Západočeská univerzita v Plzni
Fakulta aplikovaných věd
Katedra informatiky a výpočetní techniky

# Bakalářská práce

# Porovnání návrhářů grafického uživatelského rozhraní pro Javu

Plzeň, 2019 Jan Smolař

Místo této strany bude zadání práce

## PROHLÁŠENÍ

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

V Plzni dne 30. dubna 2019

Jan Smolař

## ABSTRACT

This Bachelor thesis is focused on graphical user interface in Java programming language. Available libraries and their elements for creating graphical user interface are explored in detail.

The testing of available graphical user interface designers for Java is the part of the thesis as well. The testing is done using multiple test scenarios, which are supposed to test properties and functionality of the selected designers. Result of each test scenario is evaluated by predefined criteria and compared with all the other designers. The goal is to find the best graphical user interface designer.

There were four integrated development environments found during the initial survey, which contained their own designer. These four development environments are Eclipse, NetBeans, IntelliJ IDEA a JDeveloper. They are tested by five increasingly difficult test scenarios and evaluated by five criteria. After the evaluation of all the results it was clear that the best graphical user interface designer in Java is part of the Eclipse development environment.

## ABSTRAKT

Bakalářská práce se zaměřuje na grafické uživatelské rozhraní v programovacím jazyce Java. Podrobně jsou prozkoumány dostupné knihovny a jejich prvky pro tvorbu grafického uživatelského rozhraní.

Součástí práce je také otestování dostupných návrhářů grafického uživatelského rozhraní pro jazyk Java. Testování je prováděno pomocí testovacích scénářů, které mají za úkol prověřit vlastnosti a funkcionalitu zvolených návrhářů. Výsledky jednotlivých scénářů jsou podle definovaných kritérií ohodnoceny a porovnány s ostatními návrháři. Cílem je nalézt nejlepšího návrháře grafického uživatelského rozhraní.

Během průzkumu vývojových prostředí byly nalezeny čtyři, které obsahují vlastního návrháře. Tyto čtyři vývojová prostředí jsou Eclipse, NetBeans, IntelliJ IDEA a JDeveloper. Testovány jsou pomocí pěti testovacích scénářů rostoucí složitosti a hodnoceny podle pěti kritérií. Po vyhodnocení všech výsledků je dosaženo závěru, že nejlepší návrhář grafického uživatelského rozhraní pro jazyk Java je součástí vývojového prostředí Eclipse.

# OBSAH

# 1 Úvod

Vytváření grafických uživatelských rozhraní, dále jen GUI (z anglického Graphical User Interface), je v dnešním programování samozřejmá věc. Většina novějších aplikací či programů obsahuje určitou formu grafické komunikace s uživatelem. Tvorba a návrh GUI se staly tedy nezbytnou součástí tvorby jakéhokoli softwaru, a proto se i tvůrci vývojových prostředí zaměřují na jejich urychlení a usnadnění. Výsledkem jejich práce jsou návrháře GUI pro desktopové aplikace integrované ve vývojových prostředích, na které se bude moje práce zaměřovat.

Podstatou této práce bude nalézt existující volně dostupné integrované návrháře vývojových prostředí v programovacím jazyce Java a porovnat jejich vlastnosti a funkcionalitu. Testování návrhářů GUI bude provedeno pomocí testovacích scénářů, které budou identické pro všechny návrháře GUI. Pomocí předem stanovených kritérií bude závěrem zvolen nejlepší návrhář GUI. Ostatní vlastnosti návrháře či vývojového prostředí neuvedené v seznamu kritérií nebudou mít na volbu žádný vliv.

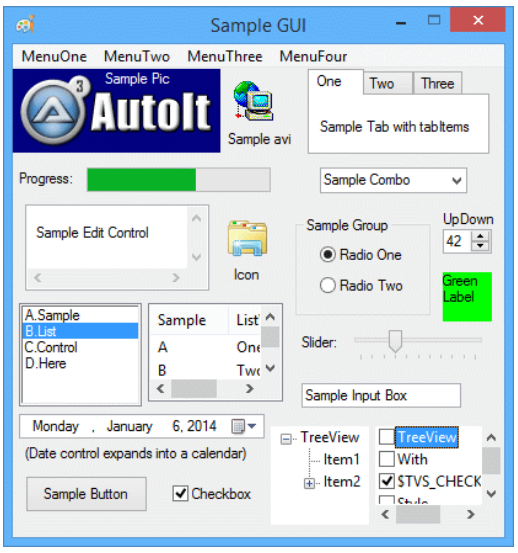
# 2 GRAFICKÉ UŽIVATELSKÉ ROZHRANÍ

Pojem grafické uživatelské rozhraní pochází z historického vývoje aplikací. Tento poměrně dlouhý název se často nepoužívá. Pokud chceme zmínit vzhled aplikace či druh komunikace s uživatelem, tak se spíše používá jeho anglická zkratka GUI.

Význam GUI je odražený v samotném názvu. Uživatelské rozhraní v názvu uvádí existenci komunikace s uživatelem, tedy způsob zobrazování informací či možnost uživatele měnit běh aplikace pomocí zadávání vstupů. Slovo grafické teprve odlišuje GUI od dalších typů uživatelských rozhraní, jelikož udává, že se jedná o uživatelské rozhraní založené na práci a interakci s grafickými prvky na obrazovce.

## 2.1 INTERAKCE S GUI

Základem každého operačního systému či aplikace využívající GUI je okno či skupina oken. V jednotlivých oknech jsou umístěny grafické prvky ke komunikaci s uživatelem, kterými jsou například tlačítka, seznamy, textová pole či menu panel, viz Obr. 2.1. Každý z grafických prvků má své vlastnosti a funkcionalitu, kterou podporuje. Uživatel může GUI ovládat pomocí myši, klávesnice nebo dotykového displeje. Klávesnice umožňuje ovládání pomocí klávesových zkratek a používá se k práci s textem, například k vyplňování textových polí. Myš se využívá často společně s klávesnicí a slouží hlavně k manipulaci s grafickými prvky. Nejčastěji se jedná o vybírání prvků, zaškrtávání polí a klikání na tlačítka. Novější technologií a v současnosti velkým trendem je ovládání pomocí dotykového displeje, který podporuje stejné funkce jako myš a klávesnice dohromady.




Obr. 2.1 Příklad GUI [1]

## 2.2 HISTORIE GUI

Počátky GUI se uvádějí v roce 1968 a za hlavní postavu v jeho vzniku se považuje Douglas Englebart. On a jeho tým se v poválečné éře zaměřili na komunikaci mezi počítačem a člověkem. Výsledkem jejich práce bylo představení většiny základních prvků současných informačních technologií na konferenci v San Franciscu v prosinci roku 1968. Mezi předvedenými prvky byly například okna, myš, síťování či video konference [2].

V roce 1973 byl vytvořen první počítač, který podporoval operační systém založený čistě na GUI pojmenovaný Xerox Alto. Obsahoval základní prvky od Douglase Englebarta, které zakomponoval do svého nového operačního systému. Díky velkému zájmu bylo vytvořeno přibližně 2000 těchto, v té době velmi drahých, strojů, které už se daly nazývat osobním počítačem. Charles P. Thacker byl odměněn za práci na tomto počítači také Turingovou cenou v roce 2009 [2].

Xerox Alto byl také základem pro vznik mnohem známějšího počítače, kterým byl Apple Macintosh od firmy Apple. Ten byl světu představen v roce 1984 a ihned se stal velkým úspěchem. Hlavním pokrokem byly překrývající se okna, složky, menu či koš. Počítač byl cenově dostupný a vytvořené GUI bylo snadno pochopitelné a použitelné, což umožnilo většímu počtu uživatelů počítač využívat [3].

V roce 1990 přišla také softwarová společnost Microsoft s operačním systémem založeným na komunikaci s uživatelem pomocí GUI. Operační systém byl pojmenován Windows 3.0, který byl následníkem operačního systému MS DOS založeného na komunikaci s uživatelem pomocí příkazové řádky. Windows 3.0 byl velmi úspěšný a stal se standartní součástí tehdejších počítačů. Nahrazen byl operačním systémem Windows 3.1 a následně Windows 95 [3].

# 3 GRAFICKÉ UŽIVATELSKÉ ROZHRANÍ V JAVĚ

V jazyce Java existuje několik knihoven, které umožňují vytvářet GUI. Běžně využívané knihovny v minulosti a současnosti jsou podrobněji popsány v následujících podkapitolách.

## 3.1 ABSTRACT WINDOW TOOLKIT

Abstract Window Toolkit, dále jen AWT, je knihovna grafických uživatelských prvků pro programovací jazyk Java. Vytvořena byla pro jednodušší přenositelnost aplikací mezi platformami, která do té doby byla složitá a také velmi nákladná. S řešením přišla firma Sun Microsystems v roce 1995, kdy světu představila knihovnu AWT [4].

Základní a největší inovací byla již dříve zmíněná přenositelnost aplikací a programů mezi různými platformami. Přenositelnost byla dosažena vytvořením rozhraní ke GUI operačních systémů. Výsledkem bylo vytvoření GUI aplikace, které při spuštění na určité platformě využívalo nástrojů a prvků GUI dané platformy. Například spuštění na operačním systému Windows má za následek, že všechny prvky GUI vypadají naprosto identicky jako prvky GUI operačního systému Windows [4].

Dalším vylepšením bylo v AWT vytvoření robustního systému na zpracovávání a zachytávání událostí. Události je možné vyvolávat pomocí interakce s určitými komponentami, jako jsou tlačítka, seznamy a další. U jednotlivých komponent bylo možné měnit jejich velikost, pozici, barvu, písmo a mnoho dalších vlastností, které umožňovaly úpravu vzhledu podle vlastních potřeb [5].

### 3.1.1 KOMPONENTY AWT

Komponenty, které se v AWT vyskytují, jsou popsány v následujících podkapitolách. Jedná se převážně o zcela základní komponenty, a proto se většina vyskytuje i v dalších knihovnách.

#### LABEL – NÁVĚŠTÍ

Komponenta `Label`, viz Obr. 3.1, by se dala popsat jako označení či popisek, jelikož nemá žádný jiný účel a význam než pojmenování oblastí či zobrazení poznámek [6]. Její text nelze označit či změnit, a proto se často využívá ve spojení s komponentami

`TextField` a `TextArea`, u kterých slouží k upřesnění zobrazovaných informací či uvedení očekávaného vstupu.

### Button – Tlačítko

Komponenta `Button`, viz Obr. 3.1, na rozdíl od komponenty `Label` už dokáže zachytit a zpracovat vstupy od uživatele v podobě stisknutí tlačítka. Reakce, kterou stisknutí tlačítka vyvolá, je zcela na tvůrci aplikace, AWT se pouze stará o zachycení události stisknutí tlačítka a spuštění příslušné metody [6].

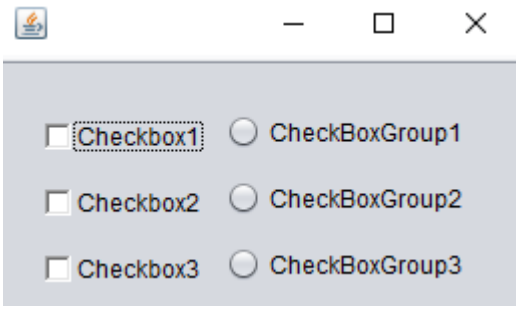

Obr. 3.1 AWT Label a AWT Button

### Checkbox – Zaškrtávací pole

Komponenta `Checkbox`, viz Obr. 3.2, představuje přepínač s dvěma polohami, a to zaškrtnuto nebo nezaškrtnuto. Komponenty typu `Checkbox` jsou na sobě zcela nezávislé, tedy lze zaškrtnout vícero možností bez jakéhokoli vlivu na ostatní. Komponenta také vyvolává událost při změně stavu a zároveň uchovává svůj aktuální stav v proměnné [6].

### CheckboxGroup

Objekt třídy `CheckboxGroup`, viz Obr. 3.2, je skupina komponent `Checkbox`. Při vložení komponent `CheckBox` do stejné skupiny začínají být jednotlivá zaškrtávací pole na sobě závislá a může být zaškrtnuté v jeden okamžik pouze jedno. Vytvořením `CheckboxGroup` se vzhled GUI nezmění, ale vznikne u každé komponenty `Checkbox` možnost patřit do nově vytvořené skupiny [6].



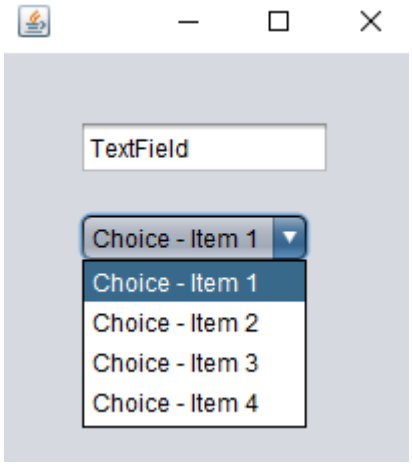

Obr. 3.2 AWT CheckBox a AWT CheckBoxGroup

### CHOICE – VÝBĚROVÝ SEZNAM

Komponenta `Choice`, viz Obr. 3.3, je vhodná, když je nutné uživatelem vybrat jednu z uvedených možností a není dostatek místa na vytvoření skupiny zaškrtávacích polí. Další výhodou, kromě úspory místa, je možnost dodatečně přidávat, ubírat nebo měnit položky seznamu bez jakéhokoli zásahu do vzhledu celkového GUI [7].

### TEXTFIELD – TEXTOVÉ POLE

Komponenta `TextField`, viz Obr. 3.3, se využívá ke čtení vstupu od uživatele nebo zobrazování výstupu určité operace. Text v poli lze libovolně upravovat až do chvíle, kdy je zmáčknuta klávesa `Enter` a vyvolána událost na zpracování vloženého textu [7].




Obr. 3.3 AWT Choice a AWT TextField

### LIST – SEZNAM

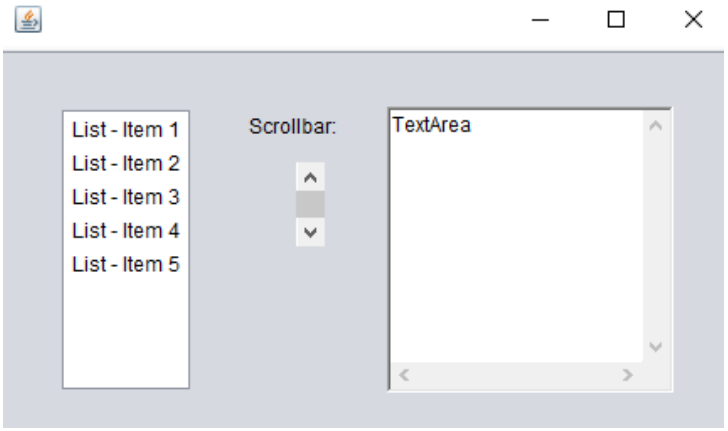
Komponenta `List`, viz Obr. 3.4, je podobná komponentě `Choice` v tom, že se jedná o výběr z možností. Seznam však dovoluje zobrazení a výběr více položek najednou. U výběrového seznamu lze vidět a zvolit pouze jednu vybranou položku, zatímco u seznamu lze vidět a zvolit více položek, třeba i všechny [7].

### SCROLLBAR – POSUVNÍK

Komponenta `Scrollbar`, viz Obr. 3.4, se většinou nepoužívá samostatně, ale je součástí ostatních komponent, jako je například dříve zmíněná komponenta `List` k zobrazení většího množství textu na malém prostoru. Většinou se přidává do komponent automaticky, ale je možné ji přidat i ručně. Může být horizontální či vertikální [7].

### TEXTAREA

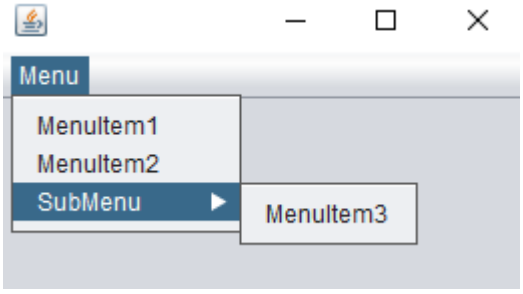
Komponenta `TextArea`, viz Obr. 3.4, je používána pokud potřebujeme pracovat s víceřádkovým textem. Počet řádek textu není nijak omezen, ale může se stát, že některé řádky textu nebudou zobrazené. Na rozdíl od komponenty `TextField` není událost generována stiskem klávesy `Enter`, ale jakoukoli změnou v textovém poli [7].




Obr. 3.4 AWT List, AWT Scrollbar a AWT TextArea

### MENUBAR

Komponenta `MenuBar`, viz Obr. 3.5, slouží k zobrazení a práci s menu panelem. Do menu panelu lze přidat tři druhy komponent, a to `Menu`, `MenuItem` a `CheckboxMenuItem`. `MenuItem` je individuální položka menu a `CheckboxMenuItem` je komponenta `Checkbox` umístěná v menu [6].




Obr. 3.5 AWT MenuBar

### 3.1.2 KONTEJNERY AWT

Kontejnery obsahují komponenty a oddělují je tak od sebe z logického či vzhledového důvodu nebo pro správné fungování aplikace. AWT obsahuje několik kontejnerů, které jsou uvedeny a popsány v následujících podkapitolách.

#### FRAME

Kontejner `Frame`, viz Obr. 3.6, je základní a nejpoužívanější kontejner, jelikož každá aplikace musí obsahovat alespoň jeden tento objekt. `Frame` je potomek třídy `Window`, a proto se jedná o objekt typu okna. Po vytvoření kontejneru `Frame` je nutné nastavit jeho základní vlastnosti, a to viditelnost a velikost, jelikož implicitně nejsou optimálně nastaveny [7].

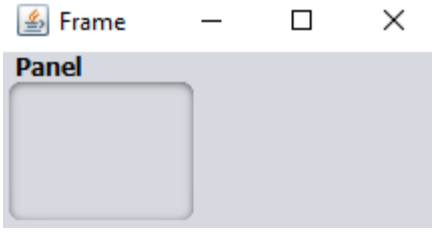
#### DIALOG

Kontejner `Dialog` je také potomek třídy `Window`. Používá se hlavně při vytváření dodatečných oken aplikace. Tato okna mohou sloužit k zobrazení důležitých informací nebo k vynucení určité akce od uživatele. Jednou z hlavních a využívaných vlastností je modálnost okna, tedy znemožnění přístupu k ostatním oknům aplikace při vytvoření okna, dokud není provedena akce, kterou okno vyžaduje [6].

Speciálním typem je kontejner `FileDialog`, který je zaměřený na práci se soubory. Okno je implicitně nastaveno jako modální a slouží k načítání a ukládání dat do souborového systému stroje, na kterém aplikace běží. Velkou odlišností od dalších kontejnerů je nemožnost umístit do něho další komponenty [7].

#### PANEL

Kontejner `Panel`, viz Obr. 3.6, slouží k oddělení komponent, ať už z důvodu logického či optického. Jedná se o sdružení komponent, které uchovává jejich rozmístění nezávisle na vlastní pozici celého kontejneru `Panel` [7].




Obr. 3.6 AWT Frame a AWT Panel

#### SCROLLPANE

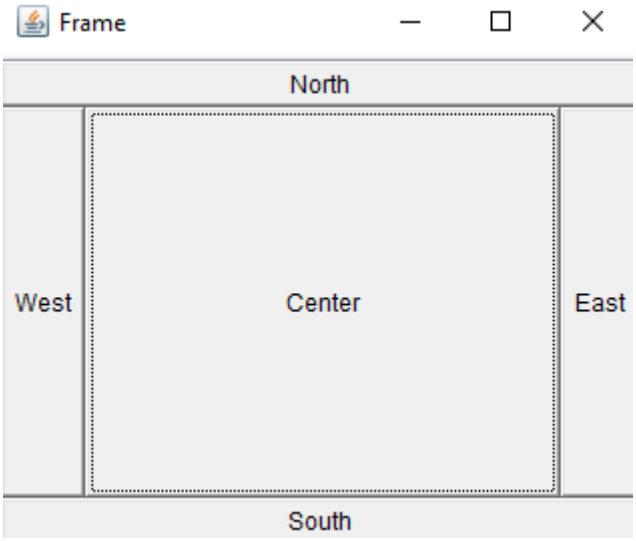
Kontejner `ScrollPane` je vzhledově podobný komponentě `TextArea`, jelikož se jedná o posouvací plochu. Výrazně se ale liší obsahem, jelikož `TextArea` obsahuje pouze text, ale `ScrollPane` obsahuje komponenty. Jeho použití není příliš časté, protože některé komponenty mohou být skryté, což není většinou optimální, ale někdy se jeho využití hodí například při nedostatku místa [7].

### 3.1.3 LAYOUTY AWT

Layouty jsou předem definovaná rozmístění komponent v určitém kontejneru. AWT nabízí možnost využití pěti layoutů, které jsou podrobněji popsány dále.

#### BORDERLAYOUT

`BorderLayout`, viz Obr. 3.7, rozděluje kontejner do pěti oblastí, které jsou pojmenované North, South, East, West a Center. Není nutné mít komponenty ve všech oblastech [8].




Obr. 3.7 BorderLayout

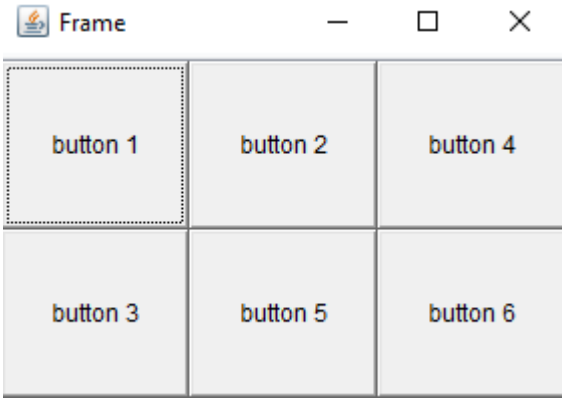
#### CARDLAYOUT

`CardLayout` bere jednotlivé komponenty v kontejneru jako karty. Kontejner je tedy takový balíček karet, kterým je možné listovat. Vždy je viditelná pouze jedna karta. První přidaná komponenta se bere jako vrchní karta. Tento layout není příliš často používaný, ale existují situace, kdy se může hodit. Příkladem takové situace může být potřeba

zobrazit různá GUI různým uživatelům. Obrázek není přiložen, protože by k porozumění layoutu nijak nepřispěl [8].

### GridLayout

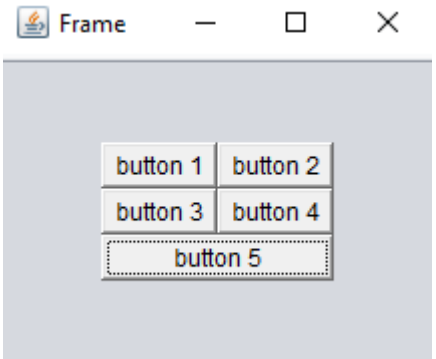
`GridLayout`, viz Obr. 3.8, rozděluje kontejner do pravidelné mřížky, kde každé políčko má stejnou velikost. Do každého políčka lze vložit pouze jednu komponentu. Počet sloupců a řádek je možné navolit nebo nechat vývojové prostředí rozhodnout [8].




Obr. 3.8 GridLayout

### GridBagLayout

`GridBagLayout`, viz Obr. 3.9, je nejvíce flexibilní z uvedených layoutů. Rozděluje kontejner do mřížky jako `GridLayout`, ale umožňuje měnit velikost jednotlivých políček a povoluje komponentám umístění na více než jednom políčku [8].

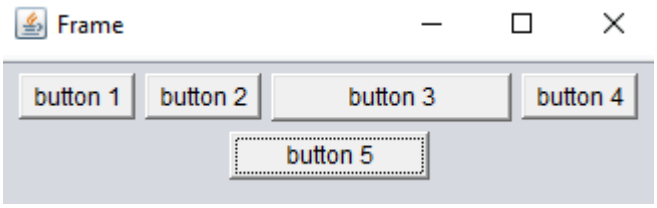



Obr. 3.9 GridBagLayout

### FlowLayout

`FlowLayout`, viz Obr. 3.10, umísťuje komponenty do řady a pokud by se další komponenta už do stejné řady nevešla, tak ji umístí do další. Layout si také dokáže velmi

dobře poradit se změnou velikosti kontejneru, jelikož všechny komponenty upraví a přemístí tak, aby byly optimálně rozmístěny v kontejneru s novou velikostí [8].




Obr. 3.10 FlowLayout

## 3.2 STANDARD WIDGET TOOLKIT

Standard Widget Toolkit, dále jen SWT, je další knihovna grafických uživatelských prvků pro programovací jazyk Java. SWT bylo vytvořeno jako další varianta k AWT a Swingu pro práci s grafickými prvky v Javě. Vytvořeno bylo firmou IBM, protože se jí zdály v té době knihovny AWT a Swing jako nedostačující. AWT byla závislá na platformě a Swing měl z počátku velké problémy s výkonem, který IBM nutně potřebovalo [9].

Knihovna podobně jako AWT používá k vykreslování komponent, neboli v SWT nazývané Widgets, grafické objekty daného operačního systému, na kterém právě běží. Obě knihovny však mají vlastní strukturu a způsob jakým tyto objekty využívají. SWT se na rozdíl od AWT může také pochlubit mnohem větším počtem komponent. SWT je v současnosti udržováno a využíváno korporací Eclipse Foundation ve vývojovém prostředí Eclipse. Přenositelnost mezi knihovnami nebyla z počátku vůbec dobrá, ale postupem času se zvyšovala a od verze Eclipse 3.0 je výrazně vylepšená [10].

### 3.2.1 KOMPONENTY SWT

Základ komponent SWT vznikl podle komponent konkurence, které dále rozvíjel. Všechny komponenty použitelné v AWT mají svého dvojníka i v SWT. Odlišnosti mezi nimi jsou pouze minimální a většinou jen vzhledové, často se jedná o totožnou komponentu pouze jinak pojmenovanou. Proto v tomto výpisu komponent SWT nebudu dále uvádět již zmíněné komponenty, které zůstaly stejné, ale zaměřím se spíše na ty, které jsou nové či výrazně změněné. Komponent v SWT bylo od začátku výrazně více než v AWT a v průběhu času se další postupně ještě přidávaly [11].

Zásadní změny obdržely položky panelu menu, u kterých vznikla výrazně větší rozmanitost a možnost větší úpravy jejich vlastností. Komponenta `Button` dostala také mnohem

větší možnost úpravy přidáním výběru typu tlačítka. Typy tlačítek jsou například tlačítko s polohami zapnuto a vypnuto, základní tlačítko s akcí při stisku či tlačítko šipky pro zobrazení více informací jiné komponenty. Rozšíření lze nalézt také v zobrazování informací, jelikož se zvětšil počet komponent k uspořádání a zpřehlednění zobrazení informací. Příkladem těchto komponent jsou komponenty `ExpandBar`, `ScrolledComposite` či `TabFolder`. Dále jsou zmíněny podrobněji komponenty, které nejsou v AWT vůbec obsaženy [11].

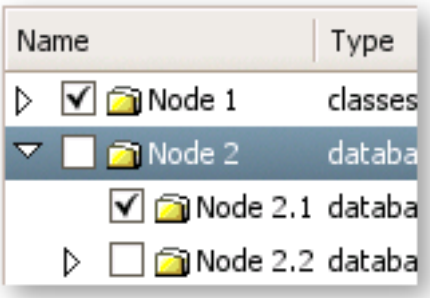
### BROWSER

Komponenta `Browser`, viz Obr. 3.7, slouží k zobrazení a manipulaci s HTML dokumenty. Oproti základním komponentám AWT je i tato komponenta velkým pokrokem a rozšířením [11].


Obr. 3.11 SWT Browser [11]

### TREE

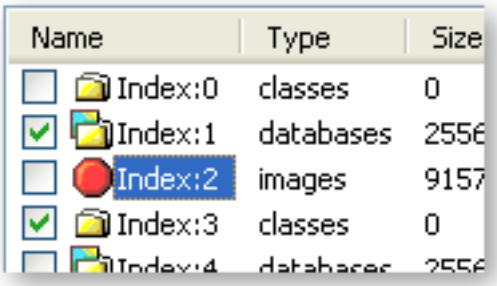
Komponenta `Tree`, viz Obr. 3.8, je používána k vytvoření seznamu prvků stromové struktury. Lze do ní přidávat samostatné prvky představující listy stromu či složky obsahující další prvky představující uzle stromu. Základní prvky stromu se mohou od sebe také lišit, což výrazně zvyšuje použitelnost této komponenty v reálných situacích [11].




Obr. 3.12 SWT Tree [11]

### TABLE

Komponenta `Table`, viz Obr. 3.9, je dalším typem komponenty pro zobrazení a procházení většího počtu prvků. Prvky lze seřadit podle jednotlivých vlastností. Práce s prvky v tabulce je přímočará a intuitivní, jelikož není odlišná od práce s tabulkou v jakémkoli jiném programu [11].




Obr. 3.13 SWT Table [11]

### DATETIME

Komponenta `DateTime`, viz Obr. 3.10, je v SWT určena k práci s datumy. Lze v ní snadno vybrat datum pomocí grafického zobrazení kalendáře. Výběrem určitého dne v zobrazeném kalendáři dojde i k automatickému převodu datumu na číselné zobrazení [11].


Obr. 3.14 SWT DateTime [11]

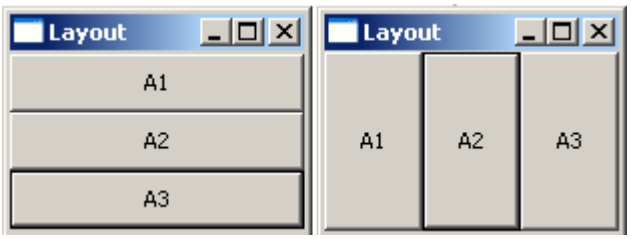
## 3.2.2 LAYOUTY SWT

SWT obsahuje pět layoutů. Layouty i v SWT jsou předem definovaná rozmístění komponent v určitém kontejneru, které jsou v této knihovně nazývány Composite. Pro jednoduchost budu nadále používat slovo kontejner pro objekt obsahující komponenty. Layout SWT s názvem `GridLayout` funguje naprosto stejně jako `GridBagLayout` obsažený v knihovně AWT. To platí i pro layouty `StackLayout`

v SWT a `CardLayout` v AWT. Zbylé tři layouty SWT jsou popsány podrobněji v následujících podkapitolách [12].

### FillLayout

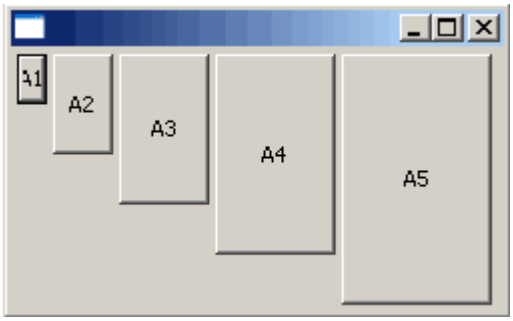
`FillLayout`, viz Obr. 3.15, je nejjednodušší layout, který umisťuje všechny komponenty do jednoho sloupce či řádku. Komponenty mají všechny stejnou velikost, která se mění s velikostí kontejneru [12].




Obr. 3.15 FillLayout [12]

### RowLayout

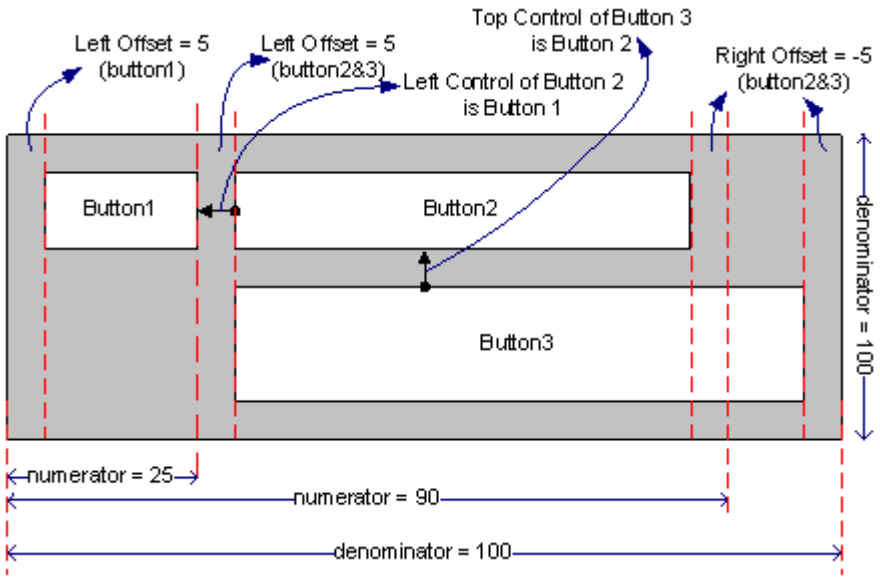
`RowLayout`, viz Obr. 3.16, umisťuje komponenty do řádků nebo sloupců. Oproti `FillLayout` však lze komponenty rozmístit do více sloupců či řádků. Vlastnosti layoutu jsou také výrazně rozšířeny. Lze nastavit komponentám různé velikosti, změnu velikosti komponent podle kontejneru či stálou pozici komponent v kontejneru [12].




Obr. 3.16 RowLayout [12]

### FormLayout

`FormLayout` je oproti ostatním layoutům více flexibilní. Předem se neudává velikost komponent, jelikož se sama mění podle aktuální velikosti kontejneru, ve kterém se nachází. Pevně se neudává ani pozice komponent, ale určují se pouze vztahy a závislosti jednotlivých komponent mezi sebou nebo s okraji kontejneru. Tyto vlastnosti jsou znázorněny na Obr. 3.17 [12].




Obr. 3.17 FormLayout [12]

## 3.3 SWING

Swing je následník AWT od firmy Sun Microsystems. Jedná se opět o knihovnu grafických uživatelských prvků na platformě Java. Swing je pokus o nápravu větších nedostatků nalezených u AWT jako například malý počet komponent. Hlavním rozdílem je nezávislost na platformě, kterou Swing dosáhl vlastním vykreslováním komponent bez využití vykreslování operačním systémem. Swing je vytvořen na architektuře Model-View-Controller, což ulehčuje práci při úpravě funkcionalit komponent [13].

### 3.3.1 KOMPONENTY SWING

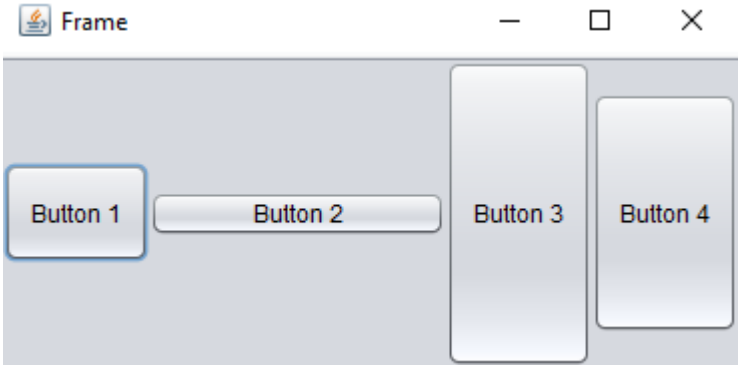
Komponenty Swingu jsou velmi blízké komponentám AWT. Rozdíly mezi nimi jsou výrazně menší než rozdíly mezi komponentami AWT a SWT. Hlavním rozdílem je jejich množství, vzhled a upravitelnost. Swing obsahuje výrazně větší počet komponent, které pokrývají skoro všechny požadavky uživatelů. Rozdíly ve vzhledu jednotlivých komponent jsou způsobeny již dříve zmíněnou nezávislostí Swingu na platformě. Všechny komponenty AWT jsou obsaženy také ve Swingu. Dokonce i všechny dříve zmíněné komponenty SWT mají svého zástupce ve Swingu. SWT a Swing obsahují velmi podobné, skoro naprosto stejné, komponenty, které se liší pouze svým vzhledem a názvem. Příkladem komponenty s totožnou funkcionalitou může být komponenta `JProgressBar` z knihovny Swing, která má svého dvojníka v komponentě `ProgressBar` z knihovny SWT, ukazující průběh či stav určité akce nebo komponenta `JTextPane` ze Swingu, jejíž duplicitou je komponenta `StyledText` v SWT, používaná k psaní formátovaného textu [14].

### 3.3.2 LAYOUTY SWING

Layouty knihovny Swing obsahují všechny layouty, které se vyskytovaly v knihovně AWT, a proto je nebudu opět podrobně popisovat. Zmíním se podrobněji v následujících podkapitolách o třech layoutech, které jsou ve Swingu zcela nové [15].

#### BOXLAYOUT

`BoxLayout`, viz Obr. 3.18, umožňuje umístit komponenty do jednoho sloupce či řádky. U jednotlivých komponent lze nastavit jejich maximální velikost a zarovnání [15].




Obr. 3.18 BoxLayout

#### GROUPLAYOUT

`GroupLayout` byl vytvořen hlavně pro použití v návrhářích GUI, ale je možné ho využít i při psaní GUI bez návrháře. Hlavní odlišností od ostatních layoutů je nezávislost horizontálního a vertikálního umístění všech komponent. Layout obsahuje ještě dvě uspořádání, a to sekvenční a paralelní. V podstatě se jedná u sekvenčního uspořádání o horizontální umístění a u paralelního uspořádání o vertikální umístění. Při vkládání nové komponenty lze vytvářet vazby na ostatní komponenty či na kontejner, do kterého komponentu přidáváme. Obrázek layoutu není uveden, protože neexistuje žádné obecné rozmístění komponent, které by bylo typické pro layout `GroupLayout` [16].

#### SPRINGLAYOUT

`SpringLayout` byl vytvořen také pro použití v návrhářích GUI. Layout je velmi flexibilní a dokáže nasimulovat chování ostatních layoutů. Umístění komponent je relativní a závisí na vztahu s jinou komponentou či kontejnerem. Layout může obsahovat právě jeden vztah s jiným objektem. Po přidání vztahu na další objekt se automaticky ruší vztah

s předchozím objektem. Obrázek není přiložen, protože by neměl žádnou vypovídající hodnotu a nepřispěl by k popisu layoutu [17].

## 3.4 JavaFX

JavaFX je knihovna používána k vytváření desktopových, mobilních a internetových aplikací. Jedná se o následníka knihovny Swing, ale nadále jsou podporovány obě knihovny. Knihovna byla vytvořena z nutnosti držet krok s dobou, protože se začalo vytvářet stále více webových aplikací a zároveň ovládání přecházelo k dotykovým displejům, které JavaFX už podporuje. Tyto funkce knihovna Swing nepodporovala, a proto se od ní začalo upouštět. Práci na JavaFX započal Chris Oliver a dokončila ji firma Sun Microsystems, která ji v roce 2008 vydala. Vývoj knihovny probíhal v programovacím jazyce JavaFX Script, ale od verze 2.0 je napsána v Javě. Hlavní výhodou JavaFX je stejně jako jejího předchůdce úplná nezávislost na platformě [18].

### 3.4.1 Komponenty JavaFX

Komponenty JavaFX jsou odvozené od komponent Swingu, a proto jsou si velmi podobné. Celkový počet komponent se nezměnil, ale pouze došlo k rozšíření a úpravě stávajících komponent. Každá komponenta obdržela další unikátní vlastnosti, které umožňují větší upravitelnost a přizpůsobení dané komponenty. Změnil se opět i vzhled všech komponent k udržení kroku s dobou a trendy, změna také pomohla k odlišení komponent JavyFX od ostatních.

### 3.4.2 Nové možnosti v tvorbě GUI

Oproti ostatním knihovnám obsahuje JavaFX zcela nové možnosti pro tvorbu GUI. Jednou z hlavních je možnost vytvořit GUI v jazyce FXML a jeho následné formátování pomocí kaskádových stylů, které umožňují úpravu či celkovou změnu vzhledu GUI bez zásahu do kódu v jazyce Java. Součástí knihovny JavaFX jsou také speciální efekty a animace, které lze jednoduše vytvořit a použít. Velkou změnou od předchozích knihoven je také možnost přidávat posluchače nejen ke komponentám, ale také k vlastnostem. To umožňuje reagovat na mnohem větší množství vstupů od uživatele, kterými může být například změna barvy, tvaru či jiné vlastnosti objektu. JavaFX také podporuje tvorbu 3D grafiky, což dále rozšiřuje její použití, kterým může být vývoj her či design objektů [19].

# 4 NÁVRHÁŘE GUI PRO JAZYK JAVA

Návrháře GUI pro jazyk Java, které jsou volně dostupné a použitelné, jsou čtyři. Jedná se o integrované návrháře známých vývojových prostředí, a to NetBeans, IntelliJ IDEA, Eclipse a JDeveloper. Eclipse se ještě trochu odlišuje od ostatních, a to tím že jeho návrhář GUI není součástí základní verze Eclipse, ale je nutné si do něho stáhnout plugin. Všechny výše zmíněné návrháře jsou zcela bezplatné, což bylo jediné kritérium při hledání návrhářů GUI pro jazyk Java. Všechny umožňují tvorbu GUI pomocí knihovny Swing a některé i pomocí jiných knihoven jako AWT či SWT. Dále se zaměřím na jednotlivé návrháře GUI a vývojové prostředí, které je poskytují.

## 4.1 NETBEANS

NetBeans je integrované vývojové prostředí vlastněné firmou Oracle Corporation. Počátky NetBeans jsou v České republice. Vše začalo jako studentský projekt s názvem Xelfi, který se postupně s rostoucí komunitou začal rozrůstat v první vývojové prostředí zaměřené na jazyk Java. Nyní je NetBeans vlastněné a sponzorované firmou Oracle, vlastníkem jazyku Java, a je uváděné jako oficiální vývojové prostředí pro jazyk Java [20].

### 4.1.1 NÁVRHÁŘ GUI

Návrhář GUI umožňuje vytvářet grafické rozhraní pomocí knihovny Swing. Lze také přidat komponenty AWT, ale jejich použití je limitované, a proto se skoro vůbec nevyužívá. Všechny dostupné komponenty AWT mají svého dvojníka ve Swingu. Tvorba GUI probíhá výběrem a umístěním komponent. Umístění komponent je zcela libovolné nebo lze využít jedno z předem definovaných rozmístění. Velkou výhodou je možnost změny či tvorby GUI díky jejich rychlosti přímo při komunikaci se zákazníkem. Návrhář také umožňuje využití vlastních zkratek pro dané komponenty pro lepší orientaci v kódu. Jednou z méně známých součástí návrháře je vizuální debugger, který umožňuje nezávisle na kódu testovat funkčnost GUI [21].

## 4.2 ECLIPSE

Eclipse je jedno z nejvíce rozšířených vývojových prostředí. Hlavní výhodou a lákadlem je jeho nesrovnatelná upravitelnost a rozšiřitelnost oproti konkurenci. Zakládá se totiž na systému pluginů, které si vývojáři mohou sami podle libosti vytvořit nebo použít některé z široké nabídky již vytvořených a dostupných pluginů. Počátky Eclipse se nachází

u firmy IBM, která se snažila sjednotit svá vývojová prostředí. V té době jich bylo několik a byly mezi sebou nekompatibilní, a proto bylo vymyšleno nové vývojové prostředí založené na již existujících. Výsledkem byl Eclipse, vývojové prostředí vytvořené speciálně pro komunikaci a sjednocení vývojových nástrojů [22].

### 4.2.1 WindowBuilder Pro

Eclipse nemá integrovaného návrháře GUI, ale lze si bezplatně stáhnout plugin s názvem WindowBuilder Pro, který poskytuje plnou funkcionalitu návrháře GUI. Umožňuje tvorbu komponent SWT, Swing i AWT. Kód vytvořený s použitím pluginu je zcela nezávislý na platformě a lze ho použít kdekoli. Plugin lze také využívat v dalších vývojových prostředích založených na Eclipse jako Rational Application Developer, Rational Software Architect či MyEclipse. Hlavní předností se často uvádí předem definovaná rozmístění komponent, která usnadňují a urychlují umisťování komponent a tvorbu jejich vzájemných vazeb. Proto i složitá okna s velkým počtem komponent jsou přehledná a srozumitelná. WindowBuilder Pro je neustále vyvíjen a vylepšován společně s Eclipsem samotným. Existují samozřejmě i další pluginy na tvorbu GUI, ale WindowBuilder Pro nemá v tomto směru konkurenci a je hlavní volbou pro navrhování GUI v jazyce Java [23].

## 4.3 IntelliJ IDEA

IntelliJ IDEA je velmi oblíbené a často využívané vývojové prostředí pro tvorbu v jazyce Java. Vyvíjí a udržuje jí společnost JetBrains a v současnosti je rozdělena na dvě verze, a to Community a Ultimate. Verze Community je verze použitá při následujícím testování, protože se jedná o verzi volně dostupnou. První verze IntelliJ IDEA byla vydána na začátku roku 2001 a ihned se usadila velmi vysoko na žebříčku vývojových prostředí pro Javu. Její hlavní předností, již od jejího vzniku, bylo vyspělá navigace v kódu a jeho celkové udržování. Podporuje také možnost přidání pluginů, ale ne v takové míře jako Eclipse [24].

### 4.3.1 Návrhář GUI

IntelliJ IDEA má vlastního integrovaného návrháře GUI. Oproti ostatním zde zmíněným vývojovým prostředím se IntelliJ IDEA zaměřuje znatelně menší měrou na návrháře GUI. Návrhář GUI není zde považován za přednost, ale spíše nutnost kvůli nátlaku a zaměření konkurence. Není moc podporován ani rozvíjen a ukazuje se to také na jeho funkcionalitě. Umožňuje pouze tvorbu GUI pomocí knihovny Swing a žádné jiné. Dokonce neobsahuje všechny prvky ani knihovny Swing, protože v návrháři nelze vytvořit například menu

panel. Návrhář také negeneruje celý Java kód pro spuštění v jakémkoli jiném vývojovém prostředí. Vygenerovaný kód se dělí na kód v jazyce XML, který popisuje vzhled okna, a kód v jazyce Java, který udává, co použité komponenty mají dělat. Na druhou stranu nabízí velký počet předem definovaných typů rozmístění komponent i s jedním čistě vlastním. Návrhář je však stále na takové úrovni, že práce v něm zkracuje celkový čas návrhu a tvorby GUI [25].

## 4.4 JDEVELOPER

JDeveloper je vývojové prostředí vytvořené firmou Oracle Corporation. Zaměřené je přednostně na jazyk Java, ale podporuje i další jazyky jako HTML či XML. Vytvořen byl pro usnadnění všech částí vývoje aplikací od psaní kódu, testování až po optimalizaci. Hlavní myšlenkou bylo ulehčení deklarací prvků a vylepšení vizuální stránky vývoje. Další produkt firmy Oracle Corporation je postaven na stejné platformě jako JDeveloper, a to SQL Developer. Hlavní předností je jeho zaměření na celý životní cyklus aplikace [26].

### 4.4.1 NÁVRHÁŘ GUI

JDeveloper obsahuje také integrovaného návrháře GUI. Návrhář GUI podporuje knihovnu Swing a v menším rozsahu i knihovnu AWT. Java kód se automaticky generuje při práci s komponenty a lze ho i ručně upravovat. Rozmístění komponent není od začátku nastaveno a je tedy zcela na uživateli, zda chce použít některé z předem definovaných rozmístění nebo si vytvořit rozmístění vlastní. Oproti ostatním návrhářům neobsahuje žádné speciální či nové nástroje, ale na druhou stranu ani za nimi v tomto ohledu nezaostává [27].

# 5 TESTOVÁNÍ NÁVRHÁŘŮ GUI

Pro podrobné otestování jsem se rozhodl využít všechny výše zmíněné, volně dostupné návrháře GUI. Jsou to integrované návrháře vývojových prostředí IntelliJ IDEA, NetBeans, JDeveloper a plugin WindowBuilder Pro pro prostředí Eclipse. Podmínka volné dostupnosti neboli bezplatné licence je použita proto, aby každý, kdo chce výsledného nejlepšího návrháře využít, tak mohl učinit bez nutného zakoupení licence.

Pro otestování návrhářů jsem využil pěti testovacích případů, dále jen scénářů. Každý scénář jsem vytvořil všemi zvolenými návrháři GUI. Pro všechny scénáře byly použity stejné obecné postupy, které jsou podrobněji popsány v Příloze A.

Jednotlivé scénáře se zaměřují na určité funkcionality, které by návrháře GUI měli obsahovat. Scénáře jsou seřazeny podle náročnosti a počtu komponent, které obsahují. Všechny scénáře jsou vytvořeny v jazyku Java s použitím knihovny Swing, a proto by měly být zcela realizovatelné pomocí zvolených návrhářů. Každý scénář obsahuje určité drobné detaily, které jsem se snažil při testování dodržovat. Příkladem je formát písma, počáteční stav komponent či velikost komponent vzhledem k ostatním.

Hodnocení návrhářů a jejich vzájemné porovnání je provedeno pomocí předem určených kritérií, aby testování bylo co nejvíce objektivní. Subjektivní pohled se nedá zcela odstranit, ale při posuzování a udělování hodnocení jsem se snažil nehledět na vlastní názory a pocity. Testování jsem prováděl bez dřívější práce s danými návrháři, a proto není moje hodnocení ovlivněno předchozími znalostmi a zkušenostmi.

## 5.1 KRITÉRIA

Pro porovnání návrhářů jsem zvolil následující kritéria:

1) Délka vygenerovaného kódu
2) Délka tvorby
3) Počet dostupných grafických prvků
4) Porovnání se vzorem
5) Kvalita vygenerovaného kódu

Délka vygenerovaného kódu udává počet řádek vygenerovaného kódu a je ukazatelem, jak si vývojové prostředí dokáže poradit s tvorbou nových komponent či úpravou stávajících. Naznačuje nám také, jestli návrhář při tvorbě GUI vytváří nadbytečný kód a jaké je jeho množství.

Délka tvorby je čas v minutách, který scénář vyžadoval k vytvoření v daném vývojovém prostředí. Skládá se ze dvou hodnot, kde první hodnota uvádí čas k vytvoření scénáře bez jakékoli předchozí zkušenosti s tvorbou a úpravou potřebných komponent. Druhá hodnota uvádí čas k vytvoření stejného scénáře ihned po jeho prvním dokončení. Délka tvorby je takto rozdělena, aby zobrazovala intuitivnost použití návrháře při první tvorbě scénáře a zároveň zobrazovala čas, který je potřebný k vytvoření stejného scénáře se znalostí daného návrháře.

Počet dostupných grafických prvků uvádí celkový počet použitelných nástrojů, komponent a kontejnerů v daném návrháři. Ukazují se tím dostupné možnosti vzhledu GUI, které návrhář dokáže vytvořit. Měření proběhlo jen jednou, protože naměřená hodnota je pro všechny scénáře stejná. V tabulkách výsledků jsou však hodnoty pro úplnost uvedeny u všech scénářů.

Porovnání se vzorem je známka udělená návrháři za podobu vytvořeného GUI a jeho vzoru. Známky, které lze obdržet, jsou 1, 2, 3, 4 a 5. Známka 1 vypovídá o naprosté identičnosti vytvořeného GUI se vzorem. Známka 2 uvádí, že vytvořené GUI obsahuje lehké odchylky od vzoru, tedy určité komponenty se trochu liší. Známka 3 je udělena pokud vytvořené GUI obsahuje komponentu, která se výrazně liší od jejího vzoru. Známka 4 udává, že určitá komponenta naprosto chybí. Známka 5 by se neměla s vybranými scénáři vyskytnout, ale pokud by se vyskytla, tak by to vypovídalo o tom, že návrhář nebyl schopný vytvořit více jak jednu nezbytnou komponentu pro daný scénář.

Kvalita vygenerovaného kódu uvádí, zda vygenerovaný kód dodržuje pravidla Javy pro pojmenovávání metod a proměnných, také zohledňuje přehlednost a vzhled vygenerovaného kódu. Hodnocení je provedeno opět pomocí udělování známek, ale zde se přidělují pouze známky 1,2 a 3. Známka 1 udává, že vygenerovaný kód splňuje pravidla Javy a je rozumně strukturovaný a přehledný. Známka 2 naznačuje, že pár pravidel bylo porušeno nebo je orientace ve vygenerovaném kódu složitá. Známka 3 vypovídá o naprostém ignorování pravidel Javy či o chybějící struktuře kódu.

## 5.2 SCÉNÁŘ 1


Obr. 5.1 Scénář 1 [28]

Scénář 1, viz Obr. 5.1, se zaměřuje především na základní prvky a komponenty každého GUI. Testuje obtížnost vytvoření primitivních oken a zároveň klade důraz na vygenerovaný kód, protože i ten by měl odrážet jednoduchost tohoto scénáře. Obsahuje pouze komponenty `Label`, `Button` a `TextField`, které se vyskytují skoro ve všech GUI. Zkouší se zde také úprava komponenty `Label`, a to změnou písma a jeho formátu.

### 5.2.1 NETBEANS

Tab. 5.1 Scénář 1 - NetBeans

| Kritérium | Popis kritéria | Hodnocení |
|---|---|---|
| 1 | délka kódu | 216 řádek |
| 2 | délka tvorby | 9:27 minut |
| | | 4:19 minut |
| 3 | počet dostupných prvků | 64 prvků |
| 4 | porovnání se vzorem | 1 |
| 5 | kvalita kódu | 2 |

Kód obsahuje velké množství komentářů, které ho značně roztahují a prodlužují. Na druhou stranu se zde nenachází žádný nadbytečný kód. Podle kritéria 2 lze vidět, že práce při znalosti způsobu změny písma a umístění komponent je více než dvakrát

rychlejší. Počet dostupných komponent byl v tomto případě zanedbatelný kvůli jednoduchosti celého scénáře. Výsledek je naprosto identický se vzorem, a proto dostal ohodnocení 1 u kritéria 4, jak je uvedeno v Tab. 5.1, a ze stejného důvodu zde není uveden výsledný obrázek vytvořeného okna. Vygenerovaný kód splňuje standardy Javy, ale umístění komponent je velmi složitě a nepřehledně popsáno, což se odráží u hodnocení kritéria 5.

### 5.2.2 JDEVELOPER

Tab. 5.2 Scénář 1 - JDeveloper

| Kritérium | Popis kritéria | Hodnocení |
|---|---|---|
| 1 | délka kódu | 268 řádek |
| 2 | délka tvorby | 8:21 minut |
| | | 4:48 minut |
| 3 | počet dostupných prvků | 48 prvků |
| 4 | porovnání se vzorem | 1 |
| 5 | kvalita kódu | 2 |

Vygenerovaný kód je poměrně dlouhý na takto poměrně jednoduchý scénář. Časy u kritéria 2, které jsou uvedeny v Tab. 5.2, opět ukazují rozdíl při seznamování se s návrhářem a jeho opětovným použitím. JDeveloper nepodporuje komponenty AWT, což se promítá do jeho celkového počtu dostupných komponent. Vzhledem k jednoduchosti scénáře to není překvapující, ale vytvořené okno bylo opět naprosto identické vzoru.

### 5.2.3 ECLIPSE

Tab. 5.3 Scénář 1 - Eclipse

| Kritérium | Popis kritéria | Hodnocení |
|---|---|---|
| 1 | délka kódu | 114 řádek |
| 2 | délka tvorby | 8:35 minut |
| | | 4:30 minut |
| 3 | počet dostupných prvků | 61 prvků |
| 4 | porovnání se vzorem | 1 |
| 5 | kvalita kódu | 1 |

Plugin WindowBuilder Pro generuje krátký, přehledný a dobře strukturovaný kód. Neobsahuje žádné zbytečné řádky kódu. V časech, uvedených v Tab. 5.3, se opět promítá

první seznámení s novým návrhářem. Počet dostupných grafických prvků, výsledné okno i kvalita kódu jsou na skvělé úrovni.

### 5.2.4 IntelliJ IDEA

Tab. 5.4 Scénář 1 - IntelliJ IDEA

| Kritérium | Popis kritéria | Hodnocení |
|---|---|---|
| 1 | délka kódu | 39/196 řádek |
| 2 | délka tvorby | 9:47 minut |
| | | 4:59 minut |
| 3 | počet dostupných prvků | 27 prvků |
| 4 | porovnání se vzorem | 2 |
| 5 | kvalita kódu | 1 |

Kvůli návrháři od IntelliJ IDEA, který generuje kód v jazyce Java a XML, jsem uvedl u kritéria 1 oba počty řádek vygenerovaného kódu, jak je uvedeno v Tab. 5.4, kde první číslo udává počet řádek kódu v jazyce Java a druhé číslo počet řádek kódu v jazyce XML. Tato funkcionalita návrháře GUI samozřejmě ovlivňuje i kritérium 5 a já dále budu hodnotit jenom část kódu napsanou v Javě, aby se kvalita kódu dala porovnávat s ostatními vývojovými prostředími. Je také na první pohled vidět, že počet dostupných komponent je zde velice omezen. Návrhář GUI také nepodporuje zcela volné umisťování komponent do prostoru a je nutné využít jedno z předem definovaných rozmístění, což se promítá do hodnocení kritéria 4. Rozdíl se vzorem není nijak znatelný, ale při úpravě velikosti okna se výrazně zvětšuje. Nejedná se o nic velkého, a proto je kritérium 4 ohodnoceno známkou 2.

## 5.3 Scénář 2


Obr. 5.2 Scénář 2 [29]

Tento scénář, viz Obr. 5.2, se zaměřuje na opakovanou tvorbu a úpravu stejné komponenty. Zároveň se testuje zarovnávání více komponent a jejich vzájemné závislosti na sobě. Používají se zde už trochu pokročilejší komponenty jako `MenuBar` či `MenuItem`. Nejčastěji použitou komponentou v tomto scénáři je ale komponenta `Button`.

### 5.3.1 NetBeans

Tab. 5.5 Scénář 2 - NetBeans

| Kritérium | Popis kritéria | Hodnocení |
|---|---|---|
| 1 | délka kódu | 550 řádek |
| 2 | délka tvorby | 16:17 minut |
| | | 10:36 minut |
| 3 | počet dostupných prvků | 64 prvků |
| 4 | porovnání se vzorem | 1 |
| 5 | kvalita kódu | 2 |

Návrhář NetBeans i s tímto scénářem neměl žádný problém a výsledné okno je opět zcela identické vzoru. Na časech, uvedených v Tab. 5.5, se promítnulo potřebné formátování a uspořádání všech komponent `Button`. Počet řádek se oproti scénáři 1 také výrazně zvětšil, což bylo opět způsobeno velkým počtem komponent `Button` a jejich posluchači, kteří tvořili značnou část kódu.

### 5.3.2 JDeveloper

Tab. 5.6 Scénář 2 - JDeveloper

| Kritérium | Popis kritéria | Hodnocení |
|---|---|---|
| 1 | délka kódu | 567 řádek |
| 2 | délka tvorby | 17:59 minut |
| | | 11:13 minut |
| 3 | počet dostupných prvků | 48 prvků |
| 4 | porovnání se vzorem | 1 |
| 5 | kvalita kódu | 2 |

Návrhář JDeveloperu si u tohoto scénáře vedl velmi podobně jako návrhář NetBeans. U všech kritérií jsou výsledky návrhářů obdobné. Trocha odlišnosti, jak je uvedeno v Tab. 5.6, se nachází pouze u kritéria 5, kde známka 2 má trochu jiný důvod. Zde je kritérium 5

hodnoceno známkou 2 kvůli velmi obtížnému a zbytečně roztaženému spouštěcímu kódu, který obsahuje přibližně 40 řádek kódu pouze na odchytnutí a zalogování čtyř výjimek.

### 5.3.3 ECLIPSE

Tab. 5.7 Scénář 2 - Eclipse

| Kritérium | Popis kritéria | Hodnocení |
|---|---|---|
| 1 | délka kódu | 268 řádek |
| 2 | délka tvorby | 11:17 minut |
| | | 6:45 minut |
| 3 | počet dostupných prvků | 61 prvků |
| 4 | porovnání se vzorem | 1 |
| 5 | kvalita kódu | 1 |

Vygenerovaný kód je opět velmi přehledný a stručný, což se promítá i do celkem nízkého počtu řádek, jak je uvedeno v Tab. 5.7. Výsledné okno se shoduje se vzorem. Ruční úprava kódu či dopisování funkčního kódu do tlačítek by bylo velmi jednoduché, jelikož každá komponenta má všechny informace přímo pod sebou na jednom místě.

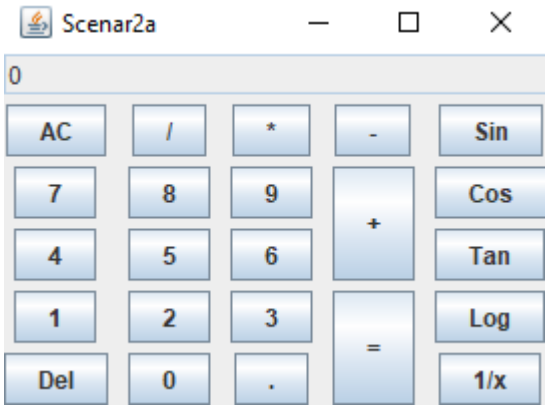
### 5.3.4 INTELLIJ IDEA

Tab. 5.8 Scénář 2 - IntelliJ IDEA

| Kritérium | Popis kritéria | Hodnocení |
|---|---|---|
| 1 | délka kódu | 179/208 řádek |
| 2 | délka tvorby | 11:11 minut |
| | | 11:53 minut |
| 3 | počet dostupných prvků | 27 prvků |
| 4 | porovnání se vzorem | 4 |
| 5 | kvalita kódu | 1 |

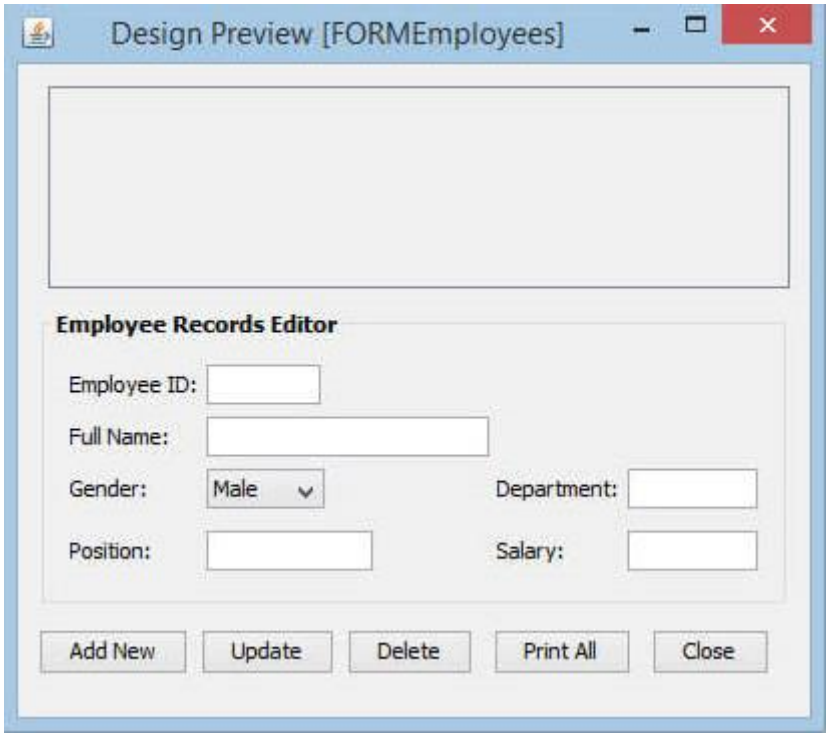
Proti scénáři 1 můžeme vidět značný nárůst počtu řádek v Javě, který je opět způsoben velkým počtem tlačítek. Známka 4 u kritéria 4, jak je uvedeno v Tab. 5.8, je z důvodu chybějících komponent `MenuBar` a `MenuItem`, které v tomto návrháři nelze vytvořit, a výsledkem je chybějící horní lišta. Nutnost použít předem definované rozmístění komponent má také za následek vlastní úpravu velikostí komponent. Při delším pojmenování komponent se komponenty sami zvětšují a je proto skoro nemožné vytvořit všechny komponenty `Button` stejné velikosti, viz Obr. 5.3. Práce s předem definovaným rozmístěním komponent se také promítla do naměřených časů, na kterých je vidět, že

při druhém měření nedošlo k žádné úspoře času. To bylo způsobeno neznalostí a nezkušeností s prací s daným rozmístěním komponent, které se promítlo stejně do obou měření.




Obr. 5.3 Scénář 2 - IntelliJ IDEA

## 5.4 Scénář 3




Obr. 5.4 Scénář 3 [30]

Scénář 3, viz Obr. 5.4, už obsahuje rozmanitější druhy komponent. Takto navržené a vytvořené okno už by se dalo využít v reálné situaci u skutečné aplikace. Jedná se o okno pro manipulaci se záznamy o zaměstnancích. Už zde existuje určitá struktura okna jako logické rozdělení okna na sekce. Novou komponentou je u scénáře 3 komponenta `ComboBox` a poprvé je zde také použit kontejner `Panel`.

### 5.4.1 NetBeans

Tab. 5.9 Scénář 3 - NetBeans

| Kritérium | Popis kritéria | Hodnocení |
|---|---|---|
| 1 | délka kódu | 355 řádků |
| 2 | délka tvorby | 13:53 minut |
| | | 6:15 minut |
| 3 | počet dostupných prvků | 64 prvků |
| 4 | porovnání se vzorem | 1 |
| 5 | kvalita kódu | 2 |

Na první pohled je opět vidět velký rozdíl v časech u kritéria 2, uvedených v Tab. 5.9, který je způsoben prací s novými prvky návrháře. Kritérium 4 je ohodnoceno známkou 1, protože mezi vzorem a vytvořeným oknem nejsou žádné rozdíly. Kritérium 5 obdrželo opět známku 2 kvůli neintuitivní strukturovanosti kódu a jeho celkové nepřehlednosti u rozmisťování komponent. Upravovat rozmístění komponent ve vygenerovaném kódu ručně by bylo velmi náročné.

### 5.4.2 JDeveloper

Tab. 5.10 Scénář 3 - JDeveloper

| Kritérium | Popis kritéria | Hodnocení |
|---|---|---|
| 1 | délka kódu | 374 řádek |
| 2 | délka tvorby | 10:23 minut |
| | | 6:23 minut |
| 3 | počet dostupných prvků | 48 prvků |
| 4 | porovnání se vzorem | 1 |
| 5 | kvalita kódu | 2 |

Se scénářem 3 si návrhář JDeveloperu poradil také velmi slušně. Výsledné okno je naprosto identické se vzorem, a proto je u kritéria 4 známka 1, jak je uvedeno v Tab. 5.10. Počet řádek je rozumný, přihlédneme-li k zvýšené složitosti scénáře. Známka 2 u kritéria 5 je z důvodu složité orientace ve vygenerovaném kódu. Kód vypadá, že se generuje v pořadí tvorby či úpravy komponent, což by pro případnou ruční úpravu kódu nebylo optimální.

### 5.4.3 ECLIPSE

Tab. 5.11 Scénář 3 - Eclipse

| Kritérium | Popis kritéria | Hodnocení |
|---|---|---|
| 1 | délka kódu | 165 řádek |
| 2 | délka tvorby | 11:08 minut |
| | | 6:20 minut |
| 3 | počet dostupných prvků | 61 prvků |
| 4 | porovnání se vzorem | 1 |
| 5 | kvalita kódu | 1 |

Práce s návrhářem v Eclipse byla i u tohoto scénáře bezproblémová. Kritéria 4 a 5 obdrželi nekompromisně známku 1, jak je uvedeno v Tab. 5.11. Výsledek vypadá jako vzor a kód by bylo možné ručně upravit díky dobré struktuře generovaného kódu, který zároveň splňuje všechny standardy Javy.

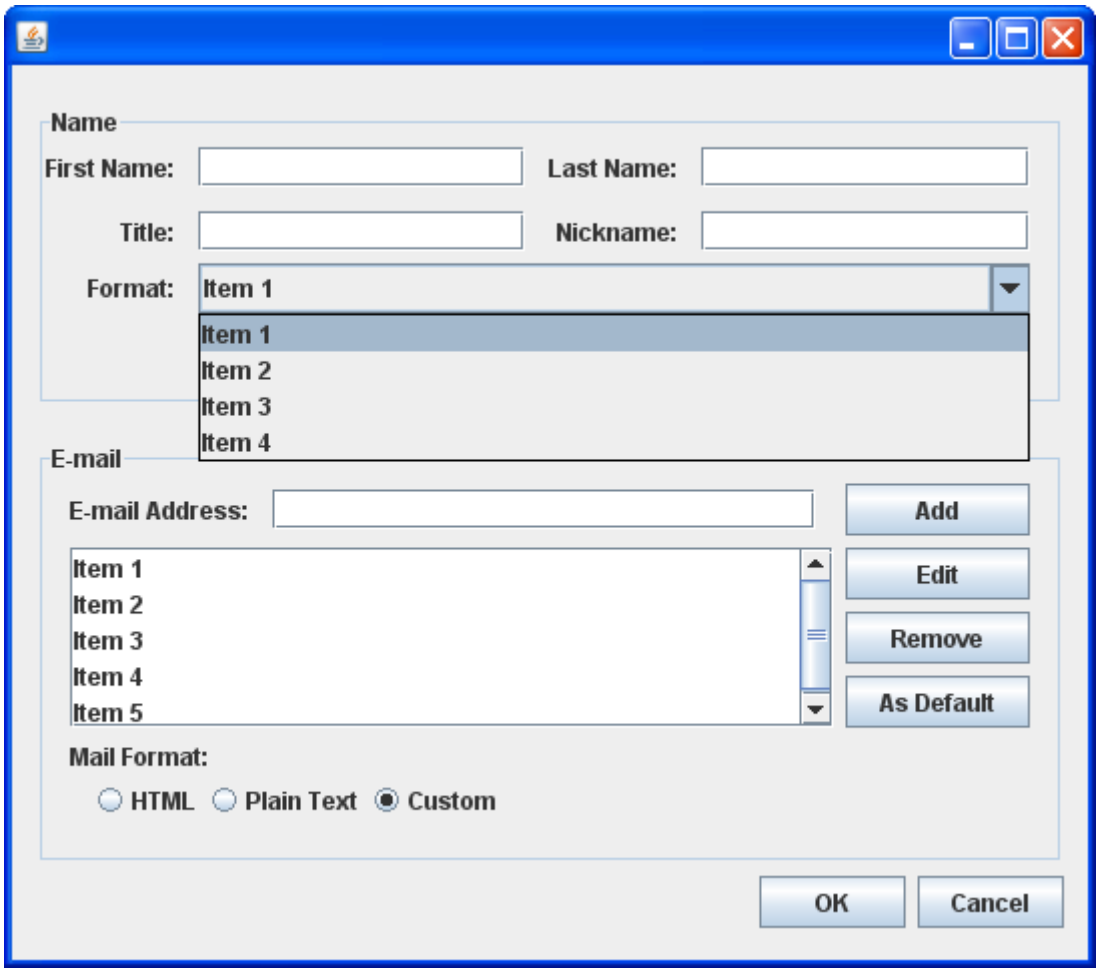
### 5.4.4 INTELLIJ IDEA

Tab. 5.12 Scénář 3 - IntelliJ IDEA

| Kritérium | Popis kritéria | Hodnocení |
|---|---|---|
| 1 | délka kódu | 72/182 řádek |
| 2 | délka tvorby | 16:20 minut |
| | | 6:14 minut |
| 3 | počet dostupných prvků | 27 prvků |
| 4 | porovnání se vzorem | 1 |
| 5 | kvalita kódu | 1 |

Vygenerovaný Java kód je přehledný a stručný. Ruční úprava kódu by nebyla obtížná. Všechny komponenty bylo možné vytvořit a podle potřeby upravit. Výsledné okno zcela odpovídá vzoru, a proto kritérium 4 má známku 1, jak je uvedeno v Tab. 5.12. Opět zde nastal velký rozdíl časů, který v tomto případě vznikl obtížnou prací s předem definovaným umístěním komponent. Scénář 3 si zakládá na správném rozmístění komponent a vytvořit identické rozmístění se ukázalo být velmi náročné, což se promítlo do prvního naměřeného času.

## 5.5 SCÉNÁŘ 4




Obr. 5.5 Scénář 4 [31]

Scénář 4, viz Obr. 5.5, využívá ještě větší počet různých komponent než scénáře před ním. Jedná se také o určitou manipulaci s osobními údaji. Okno obsahuje vnitřní strukturu, kde v každém oddíle se využívají rozdílné komponenty. Vytvořené okno může najít uplatnění v reálné situaci, a proto je nezbytné, aby si s ním návrháře dokázaly poradit.

### 5.5.1 NETBEANS

Tab. 5.13 Scénář 4 - NetBeans

| Kritérium | Popis kritéria | Hodnocení |
|---|---|---|
| 1 | délka kódu | 462 řádek |
| 2 | délka tvorby | 19:37 minut |
| | | 9:51 minut |
| 3 | počet dostupných prvků | 64 prvků |
| 4 | porovnání se vzorem | 1 |
| 5 | kvalita kódu | 2 |

Návrhář NetBeans neměl s vytvořením tohoto scénáře žádný problém. Znovu se ukazuje na naměřených časech, jak znalost daného návrháře urychluje a usnadňuje celkovou práci. Výsledek je identický se vzorem, ale nepřehlednost kódu stále snižuje známku kritériu 5, jak je uvedeno v Tab. 5.13.

### 5.5.2 JDEVELOPER

Tab. 5.14 Scénář 4 - JDeveloper

| Kritérium | Popis kritéria | Hodnocení |
|---|---|---|
| 1 | délka kódu | 495 řádek |
| 2 | délka tvorby | 12:07 minut |
| | | 7:21 minut |
| 3 | počet dostupných prvků | 48 prvků |
| 4 | porovnání se vzorem | 1 |
| 5 | kvalita kódu | 2 |

Ani pro návrhář JDeveloperu nebylo problémem tento scénář vytvořit. Nízké časy, uvedené v Tab. 5.14, dokonce odrážejí snadnou práci s návrhářem při tvorbě tohoto scénáře. Výsledné okno je naprosto stejné jako vzor, podle kterého bylo vytvořeno. I když se zde používají nové komponenty, ukazuje se, že práce s nimi je velmi obdobná jako práce s ostatními komponentami, které už byly použity. Špatná struktura a nepřehlednost generovaného kódu opět sráží známku kritériu 5.

### 5.5.3 ECLIPSE

Tab. 5.15 Scénář 4 - Eclipse

| Kritérium | Popis kritéria | Hodnocení |
|---|---|---|
| 1 | délka kódu | 223 řádek |
| 2 | délka tvorby | 14:09 minut |
| | | 9:30 minut |
| 3 | počet dostupných prvků | 61 prvků |
| 4 | porovnání se vzorem | 1 |
| 5 | kvalita kódu | 1 |

Kompaktní a přehledný kód je ohodnocen známkou 1 u kritéria 5, jak je podrobně uvedeno v Tab. 5.15. Výsledek navíc naprosto odpovídá vzoru, a proto je kritérium 4 ohodnoceno také známkou 1. Návrhář zcela splnil daný scénář.

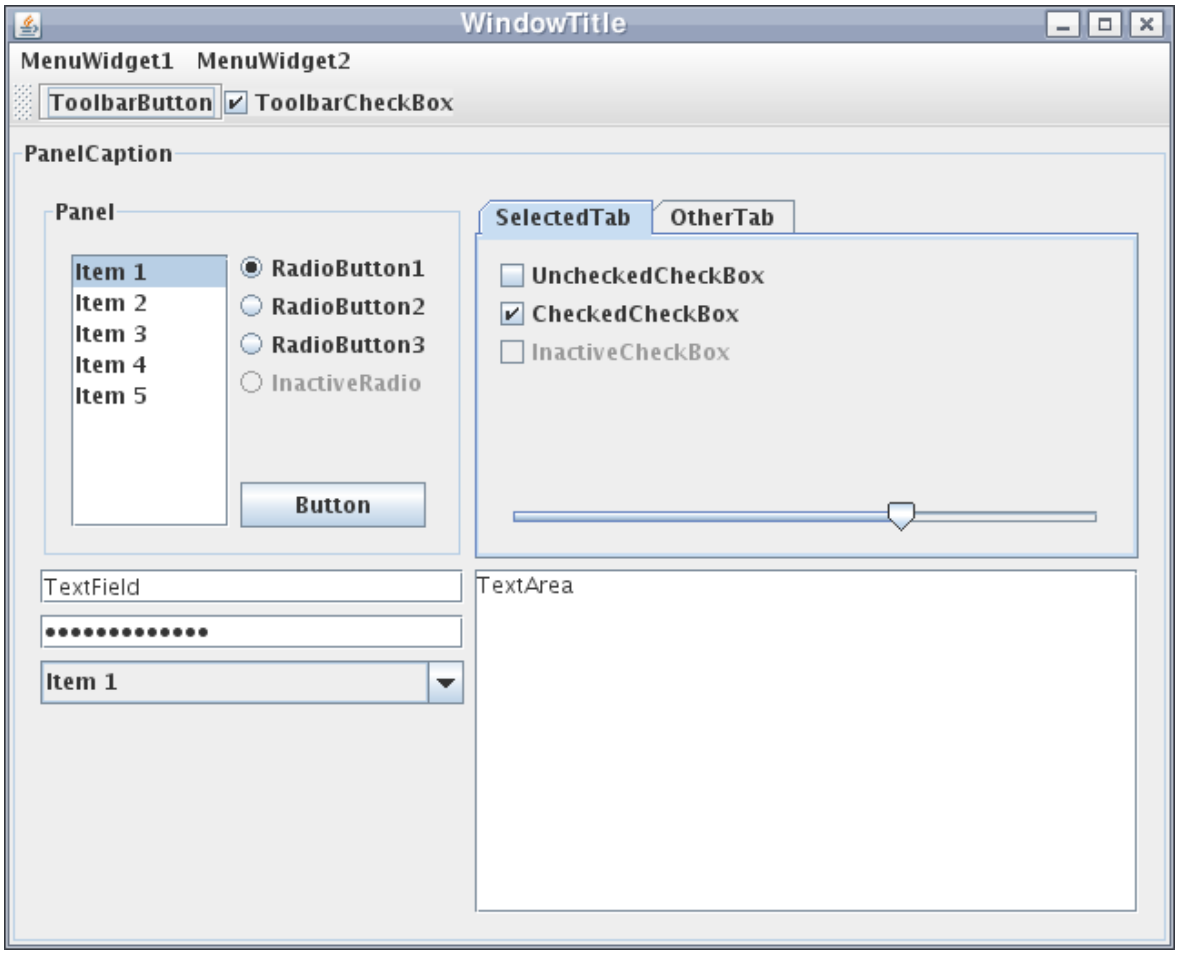
### 5.5.4 IntelliJ IDEA

Tab. 5.16 Scénář 4 - IntelliJ IDEA

| Kritérium | Popis kritéria | Hodnocení |
|---|---|---|
| 1 | délka kódu | 99/248 řádek |
| 2 | délka tvorby | 9:30 minut |
| | | 6:56 minut |
| 3 | počet dostupných prvků | 27 prvků |
| 4 | porovnání se vzorem | 1 |
| 5 | kvalita kódu | 1 |

Integrovaný návrhář IntelliJ IDEA dokázal také zcela vytvořit zadaný scénář. Kritéria 4 a 5 obdrželi známku 1, jak je uvedeno v Tab. 5.16, protože naplnily všechny nutné parametry k jejímu obdržení. Časy jsou znatelně nižší než u ostatních, protože předem definované umístění komponent bylo v tomto případě přínosné a urychlilo celkovou tvorbu okna.

## 5.6 Scénář 5




Obr. 5.6 Scénář 5 [32]

Posledním z testovacích případů je scénář 5, viz Obr. 5.6. Tento scénář obsahuje většinu běžně používaných komponent knihovny Swing, které by návrháře GUI měly být schopny vytvořit. Obsahuje oproti předchozím scénářům zcela nové komponenty a kontejnery. Scénář není vytvořen pro použití v reálné situaci, ale spíše je vytvořen pro vyzkoušení a otestování práce s GUI. Je také velmi užitečný pro otestování práce s vlastnostmi jednotlivých komponent, jelikož obsahuje velké množství specificky upravených či předem nastavených komponent. Velký důraz je kladen na dodržení těchto vlastností a jejich reprodukci v návrhářích.

### 5.6.1 NetBeans

Tab. 5.17 Scénář 5 - NetBeans

| Kritérium | Popis kritéria | Hodnocení |
|---|---|---|
| 1 | délka kódu | 449 řádek |
| 2 | délka tvorby | 18:47 minut |
| | | 9:39 minut |
| 3 | počet dostupných prvků | 64 prvků |
| 4 | porovnání se vzorem | 1 |
| 5 | kvalita kódu | 2 |

Počet vygenerovaných řádek je velmi rozumný a odpovídá složitosti scénáře. Přehlednost kódu, který popisuje rozmístění komponent, však není ani zde optimální, což se promítá do známky 2 u kritéria 5, jak je uvedeno v Tab. 5.17. První čas jasně ukazuje nezkušenost s prací s novými komponentami. Druhý čas již ukazuje potřebný čas k vytvoření daného okna při opakovaném využívání návrháře. Vytvoření takto neprimitivního a celkem rozsáhlého okna do deseti minut je velmi dobrá známka návrháře. Výsledné okno naprosto odpovídá vzoru, což se odráží ve známce u kritéria 4.

### 5.6.2 JDeveloper

Tab. 5.18 Scénář 5 - JDeveloper

| Kritérium | Popis kritéria | Hodnocení |
|---|---|---|
| 1 | délka kódu | 476 řádek |
| 2 | délka tvorby | 11:29 minut |
| | | 7:35 minut |
| 3 | počet dostupných prvků | 48 prvků |
| 4 | porovnání se vzorem | 1 |
| 5 | kvalita kódu | 2 |

Práce i výsledné okno se opět velmi podobá vývojovému prostředí NetBeans. Počet řádek je srovnatelný a dokonce kritéria 4 a 5 mají stejné známky, jak je uvedeno v Tab. 5.18. Výsledek je perfektní a zcela pokrývá vzhled vzoru. Jedinými výtkami jsou již zmíněná nepřehlednost a složitá orientace ve vygenerovaném kódu, které i zde snižují známku kritériu 5.

### 5.6.3 ECLIPSE

Tab. 5.19 Scénář 5 - Eclipse

| Kritérium | Popis kritéria | Hodnocení |
|---|---|---|
| 1 | délka kódu | 230 řádek |
| 2 | délka tvorby | 15:37 minut |
| | | 9:50 minut |
| 3 | počet dostupných prvků | 61 prvků |
| 4 | porovnání se vzorem | 1 |
| 5 | kvalita kódu | 1 |

Návrhář Eclipse si poradil i s posledním scénářem a výsledkem bylo znovu okno identické se vzorem, což mělo za následek známku 1 u kritéria 4, jak je uvedeno v Tab. 5.19. Struktura vygenerovaného kódu se oproti předchozím scénářům trochu zkomplikovala kvůli velkému počtu komponent a změn jejich vlastností. Vygenerovaný kód však zůstal přehledný a umožňuje jakoukoli ruční úpravu kódu.

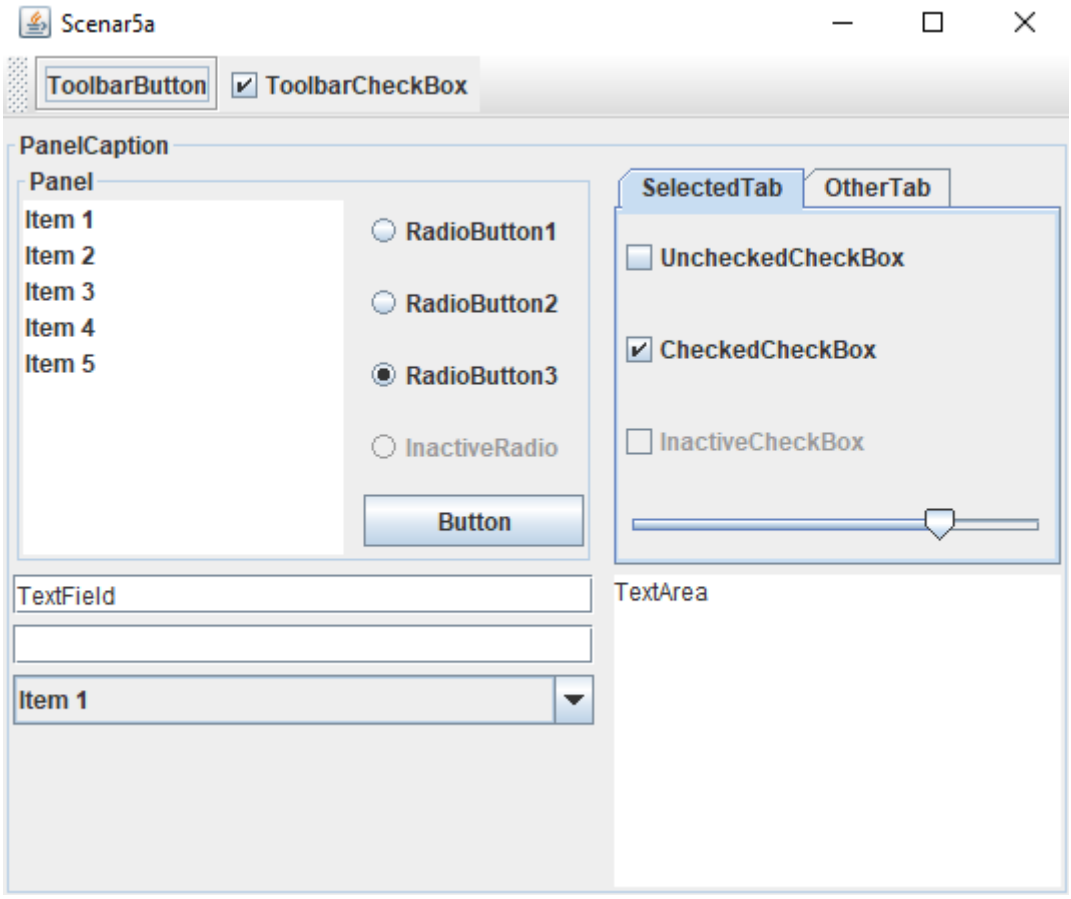
### 5.6.4 INTELLIJ IDEA

Tab. 5.20 Scénář 5 - IntelliJ IDEA

| Kritérium | Popis kritéria | Hodnocení |
|---|---|---|
| 1 | délka kódu | 103/243 řádek |
| 2 | délka tvorby | 9:42 minut |
| | | 6:52 minut |
| 3 | počet dostupných prvků | 27 prvků |
| 4 | porovnání se vzorem | 4 |
| 5 | kvalita kódu | 1 |

Jako kritický problém návrháře IntelliJ IDEA se opět ukázal nedostatek dostupných komponent. Chybějící komponenty `MenuBar`, `Menu` a `MenuItem` se zasloužily o hodnocení kritéria 4 známkou 4, jak je uvedeno v Tab. 5.20. Ostatní komponenty a jejich

vlastnosti byly naprosto v pořádku a vytvořily okno velmi podobné vzoru, viz Obr. 5.7. Kvalita a velikost vygenerovaného kódu byly také na dobré úrovni, což se odrazilo na hodnocení kritéria 5.




Obr. 5.7 Scénář 5 - IntelliJ IDEA

# 6 VYHODNOCENÍ

Vyhodnocení je provedeno na základě výsledků návrhářů u jednotlivých kritérií. Každé kritérium bude nejdříve vyhodnoceno samostatně a návrhářům bude u něj přiděleno pořadí, které bude udávat, jak si u daného kritéria vedly v porovnání s ostatními. Na konci bude poté z jejich pořadí u jednotlivých kritérií vytvořeno výsledné vyhodnocení návrhářů.

## 6.1 KRITÉRIUM 1 – DÉLKA VYGENEROVANÉHO KÓDU

Tab. 6.1 Vyhodnocení kritéria 1

| | **Scénář 1** | **Scénář 2** | **Scénář 3** | **Scénář 4** | **Scénář 5** |
|---|---|---|---|---|---|
| **NetBeans** | 216 | 550 | 355 | 462 | 449 |
| **JDeveloper** | 268 | 567 | 374 | 495 | 476 |
| **Eclipse** | 114 | 268 | 165 | 223 | 230 |
| **IntelliJ IDEA** | 39/196 | 179/208 | 72/182 | 99/248 | 103/243 |

U kritéria 1 jsou návrháře porovnávány podle naměřených hodnot, kde nejnižší hodnota se bere jako nejlepší. Pokud určitý návrhář dokáže u stejného scénáře vygenerovat menší počet řádek, což odpovídá definici kritéria 1, než jiný návrhář, tak to naznačuje, že generuje kratší, efektivnější a neroztažený kód. Kratší kód je dále pak lépe udržovatelný a jeho ruční úpravy nejsou tak náročné. Komentáře ve vygenerovaném kódu mohou naměřenou délku kódu negativně ovlivňovat, a proto je zde zároveň kritérium 5, tedy kvalita kódu, které k délce kódu nepřihlíží a pozitivně hodnotí výskyt užitečných komentářů.

Z výsledků lze vidět, že nejlépe si v tomto ohledu vedl s přehledem návrhář vývojového prostředí Eclipse, jak je uvedeno v Tab. 6.1, ve které všechny hodnoty udávají naměřené počty řádek. Výsledek návrháře IntelliJ IDEA hodnotím jako součet obou hodnot, a proto se u tohoto kritéria umístil na druhém místě. Jako další se umístily návrháře NetBeans a JDeveloper, které si vedly velmi obdobně u všech scénářů. Nejlepší návrhář u kritéria 1, tedy návrhář Eclipse, vygeneroval přibližně poloviční kód než návrhář nejhorší.

## 6.2 KRITÉRIUM 2 – DÉLKA TVORBY

Tab. 6.2 Vyhodnocení kritéria 2

| | Scénář 1 | Scénář 2 | Scénář 3 | Scénář 4 | Scénář 5 |
|---|---|---|---|---|---|
| **NetBeans** | 9:27/4:19 | 16:17/10:36 | 13:53/6:15 | 19:37/9:51 | 18:47/9:39 |
| **JDeveloper** | 8:21/4:48 | 17:59/11:13 | 10:23/6:23 | 12:07/7:21 | 11:29/7:35 |
| **Eclipse** | 8:35/4:30 | 11:17/6:45 | 11:08/6:20 | 14:09/9:30 | 15:37/9:50 |
| **IntelliJ IDEA** | 9:47/4:59 | 11:11/11:53 | 16:20/6:14 | 9:30/6:56 | 9:42/6:52 |

U hodnocení kritéria 2 jsou porovnávány dvě výsledné hodnoty samostatně. Opět se nejlepší návrhář bude vyznačovat nejnižší hodnotou. Větší váha je přidělena druhým hodnotám, jelikož udává časovou náročnost k vytvoření scénáře s předchozí znalostí daného návrháře. Při vyhodnocování je zohledněn můj vliv na výsledek testování, jelikož jsem mohl při práci s různými návrháři vytvořit jiný počet překliků a chyb, které by přímo ovlivnily výslednou časovou náročnost k vytvoření scénáře, a proto časy s rozdílem do 30 sekund hodnotím jako totožné.

Výsledné hodnoty nám ukazují, že scénáře 1 a 3 byly všemi návrháři vytvořeny za stejný časový úsek, jak je uvedeno v Tab. 6.2. Jedinou odchylkou je první hodnota návrháře IntelliJ IDEA u scénáře 3, která je způsobena prací s dříve nepoužitým předem definovaným rozmístěním komponent. Se scénářem 2 si nejlépe poradil návrhář Eclipse, který ho dokázal vytvořit skoro za poloviční čas než ostatní návrháře. Scénáře 4 a 5 byly naopak nejrychleji vytvořeny návrhářem IntelliJ IDEA, což se ukázalo v obou naměřených hodnotách. Celkově si u tohoto kritéria vedl nejlépe návrhář IntelliJ IDEA, který dosáhl nejlepších časů přes většinu scénářů a zaostal pouze u scénáře 2 za návrhářem Eclipse. Jako druhý se umístil návrhář JDeveloper, následován návrhářem Eclipse a jako poslední dopadl návrhář NetBeans.

## 6.3 KRITÉRIUM 3 – POČET DOSTUPNÝCH GRAFICKÝCH PRVKŮ

Tab. 6.3 Vyhodnocení kritéria 3

| | |
|---|---|
| **NetBeans** | 64 prvků |
| **JDeveloper** | 48 prvků |
| **Eclipse** | 61 prvků |
| **IntelliJ IDEA** | 27 prvků |

Kritérium 3, tedy počet dostupných grafických prvků, určuje nejlepšího návrháře jako návrháře s nejvíce dostupnými komponentami. Na první pohled je vidět, že si v tomto ohledu nejlépe vedl návrhář NetBeans, jak je uvedeno v Tab. 6.3, následovaný návrháři Eclipse, JDeveloper a IntelliJ IDEA.

## 6.4 KRITÉRIUM 4 – POROVNÁNÍ SE VZOREM

Tab. 6.4 Vyhodnocení kritéria 4

| | Scénář 1 | Scénář 2 | Scénář 3 | Scénář 4 | Scénář 5 |
|---|---|---|---|---|---|
| **NetBeans** | 1 | 1 | 1 | 1 | 1 |
| **JDeveloper** | 1 | 1 | 1 | 1 | 1 |
| **Eclipse** | 1 | 1 | 1 | 1 | 1 |
| **IntelliJ IDEA** | 2 | 4 | 1 | 1 | 4 |

Kritérium 4 bylo hodnoceno známkami 1, 2, 3, 4 a 5, kde známka 1 je nejlepší a známka 5 nejhorší. Návrháře NetBeans, JDeveloper a Eclipse si všechny vedly skvěle a dokázaly pro všechny scénáře vytvořit okno naprosto identické zadanému vzoru. Pouze návrhář IntelliJ IDEA měl problém se scénáři 2 a 5, jak je uvedeno v Tab. 6.4, kde se ukázal nižší počet dostupných komponent jako kritický problém. Se scénářem 1 si návrhář poradit dokázal, ale s lehkými odchylkami od vzoru.

## 6.5 KRITÉRIUM 5 – KVALITA KÓDU

Tab. 6.5 Vyhodnocení kritéria 5

| | Scénář 1 | Scénář 2 | Scénář 3 | Scénář 4 | Scénář 5 |
|---|---|---|---|---|---|
| **NetBeans** | 2 | 2 | 2 | 2 | 2 |
| **JDeveloper** | 2 | 2 | 2 | 2 | 2 |
| **Eclipse** | 1 | 1 | 1 | 1 | 1 |
| **IntelliJ IDEA** | 1 | 1 | 1 | 1 | 1 |

U kritéria 5 byla možnost obdržet známky 1, 2 a 3, kde známka 1 je brána za nejlepší a známka 3 za nejhorší. Každý návrhář byl přes všechny scénáře ohodnocen stejnou známkou, což naznačuje konzistenci v generování kódu u všech návrhářů. Nejlépe si vedly návrháře Eclipse a IntelliJ IDEA, které obdržely známku 1, jak je uvedeno v Tab. 6.5. U návrháře IntelliJ IDEA je hodnocení zkresleno rozdělením generování kódu do jazyku Java a jazyku XML, nemění to však nic na tom, že kód v jazyku Java byl přehledný, strukturovaný a snadno upravitelný. Návrháře NetBeans a JDeveloper trochu zaostaly se

známkou 2, kterou způsobily stejné chyby, a to složitá orientace ve vygenerovaném kódu, nepřehlednost kódu zaměřeného na rozmístění komponent a z toho plynoucí obtížná ruční úprava kódu.

## 6.6 CELKOVÉ VYHODNOCENÍ

Tab. 6.6 Celkové vyhodnocení

| | Kritérium 1 | Kritérium 2 | Kritérium 3 | Kritérium 4 | Kritérium 5 | Součet |
|---|---|---|---|---|---|---|
| **NetBeans** | 3. | 4. | 1. | 1. | 3. | 12 |
| **JDeveloper** | 4. | 2. | 3. | 1. | 3. | 13 |
| **Eclipse** | 1. | 3. | 2. | 1. | 1. | 8 |
| **IntelliJ IDEA** | 2. | 1. | 4. | 4. | 1. | 12 |

Ve výsledcích, uvedených v Tab. 6.6, je vidět, jak dopadly návrháře v jednotlivých kritériích. Každé číslo udává pořadí, v jakém se návrhář v daném kritériu umístil v porovnání s ostatními návrháři. Po sečtení umístění u všech kritérií je vidět finální umístění jednotlivých návrhářů.

Tímto testováním se ukazuje, že nejlepším návrhářem, tedy návrhářem s nejmenším finálním součtem, je plugin WindowBuilder Pro pro vývojové prostředí Eclipse. Návrhář se umístil na prvním místě u tří z pěti hodnocených kritérií. Jeho hlavní předností se ukázala být kvalita vygenerovaného kódu. Kód, který návrhář vytvořil, je velmi přehledný, lehce upravitelný a neobsahuje žádný nadbytečný kód. Velkým lákadlem pro použití návrháře Eclipse je také velké množství dostupných komponent a nástrojů.

Na druhém místě se podle finálního součtu umístily návrháře NetBeans a IntelliJ IDEA. Pro více nejhorších umístění návrháře IntelliJ IDEA však druhé místo připadá návrháři NetBeans. Na rozhodnutí o druhém místě měl také vliv fakt, že návrhář IntelliJ IDEA negeneruje pouze Java kód jako ostatní návrháře, což výrazně zkresluje kritérium 1 a 5 a může značně snížit celkovou přenositelnost kódu. Návrhář NetBeans se ukázal jako nejlepší v celkovém počtu dostupných komponent. Podstatným nedostatkem se nakonec ukázal být generovaný kód, jelikož nebyl na takové úrovni jako u návrhářů Eclipse či IntelliJ IDEA.

Na třetím místě se tedy umístil návrhář IntelliJ IDEA, který přes pár větších nedostatků generuje velmi strukturovaný a přehledný kód. Největším nedostatkem návrháře se

ukázal být nedostatečný počet dostupných komponent, který velmi ovlivnil hodnocení návrháře u kritérií 3 a 4.

Na posledním čtvrtém místě se umístil návrhář JDeveloper. Návrhář nezaostává za ostatními o moc, ale obsahuje patrné nedostatky oproti ostatním. U žádného z kritérií nebyl výrazně lepší než ostatní, a proto také obsadil poslední místo. Generovaný kód je velmi srovnatelný s kódem, který generuje návrhář NetBeans, ale obsahuje ještě trochu více neužitečných komentářů a nadbytečného roztahování kódu. Dokázal vytvořit všechna okna identická se vzorem, ale ani v jednom ohledu se neukázal jako nejlepší z použitých návrhářů.

Po mém testování dopadly návrháře takto, nejlepším ve většině testovaných vlastností se ukázal být návrhář Eclipse, následovaný návrhářem NetBeans, za kterým se umístil návrhář IntelliJ IDEA a na posledním místě se umístil návrhář JDeveloper. Testované návrháře se neliší mnoha vlastnostmi a jsou schopni si poradit se stejnými problémy. Nejvíce se liší návrhář IntelliJ IDEA, již dříve zmíněným, generovaným kódem ve dvou jazycích a menším počtem dostupných komponent. Jinak jsou si návrháře velmi podobné a při výběru návrháře GUI by se mělo vzít v úvahu i preferované vývojové prostředí, jelikož měnit vývojové prostředí kvůli návrháři GUI se nevyplatí. Všechny testované návrháře GUI jsou obstojné a na velmi dobré úrovni.

# 7 ZÁVĚR

Bakalářská práce obsahuje teoretický úvod do problematiky grafických uživatelských rozhraní. Zaměřuje se především na GUI v programovacím jazyce Java, tedy na knihovny používané dříve jako AWT, ale zároveň i současné jako Swing, SWT či JavaFX. Hlavním zaměřením práce bylo porovnat dostupné integrované návrháře GUI vývojových prostředí pro jazyk Java. Při hledání vývojových prostředí pro jazyk Java byly nalezeny čtyři, které obsahují návrháře GUI. Oproti zadání nebyly podrobně testovány pouze dva, ale všechny nalezené návrháře pomocí pěti testovacích scénářů. Následně podle pěti předem určených kritérií byly návrháře ohodnoceny a jejich výsledky porovnány s ostatními. Nejlepším návrhářem GUI pro jazyk Java se ukázal být plugin WindowBuilder Pro pro vývojové prostředí Eclipse.

Práci by bylo možné rozšířit přidáním většího počtu testovacích scénářů, které by prověřily další vlastnosti a funkcionalitu návrhářů GUI. Dále by bylo možné použít skupinu testerů, kteří by testovali dané scénáře podle stejných pravidel (viz Příloha A). Zprůměrování výsledků by pak pomohlo odstínit případné individuální vlivy jednotlivých testerů a vedlo by k více objektivním a přesnějším výsledkům.

## SEZNAM LITERATURY


[1] AutoIt, „GUI Reference,“ AutoIt, 2019. [Online]. Available: https://www.autoit-script.com/autoit3/docs/guiref/GUIRef.htm. [Přístup získán 20 Duben 2019].

[2] The Centre for Computing History, „The Graphical User Interface,“ The Centre for Computing History, 2019. [Online]. Available: http://www.computinghistory.org.uk-/det/43285/The-Graphical-User-Interface/. [Přístup získán 26 Duben 2019].

[3] Sensomatic, „The Graphical User Interface.,“ Sensomatic, 2019. [Online]. Available: https://www.sensomatic.com/chz/gui/history.html. [Přístup získán 26 Duben 2019].

[4] J. Zukowski, „Abstract Window Toolkit Overview,“ O'Reilly - Safari, 2019. [Online]. Available: https://www.oreilly.com/library/view/java-awt-reference/97815659224-02/04_chapter-01.html. [Přístup získán 19 Prosinec 2018].

[5] Techopedia, „Abstract Window Toolkit (AWT),“ Techopedia, 2019. [Online]. Available: https://www.techopedia.com/definition/3735/abstract-window-toolkit-awt. [Přístup získán 29 Listopad 2018].

[6] V. Piroumian, „The AWT Components,“ informIT, 21 Srpen 2001. [Online]. Available: http://www.informit.com/articles/article.aspx?p=131113. [Přístup získán 26 Březen 2019].

[7] P. Herout, JAVA - grafické uživatelské prostředí a čeština, České Budějovice: KOPP, 2012.

[8] Oracle, „Exploring the AWT Layout Managers,“ Oracle, November 2001. [Online]. Available: https://www.oracle.com/technetwork/articles/javase/awtlayoutmgr-137229.html. [Přístup získán 26 Duben 2019].

[9] Leepoint.net, „Swing vs AWT vs SWT vs ...,“ Leepoint.net, 2019. [Online]. Available: https://www.leepoint.net/JavaBasics/gui/gui-commentary/guicom-25-swingawt-swtxul.html. [Přístup získán 26 Duben 2019].

[10] E. Clayberg a D. Rubel, Eclipse: Building Commercial-Quality Plug-ins, Second Edition, Addison Wesley Professional, 2006.

[11] The Eclipse Foundation, „SWT Widgets,“ Eclipse.org, 2019. [Online]. Available: https://www.eclipse.org/swt/widgets/. [Přístup získán 6 Březen 2019].

[12] developer.com, „Constructing SWT Layouts,“ developer.com, 15 Duben 2004. [Online]. Available: https://www.developer.com/design/article.php/3340621/-Constructing-SWT-Layouts.htm. [Přístup získán 26 Duben 2019].

[13] JavaTPoint, „Java Swing Tutorial,“ JavaTPoint, 2019. [Online]. Available: https://www.javatpoint.com/java-swing. [Přístup získán 10 Prosinec 2018].

[14] Massachusetts Institute of Technology, „A Visual Guide to Swing Components,“ Massachusetts Institute of Technology, 2019. [Online]. Available: http://web.mit.edu/6.005/www/sp14/psets/ps4/java-6-tutorial/components.html. [Přístup získán 29 Březen 2019].

[15] Oracle, „A Visual Guide to Layout Managers,“ Oracle, 2019. [Online]. Available: https://docs.oracle.com/javase/tutorial/uiswing/layout/visual.html. [Přístup získán 26 Duben 2019].

[16] Oracle, „How to Use GroupLayout,“ Oracle, 2019. [Online]. Available: https://docs.oracle.com/javase/tutorial/uiswing/layout/group.html. [Přístup získán 26 Duben 2019].

[17] Oracle, „How to Use SpringLayout,“ Oracle, 2019. [Online]. Available: https://docs.oracle.com/javase/tutorial/uiswing/layout/spring.html. [Přístup získán 26 Duben 2019].

[18] JavaTPoint, „JavaFX Overview,“ JavaTPoint, 2019. [Online]. Available: https://www.javatpoint.com/javafx-overview. [Přístup získán 28 Prosinec 2018].

[19] EDUCBA, „JavaFX vs Swing,“ EDUCBA, 2019. [Online]. Available: https://www.educba.com/javafx-vs-swing/. [Přístup získán 26 Duben 2019].

[20] NetBeans.org, „A Brief History of NetBeans,“ NetBeans.org, 2019. [Online]. Available: https://netbeans.org/about/history.html. [Přístup získán 3 Březen 2019].

[21] NetBeans.org, „Swing GUI Builder (formerly Project Matisse),“ NetBeans.org, 2019. [Online]. Available: https://netbeans.org/features/java/swing.html. [Přístup získán 10 Duben 2019].

[22] N. Veys, „FAQ Where did Eclipse come from?,“ Eclipse Foundation, 7 Červen 2006. [Online]. Available: https://wiki.eclipse.org/FAQ_Where_did_Eclipse_come_from-%3F. [Přístup získán 13 Duben 2019].

[23] Eclipse.org, „WindowBuilder,“ Eclipse.org, 2019. [Online]. Available: https://www.eclipse.org/windowbuilder/. [Přístup získán 20 Březen 2019].

[24] JavaTPoint, „IntelliJ IDEA Tutorial,“ JavaTPoint, 2019. [Online]. Available: https://www.javatpoint.com/intellij-idea-tutorial. [Přístup získán 12 Duben 2019].

[25] JetBrains, „GUI Designer Basics,“ JetBrains, 17 Duben 2019. [Online]. Available: https://www.jetbrains.com/help/idea/gui-designer-basics.html. [Přístup získán 18 Duben 2019].

[26] Oracle, „Oracle JDeveloper,“ Oracle, 2019. [Online]. Available: https://www.oracle.com/technetwork/developer-tools/jdev/overview/index.html. [Přístup získán 26 Březen 2019].

[27] Oracle, „About JDeveloper's User Interface Design Tools,“ Oracle, 2019. [Online]. Available: https://download.oracle.com/otn_hosted_doc/jdeveloper/j2ee101302/-developing_gui_clients_j2ee/ui_adesigntoolreqs.html. [Přístup získán 3 Březen 2019].

[28] ThaiCreate.com, „How to use : Java GUI Member Register Form and Validation Data,“ ThaiCreate.com, 2019. [Online]. Available: https://www.thaicreate.com-/java/java-gui-example-register-form-validation-data.html. [Přístup získán 18 Prosinec 2018].

[29] J. Paul, „How can I create a simple calculator with GUI in Java?,“ Quora, 9 Červenec

2016. [Online]. Available: https://www.quora.com/How-can-I-create-a-simple-calculator-with-GUI-in-Java. [Přístup získán 18 Prosinec 2018].

[30] Kode Blog Tutorials, „Insider's Guide: Java Swing JDBC CRUD Example with Jasper Reports,“ Kode Blog Tutorials, 2019. [Online]. Available: http://www.kode-blog.com/java-swing-jdbc-crud-example-with-jasper-reports. [Přístup získán 18 Prosinec 2018].

[31] JavaGUICodeExample.com, „NetBeans and Java Desktop GUI:,“ JavaGUICodeExample.com, 2019. [Online]. Available: https://www.javagui-codexample.com/javadesktopguinetbeans4.html. [Přístup získán 18 Prosinec 2018].

[32] Wikimedia Commons, „File:Gui-widgets.png,“ Wikimedia Commons, 5 Duben 2007. [Online]. Available: https://commons.wikimedia.org/wiki/File:Gui-widgets.png. [Přístup získán 18 Prosinec 2018].

## SEZNAM OBRÁZKŮ

## SEZNAM TABULEK

# PŘÍLOHY

# PŘÍLOHA A - POSTUP K OTESTOVÁNÍ NÁVRHÁŘŮ GUI

**Předpoklady:**

- Nainstalovaná potřebná vývojová prostředí
- Zvolené a připravené testovací scénáře
- Připravený měřič času (optimálně: https://www.online-stopwatch.com/)

**Postup:**

1. **krok**

   Otevřete si první vývojové prostředí, které budete testovat (na pořadí nezáleží).

2. **Krok**

   Vyzkoušejte si jak vytvořit `JFrame` ve Vámi zvoleném vývojovém prostředí (neboli okno pro návrh GUI).

3. **Krok**

   Otevřete si první testovací scénář, který bude vaším vzorem.

4. **Krok - nejdůležitější, čtěte pozorně!**

   Zapněte měření času a vytvořte GUI v návrháři, co nejvíce podobné vzoru. Měření času zapněte před vytvořením nového `JFrame` a vypněte po zkompilování Vámi vytvořeného GUI. Při vložení komponenty `Button`, `ComboBox`, `CheckBox`, `RadioButton` či `MenuItem` nezapomeňte vytvořit i posluchače těchto komponent (stačí `ActionListener`). Všechny vlastnosti objektů se snažte zcela napodobit.

   Příklady: změna velikosti/tučnosti písma, ne/schopnost zapisovat do textových polí, označené objekty jsou označené hned po zapnutí GUI, počet itemů v rozbalovacím poli je shodný a se shodnými názvy, komponenty `RadioButton` jsou v jedné skupině, textové pole na heslo není textové pole vyplněné hvězdičkami

5. **Krok**

   Zapište si čas a okamžitě zopakujte krok č. 4. Výsledkem budou dvě Vámi vytvořená stejná GUI podle jednoho vzoru a dva Vámi naměřené časy.

6. **Krok**

   Opakujte krok č. 3 až krok č. 5 s dalšími Vámi zvolenými testovacími scénáři jako novými vzory.

7. **Krok**

   Opakujte všechny kroky pro ostatní testovaná vývojová prostředí.