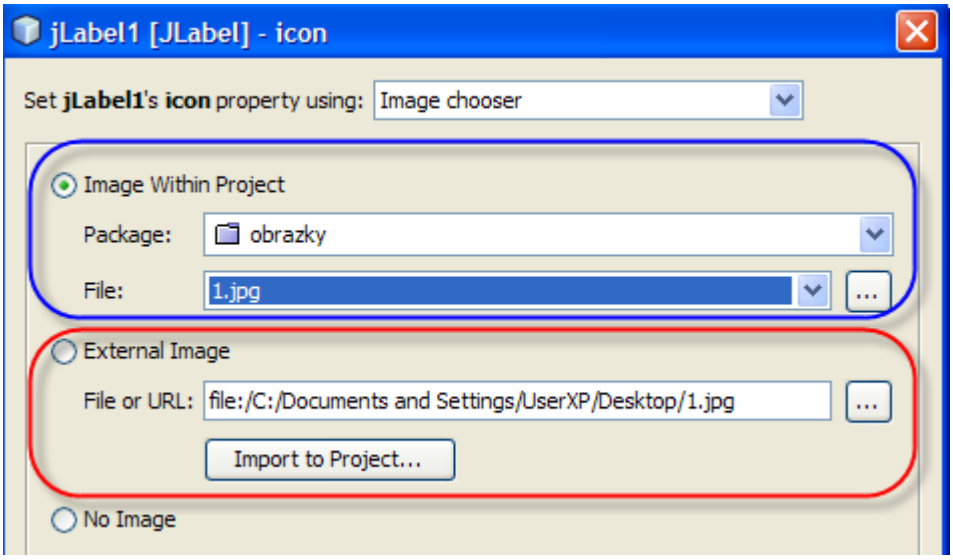
[10] Jestli-že komponenta bude přetáhnuta na bílý prostor mimo oblast návrhu, bude dodána do uzlu **Other Component** a nezobrazí se graficky.

4.2.4. Vložení obrázku

Důležitou součástí grafického rozhraní často bývají obrázky. V této kapitole bude představen způsob vložení obrázku do aplikace.
Pro vložení obrázku do aplikace je vhodné použít komponentu Label (nebo Button). Po přidání této komponenty na plátno pak následuje tento postup:

1. V okně **Properties** (podokno **Properties**) zvolte vlastnost **icon** a klikněte na tlačítko  naproti této vlastnosti.
2. V okně formátování vlastnosti zvolte **External Image** a zadejte cestu k obrázku, který bude použit v aplikaci.
3. Klikněte na **OK**.

Obrázek pak bude přidán k nadpisu.
Tento způsob je na Obr. 18 označen červenou barvou.
Při použití této metody IDE, generuje absolutní cestu k obrázku, a nekopíruje jej tak do projektu. Tím pádem bude obrázek zobrazován korektně pouze při spuštění aplikace v systému, ve kterém se daný obrázek nachází na daném místě. Nebude-li tomu tak, nebude obrázek zobrazen.
Této komplikaci lze předejít tak, že obrázek se nejdříve přidá do projektu (do nové, nebo existující složky) – zkopírování a vložení obrázku do složky, nebo jeho jednoduché „přetažení“ do IDE. Poté se opakuje stejný postup, jako v předcházejícím příkladu s tím rozdílem, že namísto **External Image** je zvoleno **Image Within Project**. Dále je pak v políčku **Package** zvolena složka, ve které se nachází potřebný obrázek, a v políčku **File** – soubor s obrázkem.
Tento způsob je na Obr. 18 označen modrou barvou.

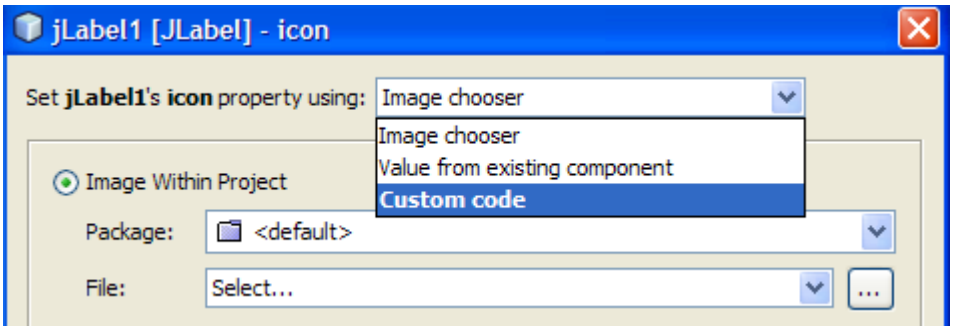


Obr. 18 Volba obrázku k dodání na plátno

Text nadpisu, který se zůstává na plátně i po dodání obrázku lze jednoduše smazat.

Obrázky v aplikaci nejsou často zadány staticky, jako v předchozím příkladu, například obrázek se může zobrazit jako odezva na určitou akci uživatele. V tomto případě je pro přístup ke zdrojům a jejich včasnému a správnému zobrazení možné dodání vlastního kódu.

V okně formátování vlastnosti **icon** zvolte **Custom Code** (viz. Obr. 19) a poté vyplňte parametr metody setIcon() (tento parametr může být vyplněn buď potřebnou logikou, nebo voláním speciální metody, naprogramované v jiném úseku třídy).



Obr. 19 Dodání vlastního kódu pro práci s obrázky

4.2.5. Výběr komponent ve formuláři

Pro výběr jedné komponenty klikněte na komponentu na plátnu nebo v oknu Inspector.
Pro výběr komponenty následující, nebo předcházející v pořadí použijte patřičně klávesy **Tab** a **Shift** - Tab.
Pro výběr rodičovské komponenty zvolené komponenty podržte klávesu **Alt** a klikněte na potřebnou komponentu.
Pro výběr sub-komponenty zvolené komponenty podržte klávesy **Alt** a **Shift** a klikněte na potřebnou komponentu.
Pro výběr několika komponent podržte klávesu **Kontrol (Ctrl)** a klikněte na potřebné komponenty na plátně nebo oknu Inspector (případně podržte klávesu **Shift** a nakreslete obdélník na plátnu – veškeré komponenty v obdélníku budou vybrány).

4.3. Zarovnání komponent

Po dodání komponent se dá patřičným způsobem uspořádat jejích zarovnání.

Pro zarovnání komponent:

1. Vyberte komponenty k zarovnání.
2. Klikněte na potřebné tlačítko zarovnání v panelu instrumentu, IDE následně změní pozice komponent a jejich ukotvení.

Pro zarovnání komponent s použitím vodicích čar:

1. Vyberte komponenty k zarovnání.
2. Dotáhněte komponentu ke komponentě, se kterou musí být zarovnaná.
3. Potom, co se objeví vodící čáry (což znamená, že obě komponenty jsou zarovnávané), zarovnejte komponenty souhlasně k čárám a klikněte ve finálně zvolené pozici komponenty.

Občas se při práci v oblasti návrhu komponenty na plátně nečekaně posouvají. To je způsobeno tím, že při návrhu formuláře uživatelem přepočítává GUI Builder nejvhodnější pozici komponent a výsledky těchto propočtů se ne vždy shodují s přáním uživatele.

Nicméně, při dodržování dále uvedených principů, lze takovémuto nepříjemnému chování komponent předejít.

1. **Již na začátku je nutné mít přesnou představu o tom, jak bude daný formulář vypadat.** Je lepší dodat komponentu rovnou na místo, které se pak již nebude měnit.
2. **Návrh je vhodné navrhovat po menších částech**. Jestliže má aplikace složitou grafickou formu doporučuje se jí rozdělit pomocí panelu na menší části – při nesprávném chováni komponent je pak snazší opravovat menší části formuláře.
3. **Je vhodné používat nástroje zarovnání** -  .
4. **Dodržovat vodící čáry**. Použití vodících čar automaticky navrhovaných nástrojem GUI Builder pří umístění komponent do kontejneru pomáhá dodržovat nejlepší odstup od ostatních komponent a zmenšuje pravděpodobnost nežádoucí změny umístění.
5. **Space Around Components**. Při potřebě pouhých kosmetických změn a editace umístění komponent vůči ostatním komponentám je vhodné používat nástroj **Space Around Components** z vytahovacího menu místo manuálního přetahování.
6. **Nástroj Same Size**. Při potřebě nastavení stejné velikosti několika komponent je vhodné použití nástroje Same Size z vytahovacího menu místo manuálního zarovnání komponent.

Pro efektivní zarovnání komponent se používají následující vizuální pomůcky:

- Vodící čáry
- Ukazatele ukotvení
- Rozměrové ukazatele

4.3.1. Vodící čáry

Vodící čáry se objevují jen při vložení, nebo posouvání komponent a ukazují nejvhodnější bod, ve kterém by měla být komponenta umístěna.

4.3.1.1. Vodící čára Inset

Definuje preferované odsazení mezi komponenty a kontejnery, ve kterých se komponenty nacházejí. Odsazení je znázorněno pomocí čárkované svislé nebo vodorovné čáry.

Obr. 20 Vodící čáry Inset

4.3.1.2. Vodící čára Offset

Definuje preferovanou mezeru mezi vedlejšími komponenty (znázorněné pomocí čárkované vertikální nebo horizontální čáry), viz Obr. 21Obr. 1 - svislá čára.

4.3.1.3. Vodící čára Baseline

Definuje preferované umístění vedlejších komponent, obsahujících text (znázorněné pomocí čárkované vertikální nebo horizontální čáry), viz Obr. 21 - vodorovná čára.

Obr. 21 Vodící čáry Baseline

4.3.1.4. Vodící čára Edge

Definuje možné zarovnání vedlejších komponent (znázorněné pomocí čárkované vertikální nebo horizontální čáry).

Existují 4 typy tohoto zarovnání:

- **Top/Bottom** – zarovnání horních/dolních okrajů dvou komponent:

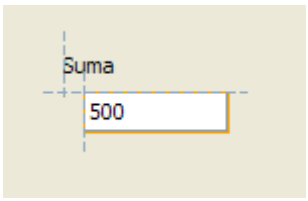
Obr. 22 Vodící čáry Edge Top/Bottom

- **Left/Right** - zarovnání levých/pravých okrajů dvou komponent

Obr. 23 Vodící čáry Edge Left/Right

4.3.1.5. Vodící čára Indentation

Indentation je speciálním typem zarovnání, při kterém je jedna komponenta umístěná pod jinou komponentou s mírným odstupem doprava. Odsazení je znázorněné pomocí dvou čárkovaných svislých čár.



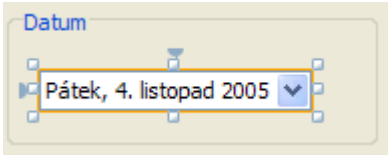
Obr. 24 Vodící čáry Indentation

4.3.2. Ukazatele ukotvení

Po výběru pozicí komponent se zobrazují pevné ukazatele ukotvení a definují vzájemné ukotvení komponent (komponenty ke kontejneru, ve kterém se nachází, nebo k jiné komponentě).

4.3.2.1. Ukazatel ukotvení Container

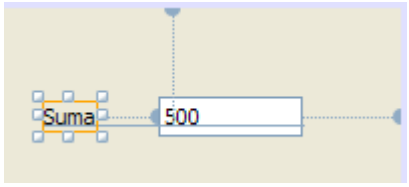
Zobrazuje ukotvení komponenty v kontejneru, ve kterém se komponenta nachází.



Obr. 25 Ukazatel ukotvení Container

4.3.2.2. Ukazatel ukotvení Component

Zobrazuje ukotvení komponent vůči jiné komponentě.



Obr. 26 Ukazatel ukotvení Component

4.3.3. Rozměrové ukazatele

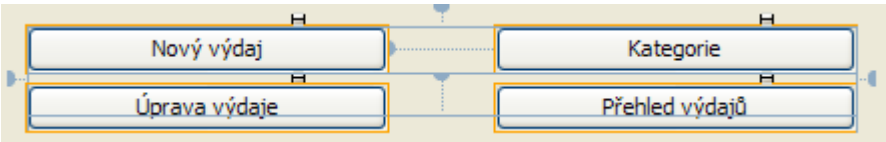
Rozměrové ukazatele slouží pro zadání rozměru komponent.

4.3.3.1. Rozměrové ukazatel Same Size

Definuje stejnou šířku nebo výšku pro skupinu komponent (vedlejší nebo vzdálené od sebe). Pro zadání stejné velikosti komponent:

1. Vyberte potřebné komponenty.
2. Klikněte pravým tlačítkem na komponentu, jejíž rozměry (výška nebo šířka) budou použity na celou skupinu.
3. Z menu vyberte **Same size -> Same Width** (a/nebo **Same Hights**)[11].

[11] Obě varianty mohou být vybrány zároveň.



Obr. 27 Ukazatele Same Size

4.3.3.2. Rozměrový ukazatel Auto-Resizing

Definuje dynamickou změnu výšky nebo šířky komponenty při změně rozměru okna.

4.4. Vlastnosti komponent

Po vložení komponenty do formuláře, je možná editace vlastností této komponenty pomocí okna **Properties**.

Pro editace vlastnosti komponent:

1. Vyberte potřebnou komponentu na plátně nebo v okně Inspector – její vlastnosti se zobrazí v okně **Properties**. Při výběru několika komponent se zobrazují jejich společné vlastnosti a změny budou aplikované ke všem vybraným komponentám.
2. Vyberte potřebnou kategorii vlastnosti zmačknutím tlačítka v horní části okna (**Properties**, **Binding**, **Events**, **Code**).
3. Editujte vlastnost komponenty v okně Properties zadáním potřebné hodnoty (případně použijte speciální editor vlastnosti, který se zobrazuje tlačítkem se třemi tečkami)
4. Klikněte na OK.

Po těchto krocích pak IDE aplikuje nový parametr k vybrané komponentě.

Pro návrat ke standardním hodnotám klikněte pravým tlačítkem na názvu vlastnosti a vyberte **Restore Default Value.**

4.4.4. Kategorie vlastnosti komponent

4.4.4.1. Properties

V části okna **Properties** se dají nastavit běžné vlastnosti jednotlivých komponent (nebo skupiny komponent), např. typ písma, pozadí, ohraničení a jiné vlastnosti typické pro danou komponentu.

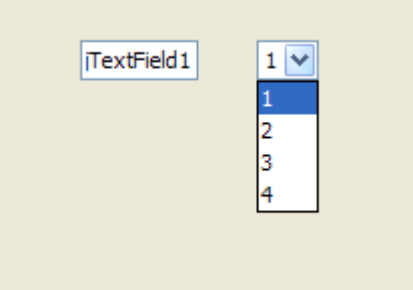
Pro pohodlí uživatele se v dolní části okna zobrazuje nápověda ke zvoleným vlastnostem. Vlastnosti, které byly manuálně změněny, se pro přehlednost zobrazují tučným písmem.

Ke stejným vlastnostem uživatel může získat přistup kliknutím pravým tlačítkem na komponentě a zvolením možnosti **Properties**.

4.4.4.2. Binding

Tato část okna **Properties** je užívaná k jednoduchému uvázání vlastností jednotlivých komponent bez použití listeneru nebo speciálního kódu. Nejlépe lze tuto vlastnost vysvětlit na konkrétním příkladě.

Například, je-li zapotřebí uvázat hodnotu zvoleného elementu v Comboboxu a hodnotu textového pole.



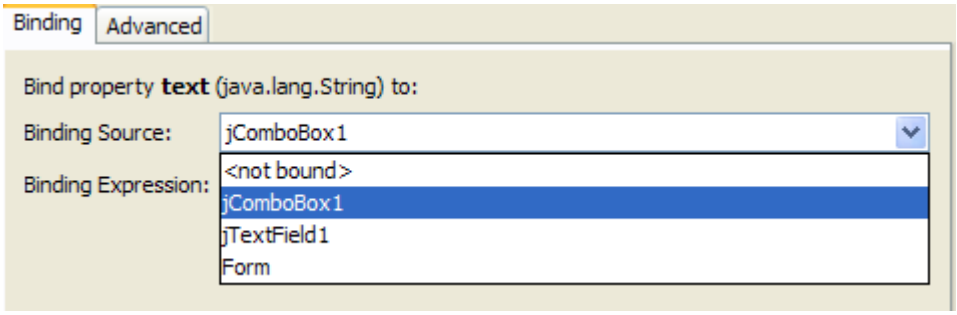
Obr. 28 Příklad vazby – objekty pro vazbu

Nejdříve je zvolen *zdroj* uvázání (binding) a jeho *objekt*. Zdroj uvázání je komponenta, ve které hodnota vlastnosti původně vzniká. Pro uvázaní komponent se v GUI Editoru nejdříve volí uvázání na objektu, a potom se definuje jeho zdroj[12].

[12] Ve zvoleném příkladu combobox uchovává předem definované hodnoty, proto tato komponenta bude využita jako zdroj uvázání.

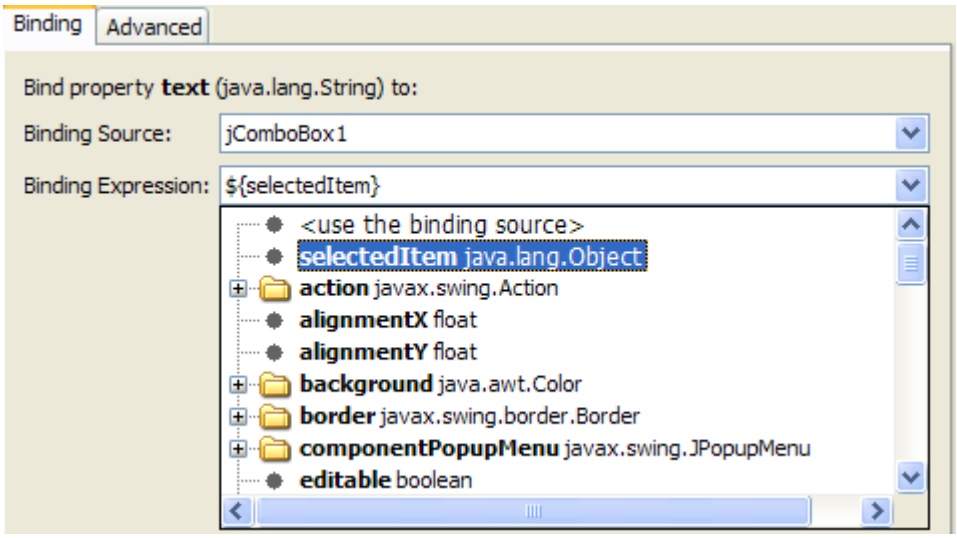
Postup pro uvázaní hodnoty comboboxu a hodnoty textového pole je pak následující:

1. Nejprve je zvolena komponenta textového pole a v okně **Properties** použito tlačítko **Binding**.
2. Následně je stisknuto tlačítko  naproti vlastnosti **text**.
3. V zobrazeném okně je zvolen zdroj uvázání



Obr. 29 Příklad vazby – volba zdroje

4. Dále je zvolena hodnota zdroje, ke které bude uvázána hodnota textového pole.



Obr. 30 Příklad vazby – hodnota zdroje

5. Ukončení nastavení vazby je pak potvrzeno stisknutím tlačítka OK.

Při výběru jakékoliv komponenty v checkboxu se pak v textovém poli odráží tato zvolená hodnota.

Obr. 31 Příklad vazby – výsledná vazba

V daném případě je vytvořená závislost oboustranná tj. nejen hodnota textového pole se mění pří změně hodnoty comboboxu, ale i naopak. Řada užitečných vlastností pak může být nastavena v sekci **Advanced –** pro více informace viz [1.10].
Podstatný je ten fakt, že výsledek použití Binding může být prohlížen již v náhledu návrhu, přičemž při použití Events nebo Connection mode je pro prohlednutí průběžných výsledků nezbytný build projektu.

4.4.4.3. Events

V této části okna **Properties** se zobrazují možné události vztahující se k dané komponentě, lze tak tedy určit, jak se bude komponenta při výskytu dané události chovat. Zdrojové objekty mohou spouštět události, na které budou reagovat objekty pomocí správců událostí.
V tomto okně je tedy možné ke každé události komponenty přiřadit určitého správce události a to následným postupem:

1. Vyberte požadovanou komponentu.
2. V okně **Properties** klikněte na tlačítko **Events**.
3. Zvolte událost v seznamu. Původní hodnota pro všechny události je <none>, tedy neurčená. Při výběru události se hodnota <none> automaticky změní na defaultní název události.

4. Klikněte na tlačítko  naproti názvu události a otevřete tak dialogového okno správce události.
5. Dodejte název správce události kliknutím na tlačítko **Add** a poté klikněte na **OK**.
6. Při potřebě dodání několika správců jedné komponentě, opakujte uvedený postup několikrát.

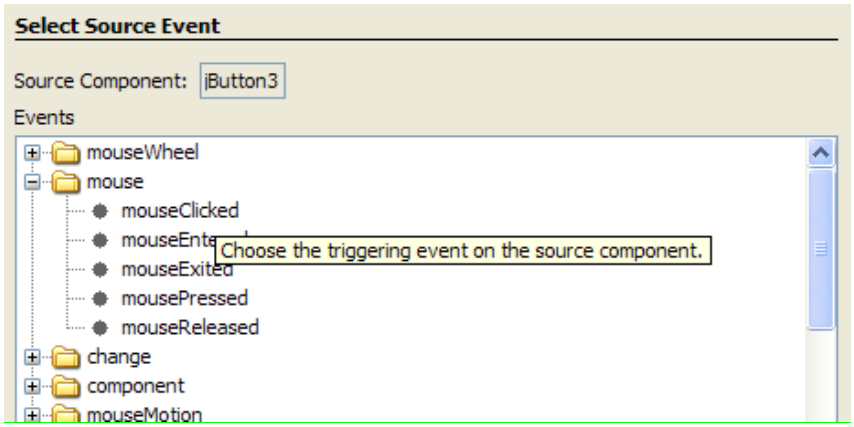
Automaticky pak bude vygenerován kód pro metodu se zvoleným názvem. Začátek a konec metody je chráněným kódem, ale kód v tělu metody je editovatelný a bude se spouštět pokaždé, když je vykonána zvolená událost.
Událost také muže být přiřazena ke komponentě i skrze kontextní menu (pravý klik na komponentu**->Events**). Události, které již byly definovány, jsou v seznamu označeny tučným písmem.

Pro **vymazání správce události** opět použijte okno **Properties**, podokno **Events** a za pomoci tlačítka  zobrazte dialogové okno správců události. Po té vymažte potřebné správce, nebo jednoduše vymažte název správce v oknu **Properties**. Po vymazání správce události je odpovídající blok kódu rovněž vymazán. Jestli-že je stejné jméno a tedy i stejný blok kódu používán několika správci, je pro vymazání tohoto bloku nutné, aby byly vymazány veškeré odkazy na tento blok.

4.4.4.4. Connection mode

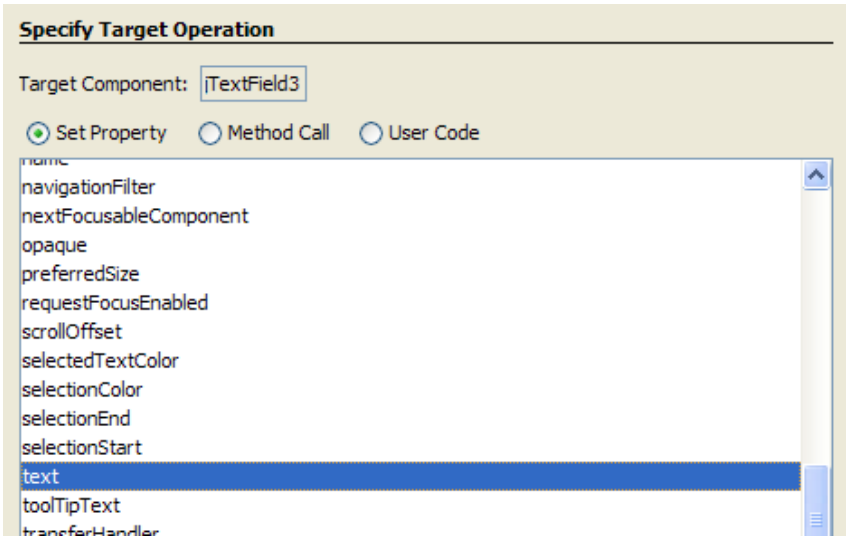
Alternativou k použití sekce **Events** je použití režimu spojení - **Connection mode**, i když to není přímo součásti okna **Palette**. Jedna se o způsob názornější a do jisté míry i pohodlnější.
Nejlépe lze použití tohoto režimu vysvětlit na konkrétním příkladě. Například, je-li zapotřebí nastavit změnu hodnoty textového pole po kliknuti na tlačítko, postup je následující:

1. Klikněte na tlačítko  pro spuštění režimu spojení.
2. Klikněte na první komponentu, akce které bude mít za následek změnu stavu (akci) druhé komponenty (komponenta bude zvýrazněna červeným rámečkem).
3. Po kliknutí na druhou komponentu se otevře okno s nabídkou událostí – Connection Wizard. (viz Obr. 32)



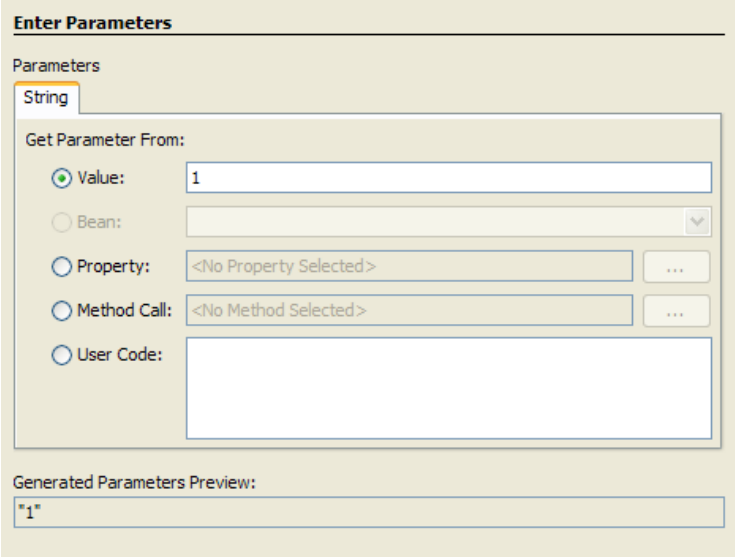
Obr. 32 Volba zdroje události

4. Vyberte potřebnou událost, na kterou bude reagovat druhá komponenta – v popisovaném příkladu to bude událost **mouseClicked**.
5. Vyberte parametr druhé komponenty, který se bude měnit – v popisovaném příkladu to bude parametr **text**. Na tomto kroku se dá také vybrat volání metody komponenty (***Metod call***) a také zadat vlastní kód (***User Code***) – viz Obr. 33.
6. Klikněte na **Next**.



Obr. 33 Volba cílové operace

7. V zobrazeném okně zadejte hodnotu parametru – to může být pevná hodnota, hodnota, zadaná za pomoci vlastností komponenty, pomocí volání metody, nebo vlastním kódem – v popisovaném příkladu bude zadána pevná hodnota „1“ – viz Obr. 34.
8. Klikněte na **Finish**.



Obr. 34 Zadání hodnoty parametru

Ve výsledku bude vygenerován kód (nechráněný), který způsobí změnu hodnoty textového pole po stisknutí tlačítka.
Pro ověření funkce tohoto kódu je nezbytné provést build projektu.

4.4.4.5. Code

Jak již bylo zmíněno v první kapitole, IDE automaticky generuje kód na základě manipulace uživatele s komponentami. Tento kód se nazývá chráněný kód, protože jej není možno jednoduše měnit v editoru, což zabraňuje jeho náhodné a chybné změně. Nicméně NetBeans GUI Builder poskytuje pohodlný způsob dodání vlastního kódu uživatele tohoto kódu, a to skrze podokno **Code** okna **Properties.** Tato možnost může být uplatněna pro každou použitou komponentu.

Za pomoci různých možností v podokně **Code** je možné:

1. Měnit název globální proměnné vygenerované pro tuto komponentu (**Variable Name**).
2. Měnit typ modifikátoru přístupu pro proměnnou (**Variable Modifiers**) – hodnoty *private*, *protected* a *public*.
3. Manuálně zadávat parametry proměnných (**Type Parameters**). Řetězec, zadaný do tohoto políčka, bude dodán k typu proměnné.
4. Zadávat deklarace proměnné jako lokální v metodě initComponents() (**Use Local Variable**).
5. Nastavovat zkrácenou volbu pro tlačítka a popisky pomocí znaku „&" v názvu místo nastavování názvů a vlastností *Mnemonics* zvlášť *(***Generate Mnenonics Code**). Použití této volby se ale nedoporučuje kvůli možné nekompatibilitě kódu.
6. Zadávat kód, který musí být vložen místo standardního výrazu new NazevTřidyKomponenty() (**Custom Creation Code**).
7. Volit typ generovaní kódu (**Code Generation**).
8. Volit soubor, do kterého má být komponenta serializovaná (**Serialize To**).
9. Dodávat zvláštní kód uživatele na různá místa automaticky generovaného kódu, a to dle pořadí zpracování kódu dané komponenty. Je tedy možné vsunout kód uživatele před (**Pre-**) a po (**Post-**) různé sekce kódu komponenty, kterými jsou:
   Pre/Post-Creation Code, Pre/Post-Init Code, Post Listeners Code, Pre/Post-Adding Code, After-All-Select Code, Pre/Post-Declaration Code.

Nápověda ke každému políčku je opět zobrazena v dolní části okna **Properties**.
Kód GUI formuláře je také možné editovat i mimo GUI Builder. Každý formulář IDE se skládá z dvou souborů:

1. Soubor **.java**, ve kterém se nachází zdrojový kód formuláře.
2. Soubor **.form** je soubor XML, ve kterém se nacházejí informace o formuláři v GUI Builder. Takový způsob uchovávání informací o formuláři je podstatně spolehlivější, než čtení informací přímo ze zdrojového kódu. Pro spuštění aplikace není soubor **.form** nutný, proto jej není nutné přikládat k archivu JAR. Nicméně po vymazání tohoto souboru již nebude možné použití nástroje GUI Builder pro editaci formuláře.

Soubor **.java** může být editován i za použití vnějších editorů, ale musí být striktně dodržována následující pravidla:

1. Nesmí se měnit obsah metody initComponents(). Tělo této metody se totiž vždy obnovuje při otevření formuláře v IDE.
2. Nesmí se vymazávat speciální komentáře, které byly dodány do zdrojového kódu GUI Builder (// GEN-...). Tyto komentáře jsou nezbytné pro správné otevření formuláře. Dávejte pozor na to, že v IDE Source Editory tyto komentáře nejsou vidět.
3. Nesmí se editovat záhlaví a zápatí správců událostí.

4.5. Rozmístění a spuštění aplikace

Pro spuštění aplikace je nutné vygenerovat složku **dist** , která je pak součástí podadresáře daného projektu: **Run ->Build main Project**. Výsledný vygenerovaný soubor **.jar** je pak spustitelným souborem aplikace.

Pro distribuci aplikace stačí jednoduše zabalit do archivu složku **dist** nacházející se v adresáři projektu. Tato složka rovněž zahrnuje soubor README s popisem veškerých detailů. Důležité však je, aby projekt měl main class (tj. spustitelnou část aplikace), jinak se při buildu soubor **dist/lib** nevytváří.

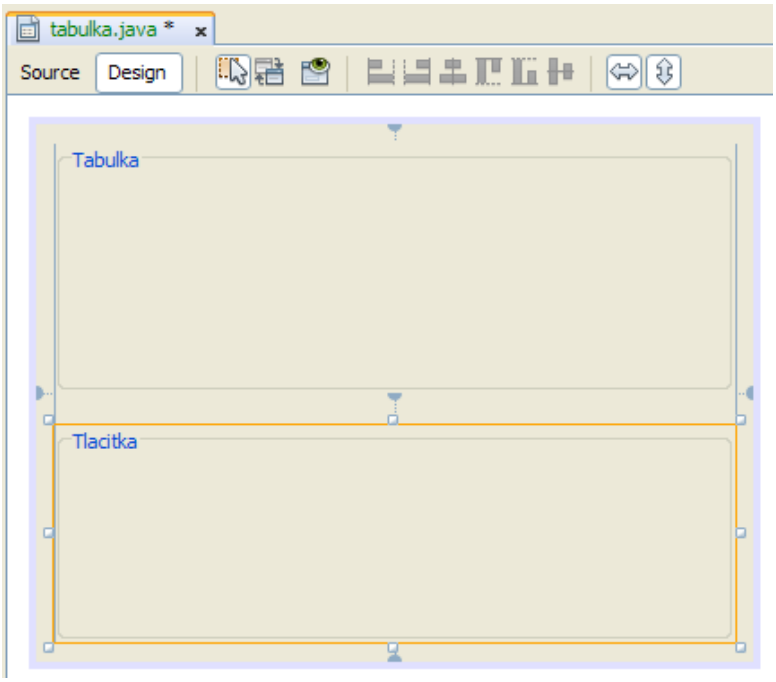
Pro manuální spuštění aplikace z příkazového řádku:

Přejděte do složky *dist* a napište následující příkaz: *java -jar <jar_name>.jar* , kde *<jar_name>* je název souboru.

5. Příklad aplikace s GUI

V této závěrečné kapitole je představen přiklad tvorby formuláře pro vkládání titulů knih do knihovnických tabulek. Příklad bude doplněn pro názornost screenshoty. [13]

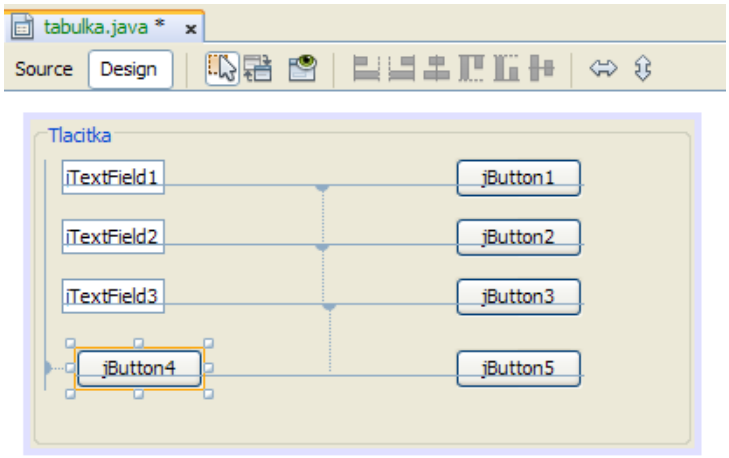
1. Vytvoříme projekt **Tabulka** a v něm Panel JFrame **Tabulka**.
2. Rozdělíme oblast návrhu na 2 časti pomocí komponent Panel a přiřadíme těmto panelům jejich názvy (**Properties ->Border -> Titled Border**) – viz Obr. 35.



Obr. 35 Dodání panelů do oblasti návrhu

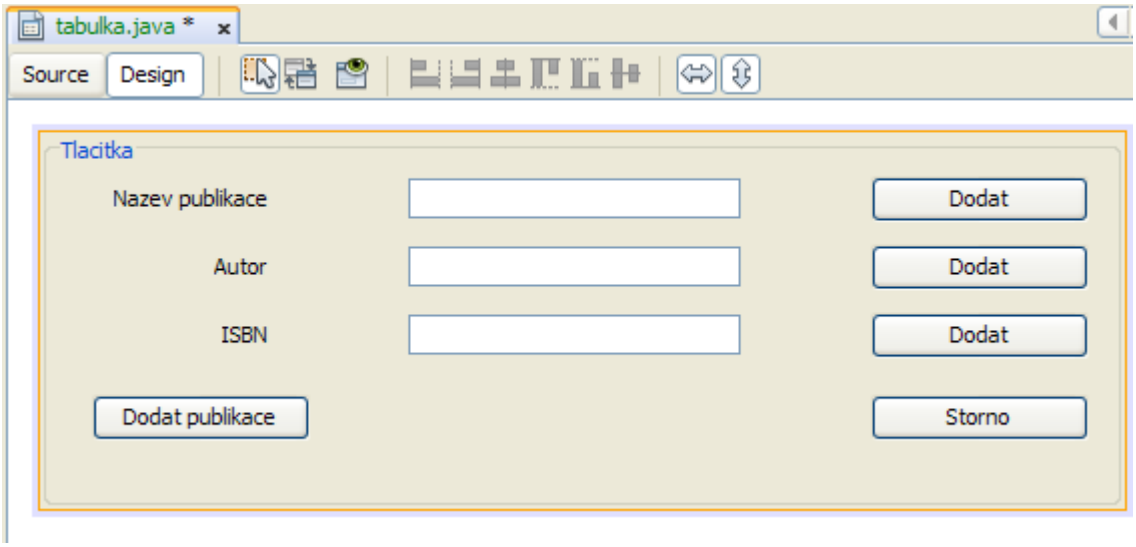
3. Do oblasti panelu **Tlacitka** dodáme textová pole a tlačítka pro budoucí zpracování nových publikací. Při umísťování komponent dodržujeme vodicí čáry, které navrhuje IDE, a k tlačítkům aplikujeme volbu **Same size** – viz Obr. 36.

[13] V tomto přikladu budou vesměs použity komponenty Swing.



Obr. 36 Postupné dodání komponent na jeden z panelů a jejich umisťování podle vodicích čár

4. Změníme názvy tlačítek a textových polí (možnost **Edit text**) – viz Obr. 37.

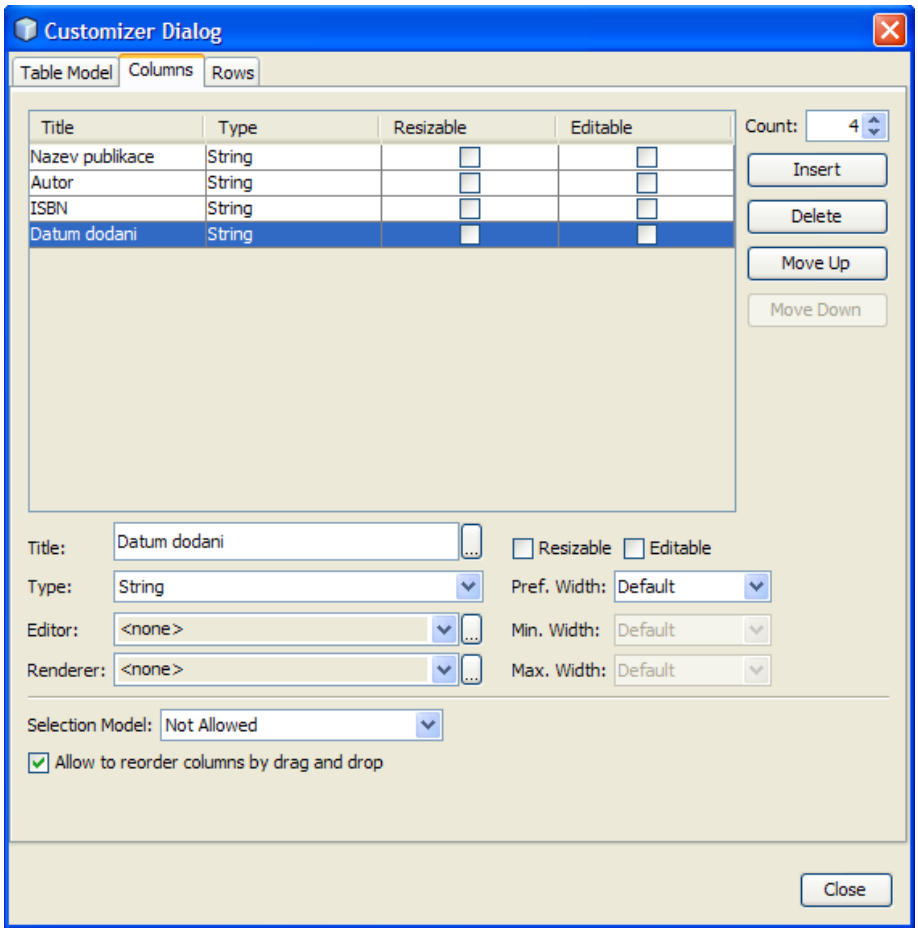


Obr. 37 Změna názvů komponent

5. Do oblasti **Tabulka** dodáme komponentu Table a zvolíme názvy sloupců (**Table Contents** -> **Columns**). [14] Jako název sloupce může být zvolen běžný text, Resource Bunde, hodnota existující komponenty, uživatelský kód (pro přístup k těmto možnostem použijte tlačítko  naproti poli **Title**). Dále může být zvolen typ názvu (Type) a také definována možnost změny jeho délky a editovatelnost (oblasti **Resizable** a **Editable**). Pro dodání/vymazání/změnu pořadí řádků mohou být také

[14] Editace této oblasti je možná jen při výběru Režimu tabulky (**Table Mode**) User specified. Při výběru jiného režimu tabulky může uživatel uvázat tabulku s jinou komponentou formuláře (**Bound**), definovat předání hodnoty z existující komponenty (**Value from Existing Component**) a dodat vlastní kód (**Custom Code**).

použita tlačítka zprava – **Insert**, **Delete**, **Move Up**. Pro dodání řádků může být také použit spinner **Count –** viz Obr. 38.

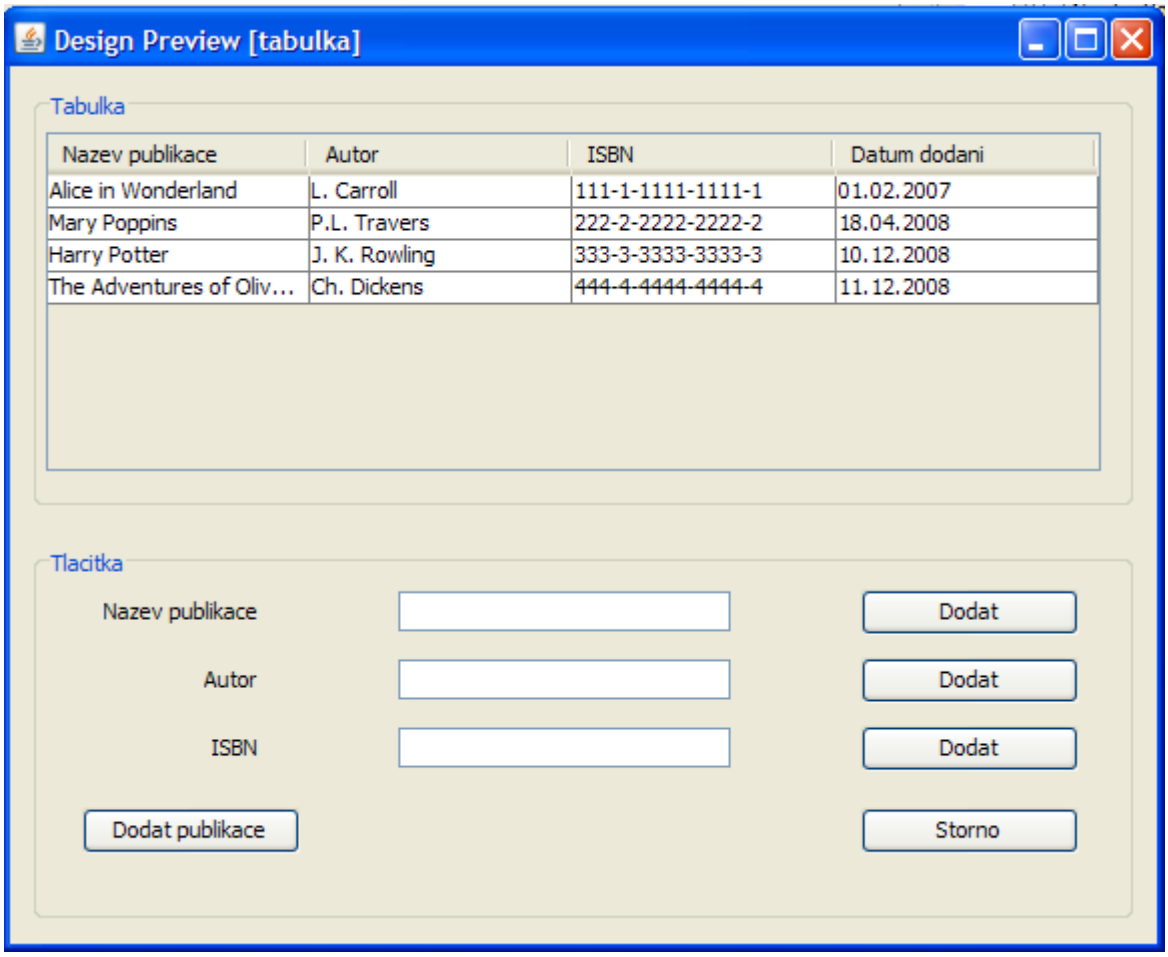


Obr. 38 Editace sloupců tabulky

6. Pro dodáni hodnot do tabulky použijeme možnost **Table Contents -> Rows**[15]**.** Dvojklikem na pole v tabulce jej otevřeme pro editaci.

Výsledek všech výše popsaných operací je zobrazen na Obr. 39. Uvázání funkčnosti textových polí, tlačítek a řádků tabulky předpokládá přímé programování ve zdrojovém kódu, a proto je mimo obsah této práce.

[15] Pro možnost editace hodnot v předcházejícím kroku musí být označen checkbox **Editable.**



Obr. 39 Návrh formuláře, vytvořený pomocí nástroje NetBeans GUI Builder.

6. Závěr

Tato bakalářská práce poskytuje čtenáři základní teoretické informace o vlastnostech a možnostech rozhraní NetBeans IDE GUI Builder a také návod na praktické užití tohoto nástroje při programování jednoduchých grafických aplikací.

Práce nepopisuje veškerou rozsáhlou funkcionalitu nástroje NetBeans IDE GUI Builder, nicméně po jejím prostudování a praktickém vyzkoušení postupů v ní popsaných, se čtenáři přiblíží logika nástroje GUI Builder a její použití a bude schopen i postupů explicitně v této práci nepopsaných.

Řada postupů, které jsou v práci popsány je výsledkem osobního zkoumání autorky a experimentování s nástrojem NetBeans IDE GUI Builder, protože popis těchto postupů v dostupné dokumentaci buď neexistuje, nebo je obtížně naleznutelný.

Ve výsledku tato práce může sloužit jako pomůcka při začátku studia nástroje NetBeans IDE GUI Builder a také pro nastínění směru dalšího rozvoje znalostí čtenáře v této oblasti. Pro prohloubení znalosti popisovaného nástroje je čtenář vyzýván pokračovat v samostatném zkoumaní nástroje NetBeans IDE GUI Builder a jeho užití při tvorbě rozmanitých grafických uživatelských rozhraní.

Seznam literatury a zdrojů

Internetové zdroje

1.1. FLETCHER, Josh. AWT vs Swing. Dostupný z WWW: <http://edn.embarcadero.com/article/26970>

1.2. MULLIGAN, Talley. GUI Building in NetBeans IDE 5.5. Dostupný z WWW: <http://netbeans.org/kb/55/quickstart-gui.html>

1.3. O'CONNER, John. Professional Swing UIs with NetBeans GUI Builder [cit. 2006-06-18]. Dostupný z WWW: <http://weblogs.java.net/blog/2006/05/18/professional-swing-uis-netbeans-gui-builder>

1.4. O'CONNER, John. Create Great-Looking GUIs With NetBeans IDE 5.5. Dostupný z WWW: <http://java.sun.com/developer/technicalArticles/tools/nb_guibuilder/>

1.5. PAVLICA Dusan. New GUI Builder (Matisse) Design Specification. [cit. 2005-07-27]. Dostupný z WWW: <http://ui.netbeans.org/docs/ui/guibuilder/index.html>

1.6. SWARTZ, Fred. Java Notes. NetBeans GUI Projects. Dostupný z WWW: <http://www.leepoint.net/notes-java/tools/netbeans/netbeans-GUI-projects.html>

1.7. The Java Tutorials. Lesson: Using the NetBeans GUI Builder. [cit. 2009-9-23]. Dostupný z WWW: <http://java.sun.com/docs/books/tutorial/javabeans/nb/index.html>

1.8. The Java Tutorials. A Visual Guide to Swing Components (Java Look and Feel).[cit. 2009-9-23]. Dostupný z WWW: <http://java.sun.com/docs/books/tutorial/ui/features/components.html>

1.9. The Java Tutorials. What is Swing?.[cit. 2009-9-23]. Dostupný z WWW: <http://java.polytechnic.edu.na/ui/overview/intro.html>

1.10. The Java Tutorials. NetBeans IDE Basics.).[cit. 2009-9-23]. Dostupný z WWW: <http://java.sun.com/docs/books/tutorial/uiswing/learn/netbeansbasics.html>

1.11. Application. Dostupný z WWW: <http://netbeans.org/kb/docs/java/gui-image-display.html>

1.12. NetBeans. Community. FaqUnexpectComponentMove. [cit. 2009-11-06]. Dostupný z WWW: <http://wiki.netbeans.org/FaqUnexpectComponentMove>

1.13. NetBeans IDE 5.5 GUI Builder Visual Feedback Legend. Dostupný z WWW: <http://netbeans.org/kb/55/quickstart-gui_legend.html>

1.14. Binding Beans and Data in a Desktop Application. Dostupný z WWW: <http://docs.huihoo.com/netbeans/6.0/kb/60/java/gui-binding.html>

Publikace

2.1. Nápověda k NetBeans IDE 6.5; dostupná z menu Help – Help Contents;

2.2. Buchalcevová, A. – Pitka L.: Vývojové prostředí NetBeans, skripta VSE, Praha 2007, ISBN 978-80-245-1206-8, 124 stran

2.3. Myatt A.; Pro Netbeans IDE 6 Rich Client Platform Edition; Apress, 2008, ISBN 978-1-4302-0439-8; 491 strana

TERMINOLOGICKÝ SLOVNIK

Termín	Anglický originál	Popis
Applet	Applet	program napsány v jazyce Java a umístěný na do stránky HTML. Je přenášen ze serveru a spouštěn na stráně klienta [2.1].
Grafické uživatelské rozhraní	GUI (Graphical User Interface)	typ uživatelského rozhraní používaný pro interakci s počítačem, které zapojuje do této interakce vedle textových elementů i elementy grafické. Akce jsou obvykle prováděny přímou manipulaci s grafickými objekty [2.1].
Hlavní metoda	Main method	v programovacím jazyce Java – metoda, kterou se program spouští [http://en.wikipedia.org/wiki/Main_function_(programming)]
Internacionalizace	Internationalization	proces, v rámci kterého aplikace se stává lokalizovatelná [1.5].
Kastomizace	Customization	přizpůsobování komponenty nebo metody potřebám uživatele [autor]
Rozvržení	Layout	grafické rozvržení komponentu v okně aplikaci [autor]
Lokalizovaná aplikace	Localized application	aplikace, která používá jazyk cílového místa použití a splňuje jeho kulturní očekávání s cílem poskytnutí pohodlnějšího a užitečnějšího uživatelského rozhraní [1.5].
Naslouchač	Listener	v jazyce Java nejvíc použivatelné správce události, po svém implementování na komponentách „naslouchají“ správám o určitých událostech na jiných komponentách [http://java.sun.com/docs/books/tutorial/uiswing/events/actionlistener.html].

Plugin	Plugin	software, který nepracuje samostatně, ale jako doplňkový modul jiné aplikace a rozšiřuje tak její funkčnost [http://cs.wikipedia.org/wiki/Plugin].
Servlet	Servlet	objekt Java, který dynamický zpracovává požadavky a vytvářena nich odpovědí [http://en.wikipedia.org/wiki/Java_Servlet]
Serializace	Serialization	převedení instanci objektu na posloupnost bitů a uložení jích na nějaké pevné úložiště, v objektovém programování se často jedná o rozhraní, které tuto funkci podporuje [http://cs.wikipedia.org/wiki/Serializace]
Vývojové prostředí - IDE	Integrated development environment	software usnadňující práci programátorů, většinou zaměřené na jeden konkrétní programovací jazyk. Obsahuje editor zdrojového kódu, kompilátor, případně interpret a většinou také debugger. [http://cs.wikipedia.org/wiki/ Integrated_development_environment]