Kapitola 1

Programátorská příručka

1.1 Úvod

1.1.1 Technologie

- Program je psaný v jazyce **Java 1.7**.
- GUI je vytvářeno pomocí knihovny **SWT** . (`http://eclipse.org/swt/`)
- Pro layout GUI byl použit **MigLayout** (`http://www.miglayout.com/`). Celé tělo pluginu je ale vykreslováno na SWT widget Composite, kterému lze nastavit libovolný vnitřní layout manager. Není tedy nutné omezovat se na MigLayout (nicméně ho vřele doporučuji).
- **Jsoup** (`http://jsoup.org/`) je skvělý HTML parser, který byl použit u pluginu SA Notify a je tedy již součásti aplikace. Jeho využití taktéž doporučuji, umožňuje velmi jednoduché extrahování informací z HTML stránky, ale i posílání HTML requestů včetně cookies.

1.1.2 Knihovny

Zde je seznam knihoven, které program používá (a které je tedy nutné přidat do projektu, aby ho bylo možné zkompilovat). Knihovny jsou přibaleny ke zdrojovým kódům.

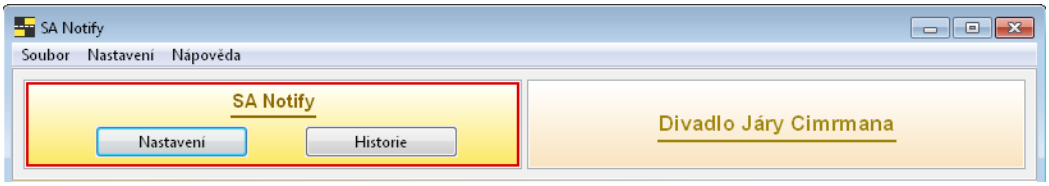
- SWT – je potřeba správná verze podle platformy! Viz `eclipse.org/swt/`.
- MigLayout
- JShortcut_my – upravená verze JShortcut (`https://github.com/jimmc/jshortcut`), která načte správnou verzi jshortcut.dll z adresáře lib/.
- Jsoup
- Apache.commons.lang.StringEscapeUtils
- Apache.commons.codec.binary

A dále tyto knihovny z IDE Eclipse (v případě importu projektu do Eclipse je tedy není nutné manuálně přidávat):

- org.eclipse.ui.forms
- org.eclipse.core.databinding
- org.eclipse.core.databinding.beans
- org.eclipse.core.databinding.observable
- org.eclipse.core.databinding.property
- org.eclipse.core.equinox.common
- org.eclipse.core.runtime
- org.eclipse.core.runtime.compatibility
- org.eclipse.jface.databinding
- org.eclipse.jface.util – pouze třída Geometry.class,

1.1.3 Pojmy

- **plugin menu** – záložka pluginu na horní liště, na které lze přepínat mezi pluginy.
- **plugin active-menu** – menu aktivní pluginu, může obsahovat tlačítka (viz obr. 1.1). Tvoří ho programátor pluginu.
- **plugin idle-menu** – menu neaktivního pluginu, je vytvořeno automaticky.
- **tělo pluginu** – samotný plugin, tedy to, co se zobrazí v hlavním okně při kliknutí na plugin menu.
- **pluginBody** – SWT Composite pluginBody je fieldem třídy Plugin.java. Na tomto Compositu je celé tělo pluginu.



Obrázek 1.1: Vlevo active-menu pluginu SA Notify, vpravo idle-menu pluginu Divadlo J.C.

1.2 Struktura aplikace

Zdrojové kódy obsahují 3 základní *Java Package*:

- **sanotify** – package obsahující kódy rodičovské aplikace
- **plugins** – package obsahující (abstraktní) třídy a interfacy související s pluginy
- **sandbox** – slouží pouze k testování, není součástí distribučního balíčku

Na obrázku 1.2 je struktura tříd typického pluginu.

Každý plugin musí obsahovat tyto třídy (respektive implementovat jejich nadtřídy):

- **NewPlugin** – třída reprezentující daný plugin, implementuje metody interfacu **PluginInterface**. Klíčová je její metoda **setPluginBody()**, která vytvoří celé tělo pluginu. *Parent* všech komponentů těla pluginu je Composite **pluginBody**,
- **NewPluginMenu** – obsahuje implementaci active-menu pluginu. Celé menu je na Composite **menuBG**, který je fieldem abstraktní nadtřídy PluginMenu.java.
- **NewPluginTableItem** – reprezentuje jednu hledanou položku (např. jeden autobus Student Agency). Implementace této třídy (podtřídy PluginTableItem) povinná není, umožňuje ale využití metody *safe()* pro ukládání probíhajících hledání, implementované ve třídě Plugin, a také metod třídy PluginHistory pro práci s historií hledání.

Dále plugin navržený na obrázku 1.2 obsahuje třídy:

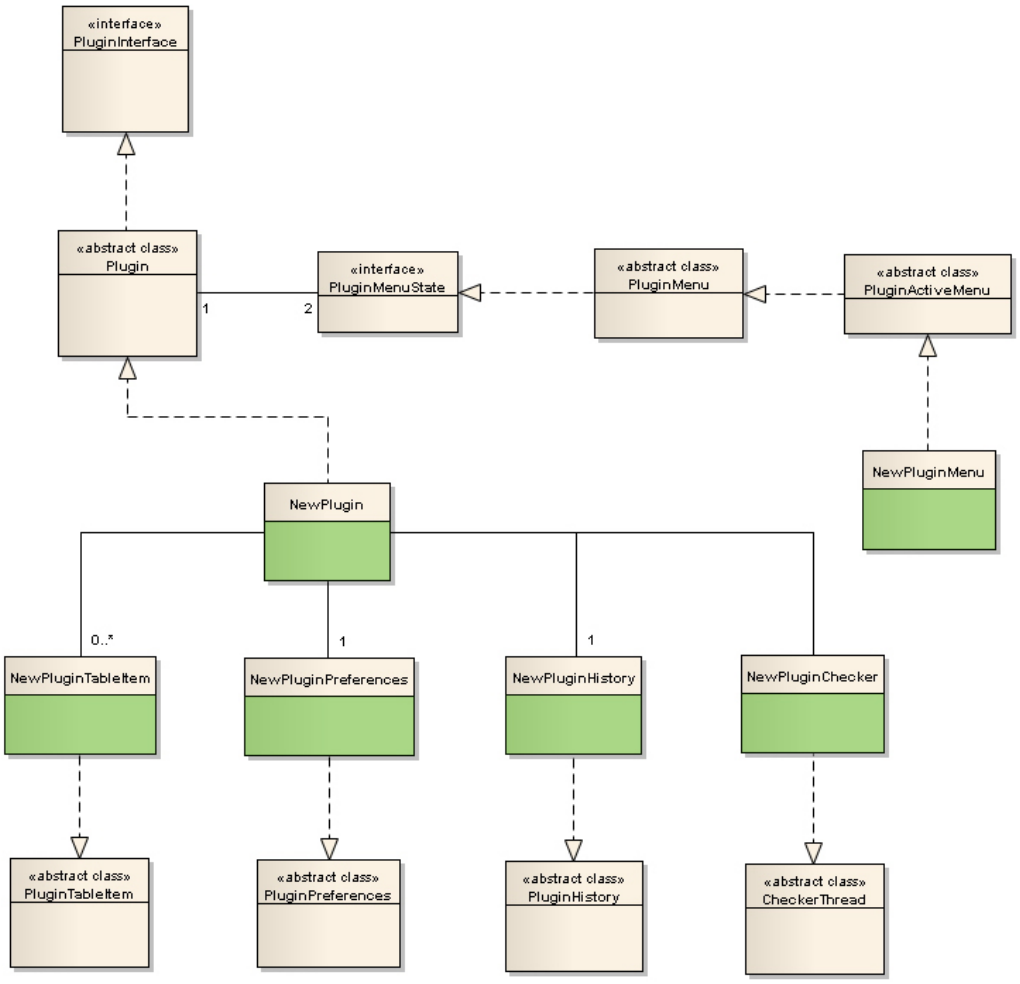
- **NewPluginPreferences** – otevře dialog Nastavení pluginu.
- **NewPluginHistory** – zobrazí historii hledání.
- **NewPluginChecker** – samostatné vlákno, které provádí cyklické hledání volných míst.

Abstraktní třídy PluginPreferences.java, PluginHistory.java a CheckerThread.java obsahují podpůrné metody, které se mohou hodit při vytváření dialogu Nastavení (PluginPreferences) dialogu Historie (PluginHistory) nebo vlákna, které se cyklicky připojuje na server a kontroluje volná místa (CheckerThread).

Tyto třídy tedy není nutné implementovat, ale mohou ušetřit hodně práce.

PluginLoader.java: V této třídě probíhá načítání pluginů. V současnosti je nutné každý nový plugin ručně přidat v metodě **loadPlugins()**:

```
plugins.add(new NewPlugin(display, shell, properties, messenger));
```



Obrázek 1.2: Třídní diagram pluginu

1.3 Jak začít

Všechny třídy nového pluginu by měly být v balíku **plugins.NewPlugin**. Tedy např. ve složce src/plugins/NewPlugin/.

Soubory pluginu (tzn. obrázky, konfigurační soubory apod.) zase ve složce /plugins/NewPlugin/.

Ve zdrojových kódech je již k dispozici adresář src/plugins/NewPlugin/, který obsahuje základní strukturu nového pluginu, přesně takovou, jako je na obr. 1.2.

Stačí tedy doplnit těla metod – především metody **NewPlugin.setPluginBody()**. Všechny metody, které je třeba implementovat, mají javadoc dokumentaci, popisující, co mají dělat. Stačí tedy nahlédnout do dokumentace nebo do kódu příslušné nadtřídy.

Není nutné použít všechny připravené třídy (viz výše – povinné jsou jen třídy NewPlugin a NewPluginMenu).

1.4 Univerzální třídy

Package **sanotify** obsahuje některé univerzální třídy, které je vhodné použít:

ImageHandler.java – statická třída, určená k načítání všech obrázků. Všechny obrázky (SWT Image) je nutné ručně disposovat při ukončení aplikace (image.dispose()), hodí se tedy mít je všechny na jednom místě.

Pro nový plugin doporučuji použít podtřídu NewPluginImageHandler.

CommentedProperties.java – podtřída java.util.Properties, která nevymaže komentáře z properties souboru při přidávání nových položek.

Messenger.java – zajišťuje zobrazování zpráv uživateli a to jak chybových, tak informačních. Pro zobrazení popup zprávy uživateli stačí zavolat např.:

```
new Messenger().message_warning("Hello world");
```

Buttons.java – zjednodušuje vytváření tlačítek. Má několik metod pro vytvoření často používaných tlačítek.

Sounds.java – přehrává zvuky – metoda **public Clip playSound(String fileName)**.

Commons.java – obsahuje zejména metody pro snadné získání hodnot uložených v CommentedProperties. Např. metoda **boolean getBoolProperty()** se pokusí nalézt zadaný klíč v CommentedProperties a převést jeho hodnotu na boolean.

ImageCodeGen.java – jednoduchá třída pro vygenerování kódu pro nový obrázek. Vygeneruje kód pro příslušné fieldy, getter a setter, který stačí zkopírovat do NewPluginImageHandler.java

1.5 Logování chyb

Pro logování chyb je určena metoda MainClass.logException(), která uloží výjimku do .log souboru ve složce /bin/log/.

Doporučený postup odchytávání chyb:

```
try { . . . }
catch (Exception e ) {
    String msg="Zpráva co se uloží s chybou.";
    MainClass.logException(e, NewPlugin.class.getName(), msg);
 }
```

1.6 Jazykové verze

Různé jazykové verze pluginu jsou uloženy v souborech **Bundle.properties**.
V balíku NewPlugin jsou připraveny 2 soubory:

- **Bundle.properties** obsahuje českou jazykovou verzi
- **Bundle_en.properties** anglickou verzi

Všechny Stringy, jejichž hodnota se má měnit podle jazykové verze, stačí nahradit touto konstrukcí:

```
ResourceBundle.getBundle("plugins/NewPlugin/Bundle").getString("Klic1");
```

A následně klíč a hodnotu doplnit do souborů – např. do Bundle.properties doplnit řádek

```
Klic1=český text
```

A do Bundle_en.properties doplnit

```
Klic1=English text
```

Programovací IDE má většinou nástroje pro rychlé nahrazení všech Stringů a přidání klíčů do Bundle souborů.
V NetBeans je to Tools → Internationalization → Internationalization Wizard.