UNIVERZITA PARDUBICE
Fakulta elektrotechniky a informatiky

Multiplatformní správce souborů v jazyce Java

Pavel Konečný

Bakalářská práce
2011

Univerzita Pardubice
Fakulta elektrotechniky a informatiky
Akademický rok: **2010/2011**

ZADÁNÍ BAKALÁŘSKÉ PRÁCE

(PROJEKTU, UMĚLECKÉHO DÍLA, UMĚLECKÉHO VÝKONU)

Jméno a příjmení:	**Pavel KONEČNÝ**
Osobní číslo:	**I07676**
Studijní program:	**B2646 Informační technologie**
Studijní obor:	**Informační technologie**
Název tématu:	**Multiplatformní správce souborů v jazyce Java**
Zadávající katedra:	**Katedra informačních technologií**

Zásady pro vypracování:

Cílem práce je vytvořit aplikaci správce souborů, která bude fungovat jak v operačních systémech Unix, tak v operačních systémech Windows. To umožní multiplatformní jazyk Java. V programu bude využit klasický dvoupanelový interface. Program bude pracovat s protokolem FTP a bude zahrnovat základní funkce souborových managerů jako kopírování, přesouvání, vyhledávání, atd. Disponovat by měl také rozšiřujícími funkcemi jako hromadné přejmenování souborů, volba vzhledu programu a možností konfigurace nástrojové lišty.

Rozsah grafických prací:

Rozsah pracovní zprávy:

Forma zpracování bakalářské práce: **tištěná/elektronická**

Seznam odborné literatury:

[1] **HEROUT, Pavel. Java : bohatství knihoven. 3. vyd. České Budějovice : Kopp, 2008. 251 s. ISBN 978-80-7232-368-5.**

[2] **HEROUT, Pavel. Java : grafické uživatelské prostředí a čeština. Dotisk 2. vyd. České Budějovice : Kopp, 2009. 316 s. ISBN 978-80-7232-328-9.**

[3] **Eckel, B. Myslíme v jazyku Java, Grada Publishing, Praha, 2001**

[4] **Oracle. The Java Tutorials [online]. Cit. 12.5.2010. Dostupné z WWW: http://download.oracle.com/javase/tutorial/**

Vedoucí bakalářské práce: **doc. Ing. Milan Javůrek, CSc.**
Katedra řízení procesů

Datum zadání bakalářské práce: **17. prosince 2010**

Termín odevzdání bakalářské práce: **13. května 2011**

Univerzita Pardubice Česká republika

L.S.

prof. Ing. Simeon Karamazov, Dr.
děkan

Ing. Lukáš Čegan, Ph.D.
vedoucí katedry

V Pardubicích dne 31. března 2011

Prohlášení autora

Prohlašuji, že jsem tuto práci vypracoval samostatně. Veškeré literární prameny a informace, které jsem v práci využil, jsou uvedeny v seznamu použité literatury.

Byl jsem seznámen s tím, že se na moji práci vztahují práva a povinnosti vyplývající ze zákona č. 121/2000 Sb., autorský zákon, zejména se skutečností, že Univerzita Pardubice má právo na uzavření licenční smlouvy o užití této práce jako školního díla podle § 60 odst. 1 autorského zákona, a s tím, že pokud dojde k užití této práce mnou nebo bude poskytnuta licence o užití jinému subjektu, je Univerzita Pardubice oprávněna ode mne požadovat přiměřený příspěvek na úhradu nákladů, které na vytvoření díla vynaložila, a to podle okolností až do jejich skutečné výše.

Souhlasím s prezenčním zpřístupněním své práce v Univerzitní knihovně.

V Pardubicích dne 10. 04. 2011 Pavel Konečný

Poděkování

Na tomto místě bych rád poděkoval hlavně svému vedoucímu práce, doc. Ing. Milanu Javůrkovi, CSc., za vstřícnost, ochotu a cenné rady při tvorbě práce. Dále svým rodičům za podporu při studiu a v neposlední řadě také svým přátelům a přítelkyni za testování aplikace.

Anotace

Tato práce se zabývá vytvořením aplikace správce souborů, kterou je možno obsluhovat z více operačních systémů. To je umožněno multiplatformním jazykem Java.

Klíčová slova

Java, správce souborů, multiplatformní, sériová komunikace, RS-232, FTP

Title

Multi-platform file manager in Java language.

Annotation

This work deals with file manager creating. This one is possible to be used in many operating systems. Java multiplatform language make it possible.

Keywords

Java, file manager, multiplatform, serial communication, RS-232, FTP

Obsah

Seznam zkratek

OS	Operační systém
PC	Personal computer
JRE	Java Runtime Environment
JDK	Java Development Kit
JVM	Java Virtual Machina
API	Application Programming Interface
Java SE	Java Standard Edition
Java EE	Java Enterprise Edition
Java ME	Java Micro Edition
PDA	Personal Digital Assistant
FTP	File Transfer Protocol
RS-232	Recommended Standard 232
LPT	Line Print Terminal
COM port	Communications Port
UML	Unified Modeling Language
MS	Microsoft
PX	Pixel

Seznam obrázků

Seznam tabulek

1 Úvod

Primárním cílem této práce je vytvoření aplikace správce souborů. Aplikace slouží k jednoduché správě souborů a adresářů. Spuštění je možno na různých operačních systémech, neboť aplikace není závislá na architektuře. Kromě základních funkcí pro správu souborů podporuje také hromadné přejmenování, tisk a vyhledávání souborů. Veškeré nastavení je ukládáno do binárního souboru. Jako rozšiřující modul implementuje komunikaci po sériové lince.

První část charakterizuje jazyk zvolený pro tuto práci – jazyk Java. Popsány jsou stručně výhody i nevýhody a také prostředky týkající se Javy.

Další část práce seznámí čtenáře s uživatelským popisem aplikace a poskytne jednoduchý manuál včetně doplňujících grafických náhledů s vysvětlením všech funkcí.

Dále je v práci obsažena část implementační. Nejprve je zmíněna zvolená technologie a poté stručný popis tříd aplikace formou jednotlivých balíčků. Zde se také nachází základní seznámení se sériovou komunikací.

Práce pokračuje UML technologií a znázorněním diagramů jednotlivých tříd. Diagramy jsou rovněž rozděleny na jednotlivé balíčky.

Závěrečná část nastíní problémy při vývoji aplikace a shrne průběh práce.

2 Java

Java je programovací jazyk firmy Sun Microsystems, uvedený na trh roku 1995. Později byla koupená firmou Oracle [1]. V roce 2007 firma Sun uvolnila zdrojové kódy a Java se potom stala *open source*[1] [1]. Je to objektově orientovaný jazyk, který vychází z jazyka C++. Také svojí syntaxí je velmi podobná svému předchůdci. V dnešní době je Java asi nejrozšířenějším jazykem z důvodu své nezávislosti na architektuře.

2.1 Jazyk Java

Zde je příklad výhod/nevýhod, porovnání jazyka Java a jeho předchůdce C++.

Výhody

Jako jedny z hlavních výhod Javy jsou:

- Přidělování a uvolňování paměti obstarává automaticky. Programátor nemusí volat destruktor objektu pomocí příkazů *delete*, nebo *free()* [2]. Uvolnění paměti zajistí tzv. *Garbage collector*. Programátor může „doporučit“ objekt k uvolnění z paměti přiřazením neplatné reference *null*.

- Narozdíl od jazyka C++ nemá *ukazatele*. Jsou zde nahrazeny *referencemi*. Proto nemůže vzniknout známá chyba, kdy programátor nastaví ukazatel mimo strukturu. Dereference je prováděna *runtime*.

- Používá takzvanou typovou kontrolu. To znamená, že všechny proměnné musí mít svůj datový typ.

- Nezávislá na architektuře. Toto je jedna z nejsilnějších stránek Javy. Nezávislost umožňuje běh aplikace na libovolném operačním systému nebo architektuře.

Nevýhody

Java má i své záporné vlastnosti:

- Pomalý start aplikace. Důvod je zapříčiněn tím, že prostředí musí nejprve přeložit program a až potom jej spustit [1].

- Paměťová náročnost aplikace je větší, protože je nutností mít v paměti celé běhové prostředí [1].

Java nepodporuje narozdíl od již zmiňovaného jazyka C++ vícenásobnou dědičnost. Pracuje s rozhraními, které mají za úkol tuto vlastnost nahradit. Zdali je to výhoda, nebo nevýhoda, je spíš otázka individuální.

[1] počítačový software s otevřeným zdrojovým kódem

2.2 Platformy Javy

Zde bude uvedeno, že existují i jiné edice Javy, než jen pro desktopové počítače. Tyto edice se jako celek nazývají *platforma Java*. Java platforma obsahuje dvě komponenty:

- Java Virtual Machine (JVM) – viz kapitola 2.3.
- Application Programming Interface (Java API) – sada standardních knihoven dodávaná s JVM [1].

Mezi nejrozšířenější platformy Javy patří:

- Java Standard Edition (Java SE) - jedná se o klasickou verzi Javy, která se již stává standardem operačních systémů Linux. Tato verze umožňuje běh aplikací na PC.
- Java Enterprise edition (Java EE) – určená pro vývoj a provoz podnikových aplikací a informačních systémů. Je to nadstavba nad základem Java SE [3].
- Java Aplety (Java Applets) – jsou (mini) aplikace umístěné do webových stránek a spuštěné ve webovém prohlížeči. Je to v podstatě klasická aplikace, nemá ovšem přístup k lokálním souborům počítače. Přistupovat k souborům na serveru[2] může.
- Java Micro Edition (Java ME) – programy pro zařízení jako jsou mobilní telefony, pagery[3], PDA[4], atd. Tyto zařízení mají omezené možnosti (napájení, paměť, zobrazovací jednotku, …) a proto nemusí zvládnout standardní edici Javy.

2.3 Java Virtual Machine (JVM)

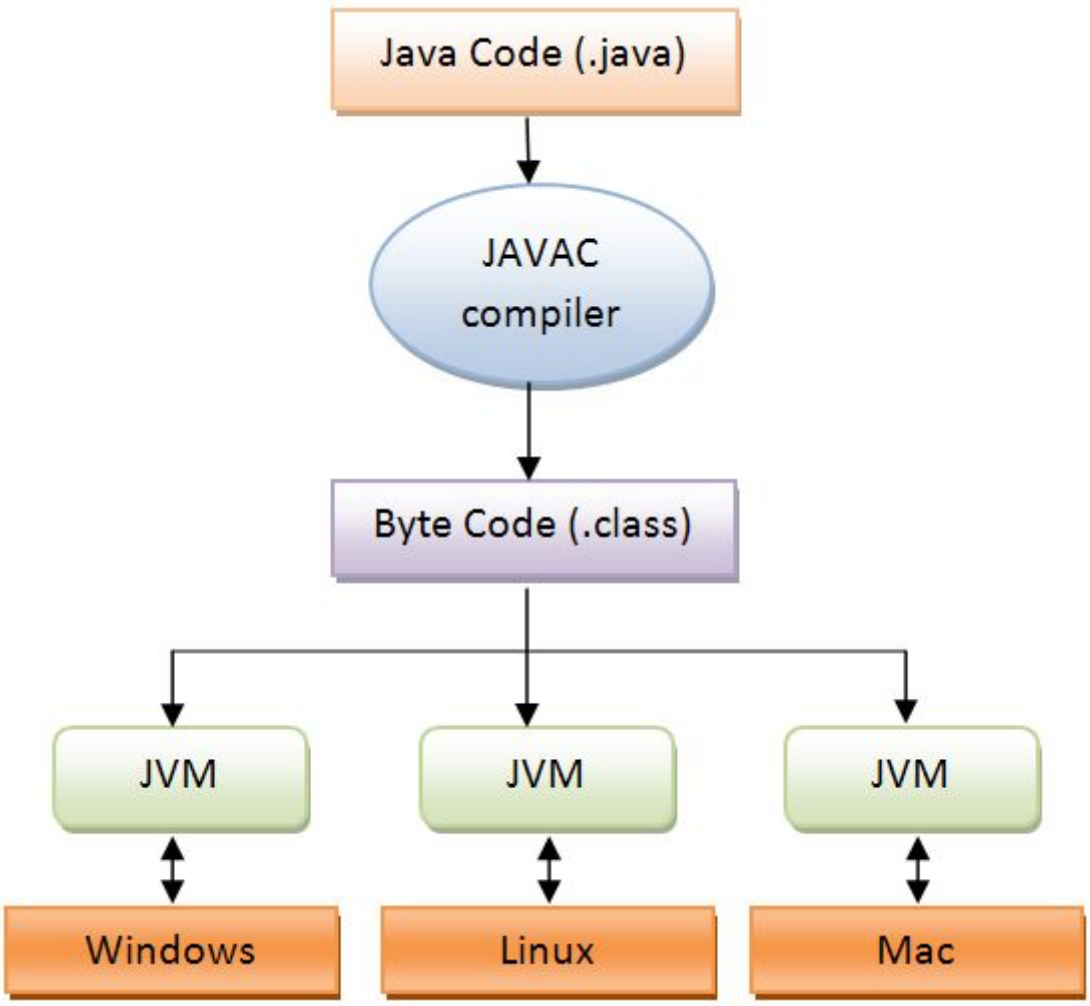
JVM je prostředí, ve kterém jsou spouštěny aplikace jazyka Java. Obsahuje sadu programů a datových struktur. Zajišťuje hospodaření s pamětí, chování procesoru, spravuje vstupy a výstupy a nezapomíná ani na bezpečnost aplikace. Jedná se vlastně o celý počítač, který ale nemá svůj reálný vzor. Proto je JVM interpretován jako virtuální počítač.

Protože program JVM je vyvinut na mnoha platformách (Microsoft Windows, Linux, Mac OS, Solaris OS), je umožněno Java aplikaci vytvořit pouze jednou a dále ji spouštět na všech těchto platformách. Takovýto program je také zabudován do většiny internetových prohlížečů, které dokáží potom spustit výše uvedené *Java Aplety*.

[2] Počítač, poskytující určité služby nebo prostředky k využití takovýchto služeb
[3] Přenosné zařízení, umožňující přijímat zprávy
[4] Kapesní počítač s dotykovou obrazovkou

Vznik Java aplikace probíhá následovně:

- Programátor napíše zdrojový kód do souboru s příponou **.java*.
- Tento kód je přeložen jako takzvaný mezikód (Java Byte Code) do souborů s příponou **.class*.
- Vytvořený mezikód se stane vstupem JVM.
- JVM spustí aplikaci na dané platformě.



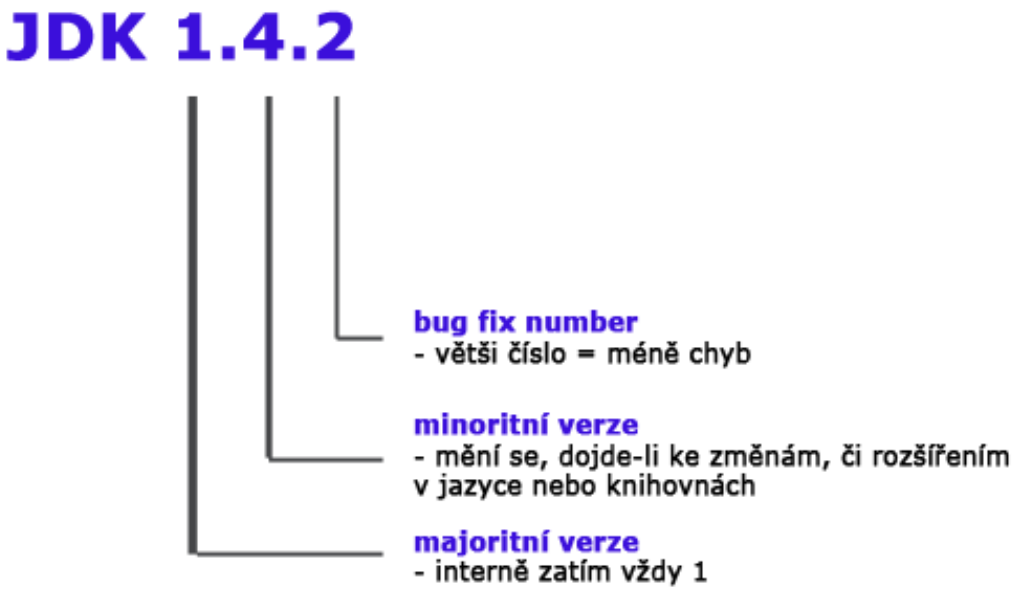
Obrázek 1 - Ukázka životního cyklu Java programu[5]

[5] Zdroj: Dostupný na WWW http://viralpatel.net/blogs/2008/12/java-virtual-machine-an-inside-story.html

2.4 Java Runtime Environment (JRE) a Java Development Kit (JDK)

- **JRE** - je prostředí, které je nutné mít nainstalované pro správný běh aplikací. Obsahuje zásuvné moduly pro prohlížeče a také standardní knihovny. Pokud nechcete Java aplikace programovat, ale jen spouštět, toto prostředí je dostačující.
- **JDK** – někdy bývá označován i jako *SDK*. Obsahuje *JRE* a přidá programovací nástroje, jako jsou např. kompiler mezikódu a také debugger[6] + další vývojové nástroje pro programátory.

Číslování JRE a JDK je shodné a má následující význam [2]:



Obrázek 2 - Ukázka významu číslování verzí JRE a JDK[7]

„Ke změnám v jazyce došlo v JDK 1.1 (např. přidány vnořené třídy), JDK 1.2 zavádí modifikátor strictfp, JDK 1.3 se liší pouze v knihovnách a utilitách, v JDK 1.4 byla přidaná možnost aserce. V JDK 1.5 je novinek nejvíce (genericita, auto un/boxing, anotace atd.).

U verze JDK 1.2 se Sun rozhodl zamotat všem hlavy a zavedl nová jména, takže JDK 1.2 až 1.4 se nazývají Java 2 Standard Edition (J2SE). Pro ještě větší zmatek se JDK 1.5 též nazývá JDK 5.0 a celý název zní Java Platform, Standard Edition (Java SE).[8]“

[6] Program používaný pro detekci chyb v jiném programu
[7] Zdroj: Vlastní
[8] Zdroj: Dostupný na WWW http://v1.dione.zcu.cz/java/uvod.html

3 Uživatelská část aplikace

3.1 Úvod do problematiky

V dnešní době jsou pro uživatele počítačů nejcennější data. Každý z nás s nimi denně pracuje – vytváří, maže, kopíruje, přesouvá, stahuje z internetu, atd. Pro tyto operace se soubory a adresáři lze použít nástroje, které nám takovou práci umožní zjednodušit nebo urychlit. Základní myšlenka byla nahradit práci s příkazovým řádkem určitým intuitivním prostředím, využívajícím klávesových zkratek, nabídek, ikon, tlačítek, atp. Proto vznikl takzvaný správce souborů, neboli *File Manager*.

Je možno použít některý z desktopových správců souborů, implementovaných přímo ve vámi používaném operačním systému. Touto volbou se vydá větší část uživatelů, ale hlavně ti, kteří používají počítač zejména pro domácí a osobní využití a mají spíše základní znalosti a nároky pro práci s PC. Uživatelé, kteří potřebují zefektivnit proces práce se soubory a adresáři, používají buď jedno panelové správce souborů, nebo dvou panelové.

3.2 O aplikaci

Primárním cílem této práce je vytvořit aplikaci pro snadnou manipulaci se soubory. Aplikace má tu výhodu, že ji lze spustit na více operačních systémech. To obstará jazyk Java, protože je nezávislý na architektuře. Aplikace bude využívat dvou panelového rozhraní. Její vzhled a základní funkce jsou inspirovány asi nepopulárnějším správcem souborů, programem *Total Commander*.

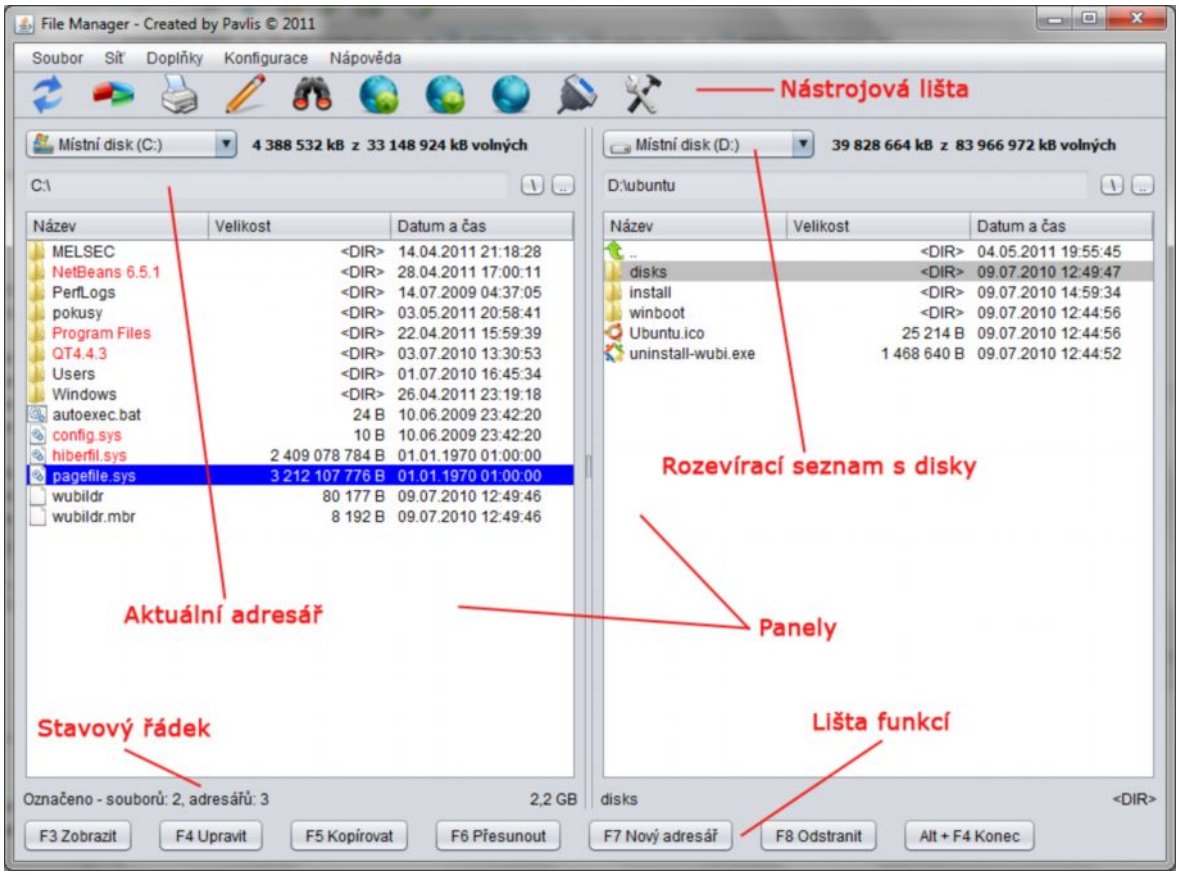
3.3 Hlavní okno

Po spuštění aplikace na vybrané architektuře nás uvítá hlavní okno. Zde se nachází klasické dvou panelové rozhraní, nástrojová lišta, menu, rozevírací seznam s aktuálními paměťovými médii v PC. Vedle rozevíracího seznamu nalezneme také údaje o volné kapacitě paměťového média a velikost paměťového média. Dále zde nalezneme lištu zobrazující aktuální cestu adresáře, ve kterém se právě nacházíme. Na okraji lišty se nachází pomocná tlačítka. Nakonec asi nejdůležitější – lišta funkcí. Ta se nachází na spodním okraji okna. Pomocí tlačítek umístěných v této liště můžeme provádět základní operace se soubory a adresáři jako:

- Zobrazení
- Úprava
- Kopírování
- Přesouvání
- Vytvoření adresáře

- Odstranění

Nad funkční lištou se nachází stavový řádek, který udává informace o aktuálním výběru, či velikosti souborů.



Obrázek 3 - Ukázka hlavního okna aplikace[9]

3.4 Dvou panelové rozhraní

Aplikace nabízí dva panely pro práci se soubory. U obou panelů nalezneme pro každý soubor nebo adresář informace, jako jsou název, velikost (nejedná-li se o adresář) a datum s časem poslední změny (viz Obrázek 3). Mezi panely se nachází oddělovač, který umožní přizpůsobení velikosti každého z panelů.

V panelu se lze pohybovat pomocí šipek nahoru a dolů nebo výběrem levého tlačítka myši. Stiskem pravého tlačítka vyvoláme editaci názvu souboru nebo adresáře. Pro vstup do adresáře můžeme použít klávesu *ENTER* nebo *dvojklik levého tlačítka* myši. Pokud tak učiníme na souboru, dojde k jeho spuštění, a to v programu, který je v operačním systému přidružen příponě spouštěného souboru. Pokud v panelu potřebujeme o(d)značit více souborů, použijeme klávesy *INSERT* nebo *MEZERNÍK*. Mezerník soubor pouze označí, ale insert nás ještě navíc posune o řádek níže (pokud je to ještě možné). K o(d)značení můžeme využít také *levé tlačítko* myši s přidrženou klávesou *CTRL*.

[9] Zdroj: Vlastní

3.5 Nástrojová lišta

Nástrojová lišta, nebo také *ToolBar* umožní rychlejší a efektivnější práci s programem. Lišta obsahuje funkční ikony. Pro popis funkce ikony stačí myš umístit na ikonu a chvíli na ní setrvat. Zobrazí se popis funkce, takzvaný *ToolTip Text*. Lišta je uživatelsky konfigurovatelná – viz 3.5.10 Možnosti.

3.5.1 Znovu načtení disků

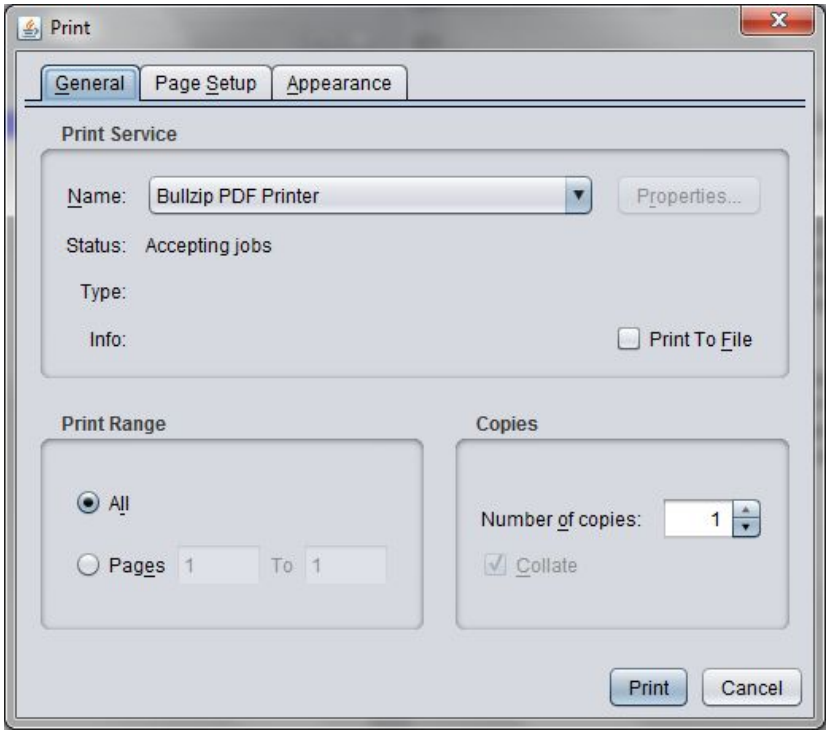
Způsobí aktualizaci rozevíracího seznamu s disky a také panelů. Tato nabídka má význam, když např. připojíme/odpojíme USB paměťové médium, síťové disky nebo když vložíme (odebereme) CD/DVD.

3.5.2 Vybrat vše

Běžně známá volba, zde vybere všechny soubory a adresáře v aktivním panelu (kromě ikony *O úroveň výš*) a označí je barvou pro vybrané soubory. Tato barva je uživatelsky konfigurovatelná – viz 3.5.10 Možnosti.

3.5.3 Tisk seznamu souborů

Při výběru této ikony se zobrazí dialogové okno, které nabídne dodatečné možnosti před tiskem. Můžeme zde vybrat tiskárnu, počet kopií, orientaci papíru, atd. Tisknout se bude obsah otevřeného adresáře v aktivním panelu. Tato volba má přiřazenou klávesovou zkratku CTRL + P.



Obrázek 4 - Ukázka tisku seznamu souborů[10]

[10] Zdroj: Vlastní

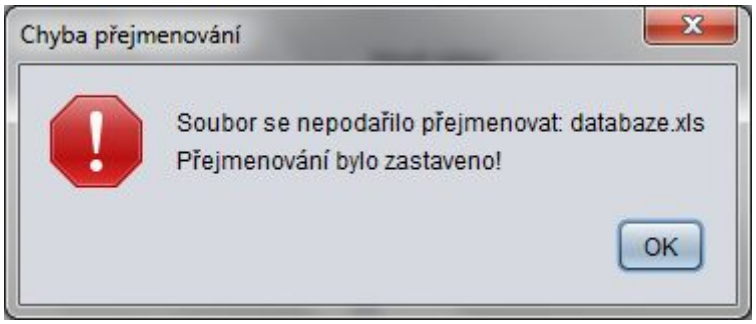
3.5.4 Hromadné přejmenování

Před výběrem této funkce označíme soubory, které chceme hromadně přejmenovat. Učinit tak můžeme za pomoci kláves *INSERT* nebo *MEZERNÍK*. Označit soubory lze i myší, a to levým tlačítkem s přidrženou klávesou *CTRL*. Pokud nezvolíme ani jednu možnost a funkci přesto spustíme, pak se za vybraný soubor nebo adresář považuje poslední aktivní. Pokud posledním aktivním adresářem je volba *O úroveň výše*, program nás upozorní, že nebyl vybrán žádný soubor.

Po vybrání souborů a kliknutí na ikonu *Hromadné přejmenování* aplikace zobrazí formulář. Vlevo najdeme kolonku *Maska přejmenování*, která stanoví masku, podle které jsou vytvořeny nové názvy souborů. Lze využít i zástupné znaky, a to:

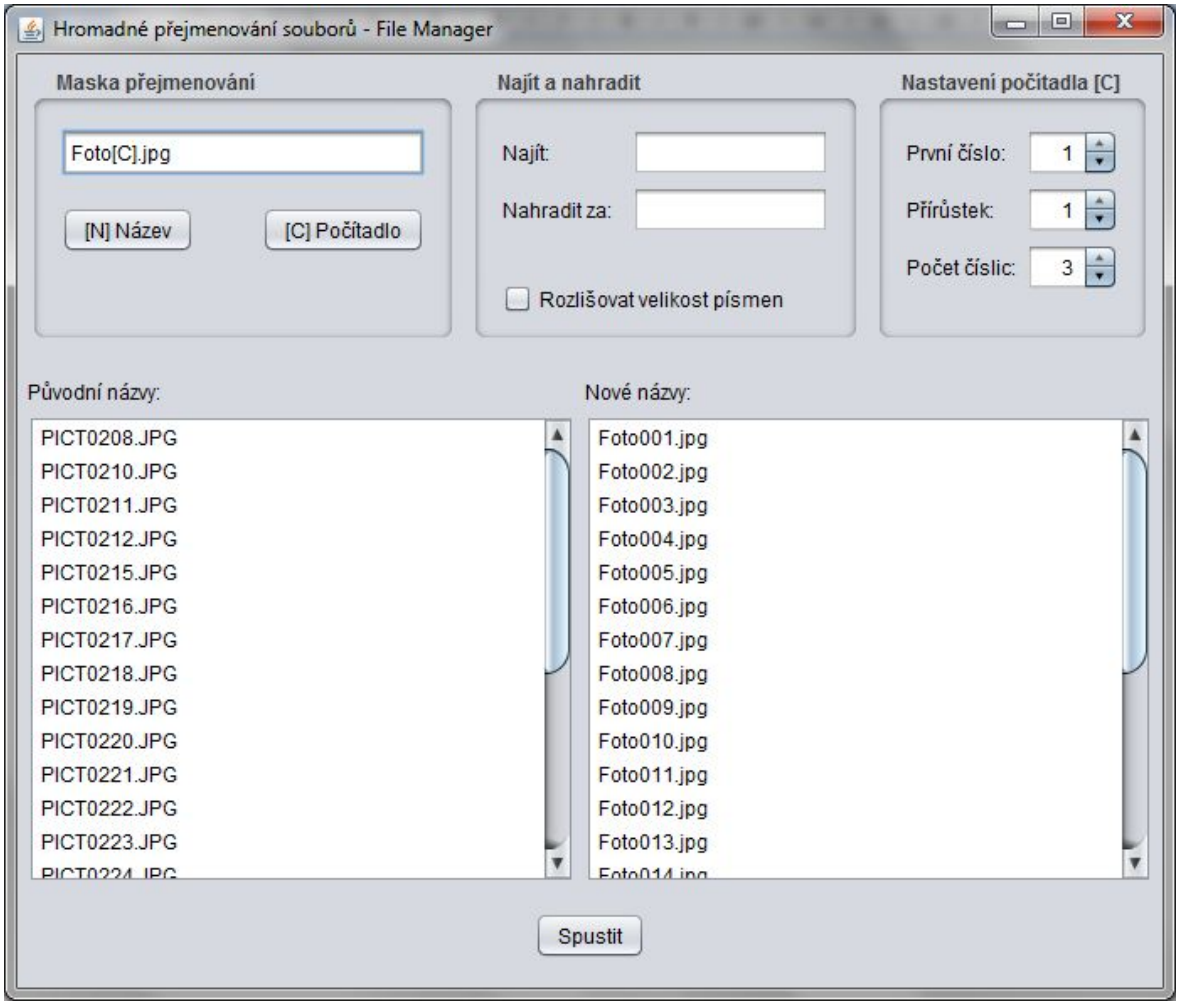
- **[N]** – Původní název souboru, nebo adresáře.
- **[C]** – Počítadlo.

Oba zástupné znaky lze zapsat ručně nebo vložit pomocnými tlačítky. Počítadlo lze přizpůsobit v oblasti *Nastavení počítadla*. Dále se na formuláři nachází kolonky pro nahrazování textu v novém názvu. Zaškrtnutím políčka *Rozlišovat velikost písmen* lze upřesnit nahrazování. Tlačítkem *Spustit* se provede příkaz hromadného přejmenování. Pokud se operace z určitého důvodu nepodaří (soubor nelze přejmenovat, protože už neexistuje, nemáme dostatečná práva, atd.) budeme upozorněni dialogovým oknem.



Obrázek 5 - Ukázka neúspěšného přejmenování souborů[11]

[11] Zdroj: Vlastní

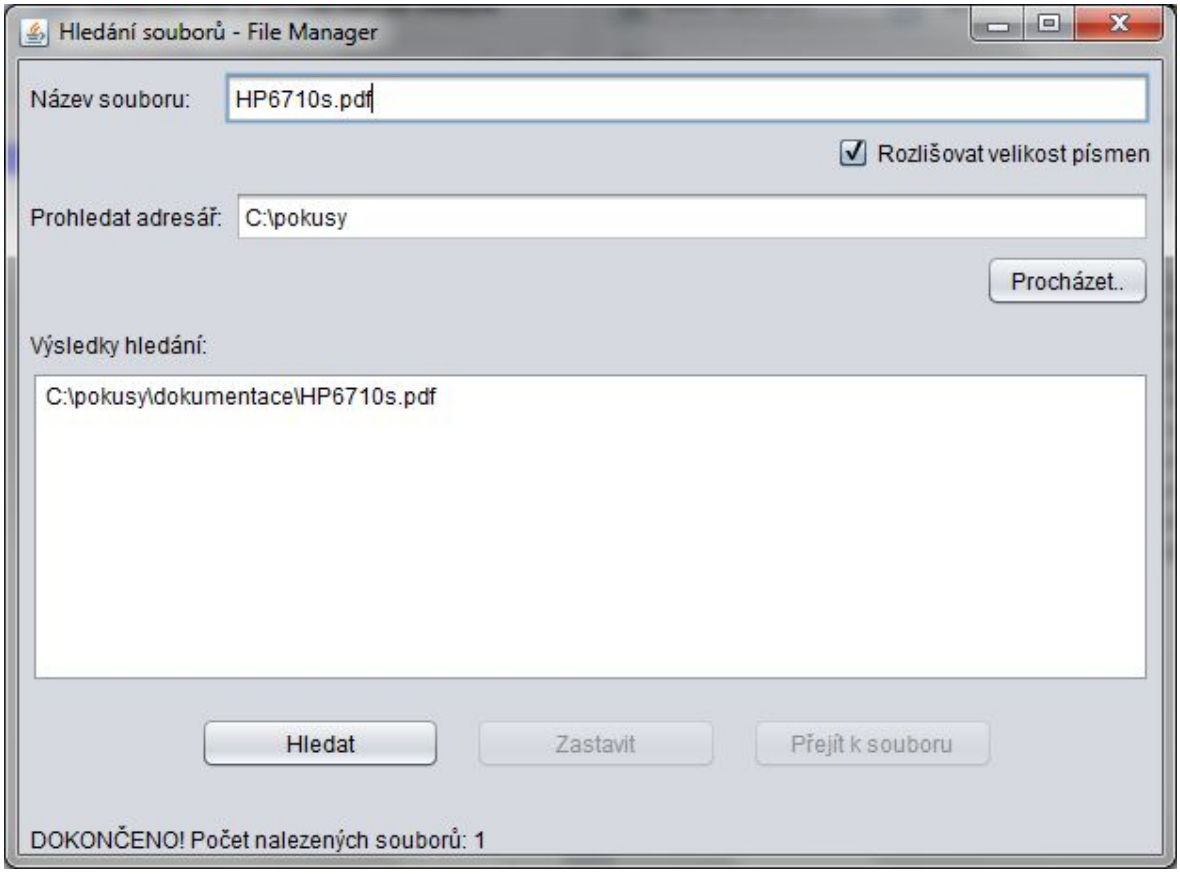


Obrázek 6 - Ukázka formuláře hromadného přejmenování[12]

3.5.5 Hledat

Další položka nástrojové lišty je funkce *Hledat*. Zobrazí okno, kde je již předvyplněna cesta prohledávaného adresáře. Je shodná s adresářem aktivního panelu před výběrem této funkce. Zde zadáme jméno hledaného souboru nebo adresáře. Vyhledávané soubory můžeme omezit volbou *Rozlišovat velikost písmen*. Cestu prohledávaného adresáře můžeme nastavit buď ručně, nebo využít tlačítka *Procházet* a zvolit adresář z dialogového okna. Celý proces zahájíme stisknutím tlačítka *Spustit*. Přerušit jej můžeme tlačítkem *Zastavit*. V průběhu vyhledávání se začne plnit list *Výsledky hledání*. Po dokončení vyhledávání se na dolním okraji okna zobrazí text *„DOKONČENO! Počet nalezených souborů: X*[13]*"* a program ohlásí konec také pípnutím. Pokud bylo hledání úspěšné, uvidíme celou cestu k nalezenému souboru v listu s výsledky hledání. Po jeho označení a vybrání tlačítka *Přejít k souboru* se na pozadí otevře v aktivním panelu adresář, ve kterém se nachází nalezený soubor a zároveň se také označí.

[12] Zdroj: Vlastní
[13] X zde zastupuje počet nalezených souborů

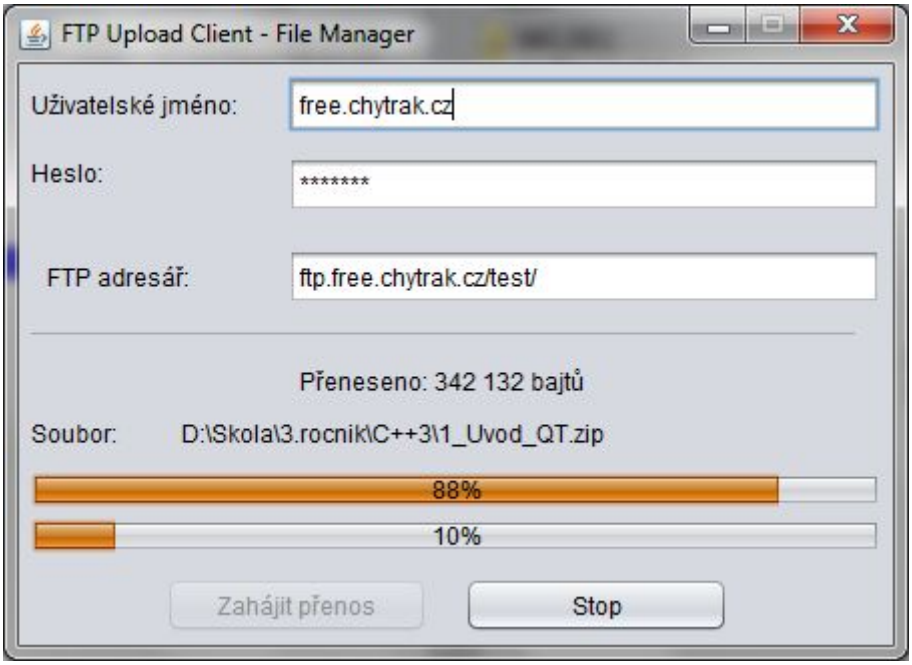


Obrázek 7 - Ukázka okna pro vyhledávání souborů[14]

3.5.6 FTP Upload

Tato funkce nahraje vybrané soubory na server, a to pomocí protokolu *FTP (File Transfer Protocol)*. Před jejím výběrem označíme požadované soubory určené ke kopírování na server. Dialogové okno může vypadat potom následovně:

[14] Zdroj: Vlastní



Obrázek 8 - Ukázka dialogového okna pro upload souborů[15]

Jak je na první pohled vidět, je nutno vyplnit pole:

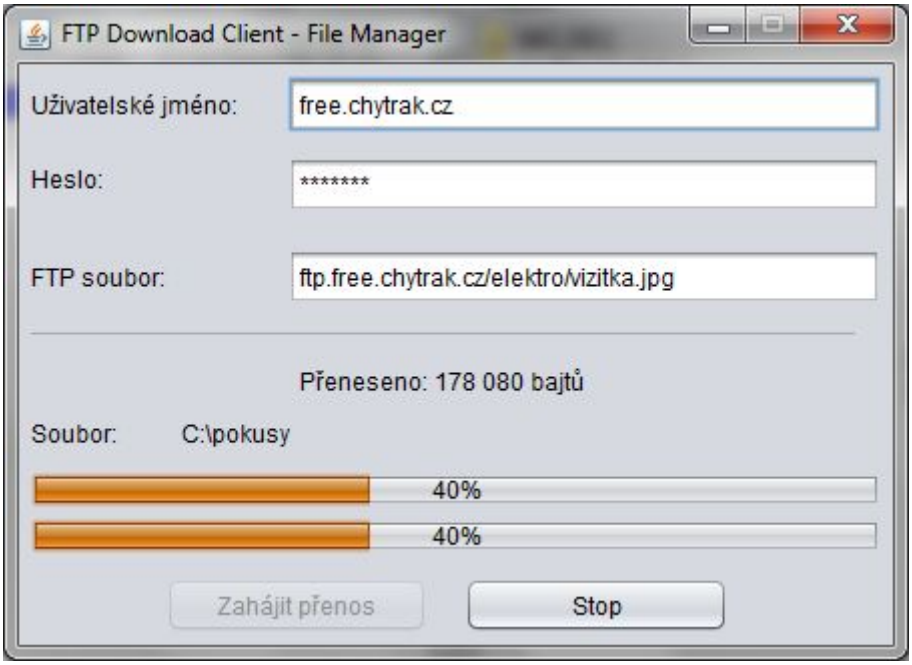
- **Uživatelské jméno** – jméno získané od administrátora serveru nebo pověřené osoby.
- **Heslo** – heslo získané od administrátora serveru nebo pověřené osoby. Toto pole má specifické vlastnosti. Heslo je maskováno zástupným znakem * a také standardní možnost kopírování tohoto pole (CTRL + C) je zde z bezpečnostních důvodů potlačena.
- **FTP adresář** – existující adresář FTP serveru, kde se uloží požadované soubory. Je třeba dbát na to, že většina serverů rozlišuje velikost písmen. Tzn. že adresář *„Test/“* není to stejné jako adresář *„test/.“*

Pod částí s připojením se nachází část přenosová. Jak znázorňuje Obrázek 8 přenos souborů již začal. Můžeme zde vidět celkový počet přenesených bajtů, absolutní cestu aktuálně přenášeného souboru a také procentuálně vyjádřený stav dokončení procesu. Horní procenta zachycují dokončení přenosu právě kopírovaného souboru a dolní procenta dokončení kopírování všech požadovaných souborů.

3.5.7 FTP Download

Funkce je obdoba předešlé. Rozdíl je v tom, že soubor z lokálního PC nekopírujeme na server, nýbrž ze serveru do PC. Před vybráním této funkce otevřeme v panelu aplikace cílový adresář, do kterého se má požadovaný soubor nakopírovat.

[15] Zdroj: Vlastní



Obrázek 9 - Ukázka dialogového okna pro download souborů[16]

I zde je třeba vyplnit následující údaje:

- **Uživatelské jméno** – jméno získané od administrátora serveru nebo pověřené osoby.
- **Heslo** – heslo získané od administrátora serveru nebo pověřené osoby. Toto pole má specifické vlastnosti. Heslo je maskováno zástupným znakem * a také standardní možnost kopírování tohoto pole (CTRL + C) je zde z bezpečnostních důvodů potlačena.
- **FTP soubor** – existující soubor na serveru, který má být zkopírován do PC. Opět je třeba dodržet velikost písmen.

Okno disponuje vlastností zobrazení přenesených bajtů, zobrazení zvoleného cílového adresáře a také procentuální dokončení úlohy.

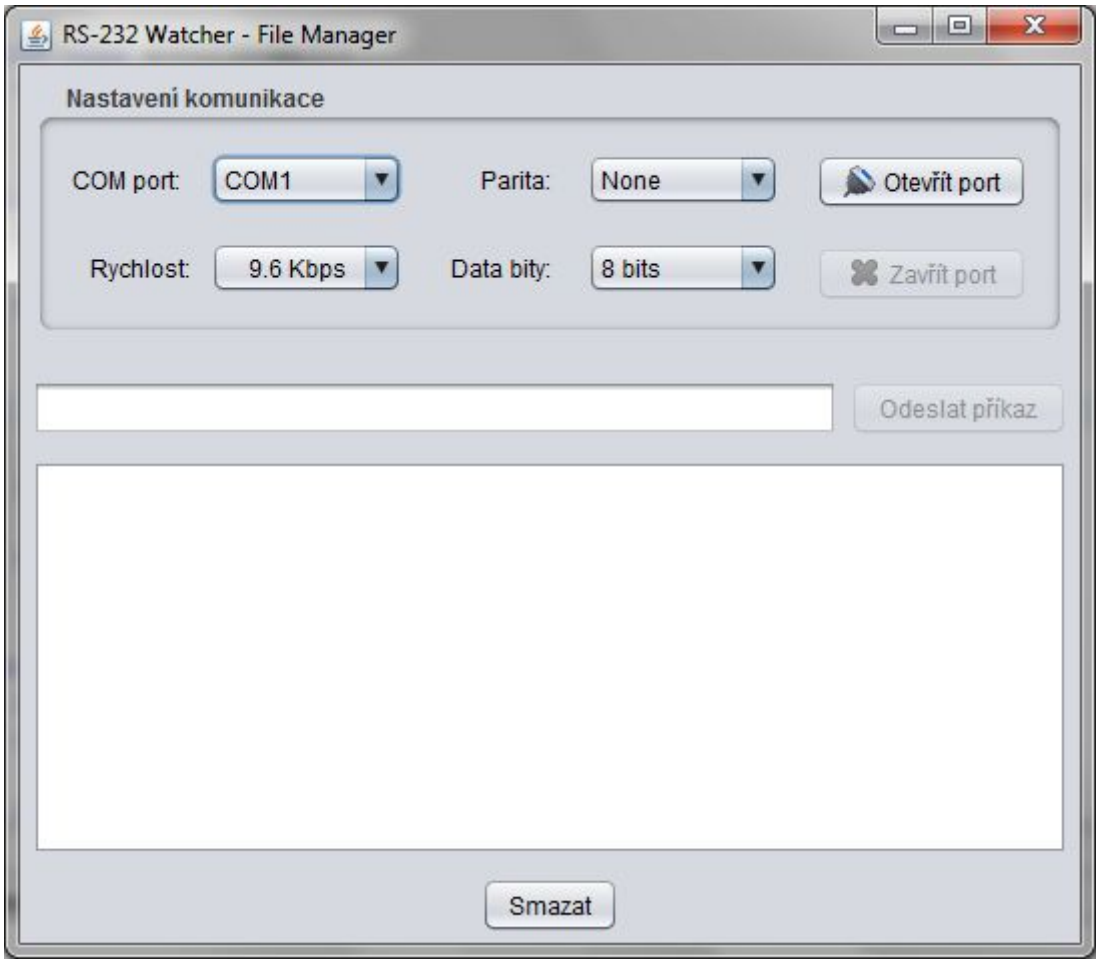
3.5.8 Otevřít prohlížeč

Funkce je zde pro rychlejší přístup k výchozímu prohlížeči webových stránek. Spustí prohlížeč a implicitně zobrazí adresu *http://www.google.cz*.

3.5.9 RS-232 Watcher

Tak jako někteří jiní správci souborů zahrnují do svých rozšířených možností komunikaci přes LPT[17] (např. Total Commander), tak i tato aplikace umožňuje obdobného rozšíření. Je zde umožněna sériová komunikace, nebo-li *RS-232*. Po výběru této možnosti se zobrazí následující dialogové okno:

[16] Zdroj: Vlastní
[17] Paralelní port pro komunikaci zařízení nebo PC



Obrázek 10 - Ukázka okna se sériovou komunikací[18]

Po vytvoření okna uvidíme na jeho dolním okraji oznámení, že se prohledávají COM porty. V tu chvíli aplikace zjišťuje, které sériové porty jsou ve vašem PC dostupné. Po dokončení operace nám okno umožní volbu *Otevřít port*. Před stiskem tohoto tlačítka si nastavíme sériovou komunikaci a to:

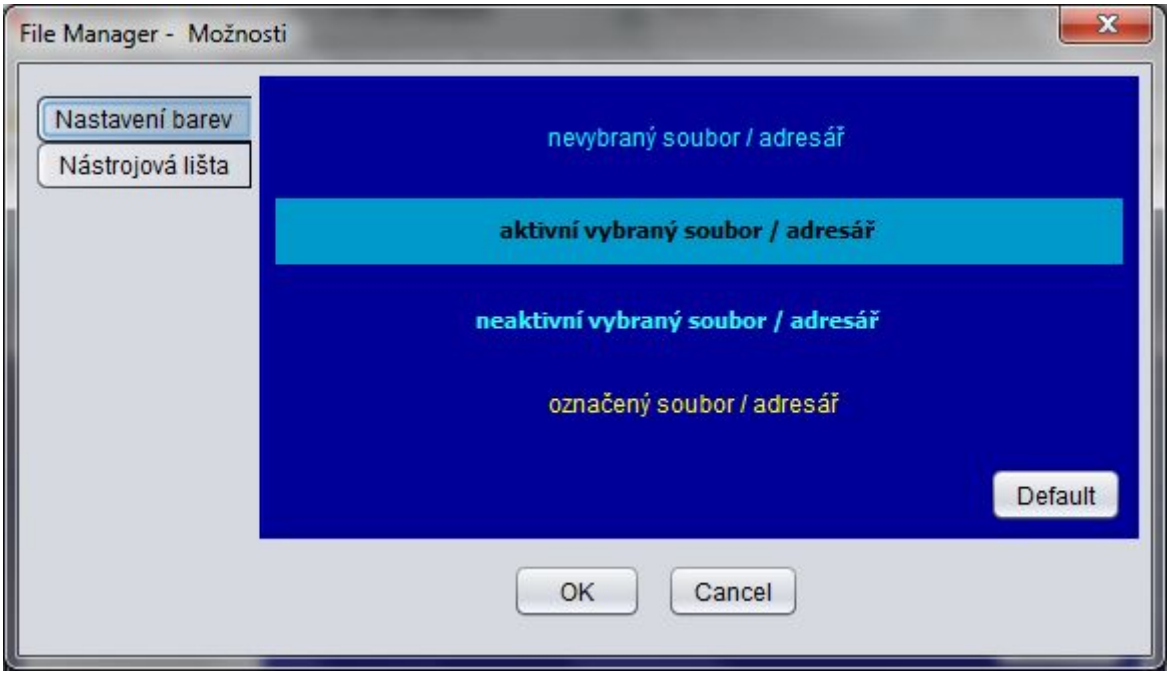
- COM port
- Rychlost komunikace
- Paritu
- Datové bity

Po úspěšném otevření portu (port může být obsazen jinou aplikací) můžeme na dané zařízení odesílat dotazy prostřednictvím řádku s tlačítkem *Odeslat příkaz*. O něco níže se nachází textové okno, které zachycuje, co připojené zařízení posílá. Pokud před uzavřením okna neuzavřeme sériový port, aplikace to udělá automaticky.

[18] Zdroj: Vlastní

3.5.10 Možnosti

Zde máme možnost upravit uživatelské prostředí aplikace. Upravovat lze nastavení barev nebo nástrojovou lištu. Úprava *Nastavení barev* může mít například takovýto vzhled:

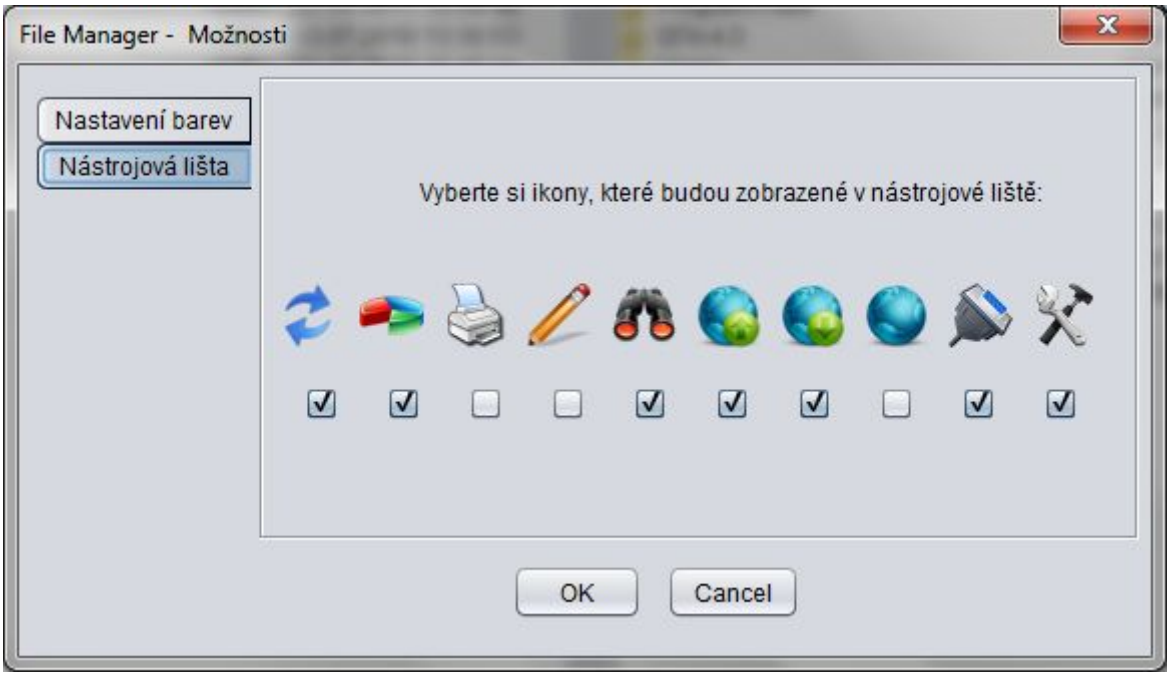


Obrázek 11 - Ukázka okna nastavení barev[19]

Pro změnu barvy stačí kliknout levým tlačítkem myši na požadovaný text nebo pozadí. Zobrazí se modální[20] dialogové okno s výběrem barvy. Tlačítkem *Default* lze obnovit původní barevné nastavení. Tlačítko *OK* uzavře okno, změní barvy v aplikaci a zároveň také toto nastavení barev uloží do souboru *config.dat*. Ukládání je nezbytné z důvodu, aby při opětovném spuštění aplikace zůstalo zachováno nastavení barev. Tlačítko *Cancel* ignoruje úpravy provedené v kartě *Možnosti* a zanechá původní nastavení aplikace.

Kromě *Nastavení barev* obsahuje karta *Možnosti* také volbu *Nástrojová lišta*. Zde si můžeme zvolit, které ikony chceme mít v nástrojové liště. Při volbě *OK* dojde opět k uložení konfigurace do souboru *config.dat* a také k aktualizování seznamu ikon v nástrojové liště.

[19] Zdroj: Vlastní
[20] Modální okno je takové okno, které zůstává neustále aktivní a nad všemi ostatními okny



Obrázek 12 - Ukázka nastavení nástrojové lišty[21]

3.6 Menu

Funkce menu jsou přehledně umístěny do jednotlivých kategorií pro snadnou orientaci. Menu aplikace nabízí následující kategorie:

- Soubor
- Síť
- Doplňky
- Konfigurace
- Nápověda

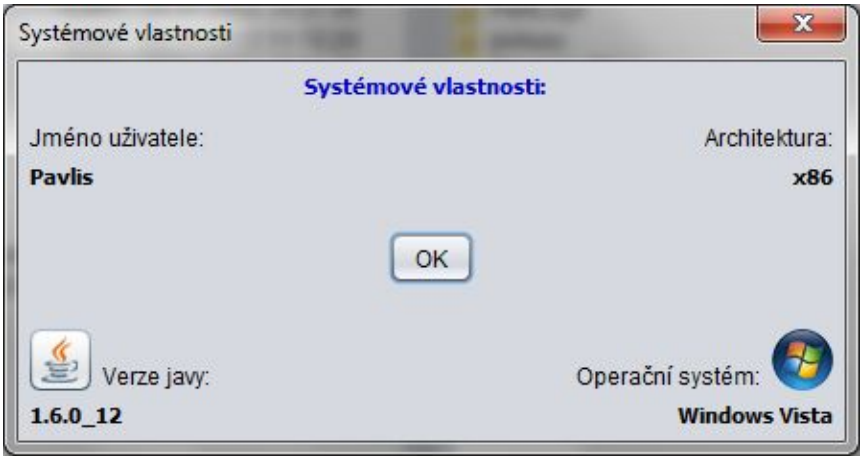
Soubor

Kategorie soubor nabízí tyto možnosti:

- Znovu načtení disků – viz kapitola 3.5.1.
- Hledat – viz kapitola 3.5.5.
- Hromadné přejmenování – viz kapitola 3.5.4.
- Tisk seznamu souborů – viz kapitola 3.5.3.

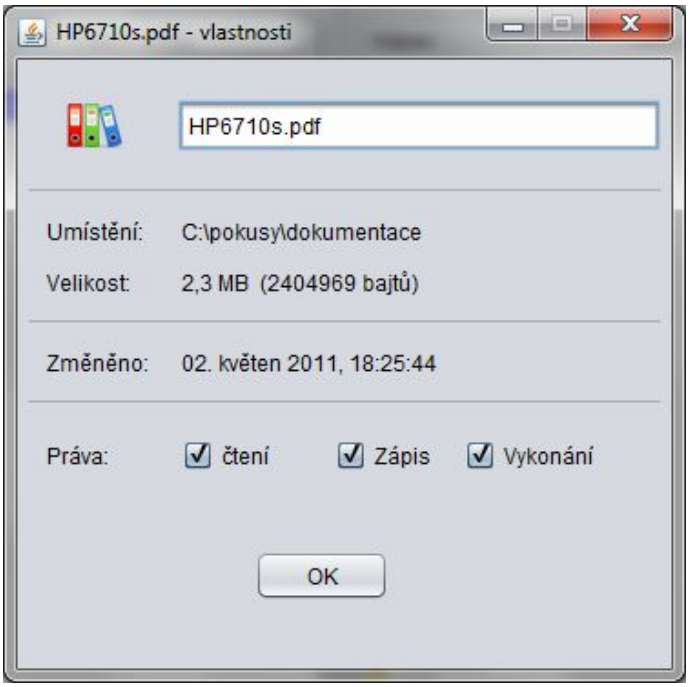
[21] Zdroj: Vlastní

- Systémové vlastnosti – vyvolá okno, které obsahuje některé vlastnosti systému. Jsou to jméno aktuálně přihlášeného uživatele, verze Javy (JDK), architektura operačního systému a nakonec verze systému. Verze systému je doprovázena ikonou systému. Aplikace obsahuje ikony pro operační systém MS Windows, Linux a Mac. Pro ostatní OS zobrazí univerzální ikonu.



Obrázek 13 - Ukázka systémových vlastností[22]

- Vlastnosti – dialogové okno, které zobrazuje vlastnosti označených souborů nebo adresářů. Zde můžeme změnit také práva souboru (pokud k tomu máme oprávnění). Při výběru několika souborů nebo adresářů je také zobrazen počet souborů a počet adresářů. Dialogové okno lze také vyvolat klávesovou zkratkou *ALT + ENTER*.



Obrázek 14 - Ukázka vlastností souborů[23]

[22] Zdroj: Vlastní

- Konec – Ukončí aplikaci.

Síť

V kategorii síť nalezneme následující možnosti:

- FTP Upload – viz kapitola 3.5.6.
- FTP Download – viz kapitola 3.5.7.

Doplňky

Zde je jediná volba:

- RS-232 Watcher – viz kapitola 3.5.9.

Konfigurace

Konfigurace obsahuje:

- Možnosti – viz kapitola 3.5.10.
- Nástrojová lišta – výběr volby nastaví zobrazení/skrytí nástrojové lišty.

Nápověda

Kategorie nabízí možnost:

- O programu – formulář s informacemi o programu.

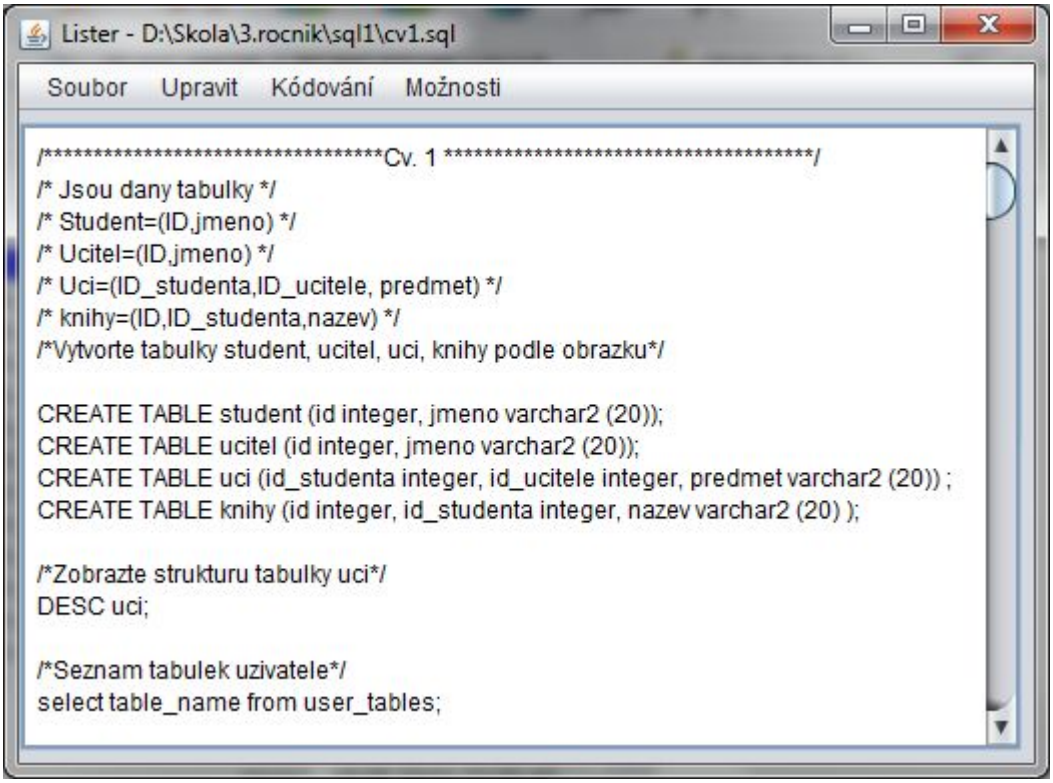
3.7 Práce se soubory

3.7.1 F3 Zobrazit

Funkce, která je dostupná pouze pro soubory, nikoliv adresáře. Načte v textové podobě aktivní soubor z aktivního panelu do dialogového okna *Lister*. Okno má vlastní menu a v něm několik možností. Mezi hlavní patří:

- Možnost uložit si text jako nový textový soubor (klávesová zkratka *CTRL + S*).
- Změna kódování souboru – Cp1250, UTF8, ISO8859_2.
- Možnost výběru zalomení řádků.

[23] Zdroj: Vlastní



Obrázek 15 - Ukázka okna pro zobrazení souborů[24]

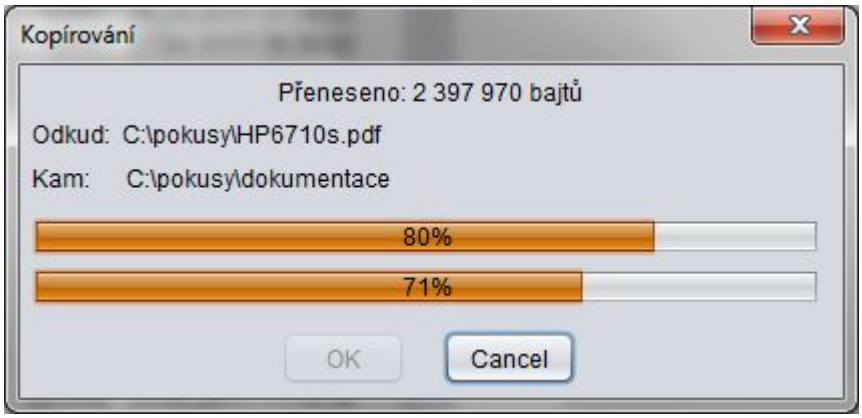
3.7.2 F4 Upravit

Volba opět pouze pro soubory. Po kliknutí na tlačítko *F4 Editovat* se soubor načte v textové podobě a otevře se v textovém nebo výchozím editoru OS pro daný typ souboru. Pokud tuto funkci budeme chtít aplikovat na adresáře, aplikace nás upozorní, že operace není možná.

3.7.3 F5 Kopírovat

Tato funkce pracuje s oběma panely zároveň. Jeden z nich má funkci „zdrojového“ a druhý „cílového.“ Který bude který, záleží na vás. Ve zdrojovém panelu označíme soubory nebo adresáře, které chceme kopírovat. K označení lze použít klávesu *INSERT*, *MEZERNÍK*, nebo *levé tlačítko myši + CTRL*. Pokud se jedná o jeden soubor nebo adresář, stačí, když zůstane aktivní. V cílovém panelu potom otevřeme adresář, do kterého se soubory mají kopírovat. Před zavoláním funkce musí zůstat aktivní zdrojový panel. Pro úspěšné kopírování je nutno mít právo čtení a zápisu u souborů a adresářů, se kterými pracujeme. Jinak aplikace oznámí neúspěšnou operaci.

[24] Zdroj: Vlastní



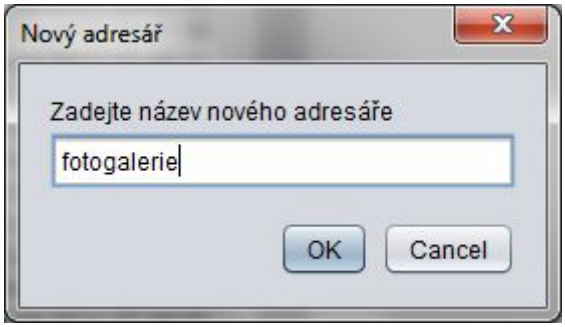
Obrázek 16 - Ukázka okna pro kopírování souborů[25]

3.7.4 F6 Přesunout

Vychází z principu kopírování – viz kapitola 3.7.3. Rozdíl je v tom, že zdrojové soubory jsou po zkopírování do cílového adresáře odstraněny. Opět je nutno mít dostatečná práva při provádění této operace.

3.7.5 F7 Nový adresář

Vytvoří nový adresář do právě otevřeného adresáře v aktivním panelu. Dialogovým oknem budeme vyzváni k zadání názvu adresáře. Název nesmí být prázdný, nesmí se rovnat dvěma tečkám a nesmí se opakovat s názvem jiného adresáře uvnitř právě otevřeného adresáře. V opačném případě nás aplikace upozorní chybovou hláškou.



Obrázek 17 - Ukázka vytvoření nového adresáře[26]

3.7.6 F8 Odstranit

Funkce pro odstranění souborů a adresářů. Soubory a adresáře určené k odstranění můžeme opět označit několika způsoby. Klávesou *INSERT*, *MEZERNÍK* nebo *levým tlačítkem myší + CTRL*. Aplikace nás před odstraněním ještě vyzve pro potvrzení operace. Po té soubory odstraní z paměťového média.

3.8 Klávesové zkratky

Klávesové zkratky jsou kombinace určených kláves, které po stisknutí vyvolají nějakou akci. Urychlují práci s aplikací a umožňují tak efektivnější činnost uživatele.

[25] Zdroj: Vlastní
[26] Zdroj: Vlastní

3.8.1 Klávesové zkratky

Následující seznam pojednává o všech naprogramovaných klávesových zkratkách, kterými aplikace disponuje.

Tabulka 1 - Význam klávesových zkratek aplikace

Klávesová zkratka	Popis funkce
ENTER	Otevře vybraný adresář / Spustí vybraný soubor.
BACKSPACE	Vrátí se „O úroveň výš“ – otevře nadřazený adresář.
ŠIPKA NAHORU	Zaktivní adresář (soubor), který je nad aktuálně vybraným.
ŠIPKA DOLŮ	Zaktivní adresář (soubor), který je pod aktuálně vybraným.
HOME	Zaktivní první adresář (soubor) v panelu.
END	Zaktivní poslední adresář (soubor) v panelu.
CTRL + A	Označí všechny soubory (adresáře).
INSERT	Označí aktivní adresář (soubor) a pokud je to možné, nastaví jako aktivní další následující adresář (soubor).
MEZERNÍK	Označí aktivní adresář (soubor).
DELETE	Odstraní aktivní (vybrané) soubor(y) / adresář(e).
TAB	Mění aktivnost panelu mezi levým a pravým panelem.
ALT + ENTER	Zobrazí dialogové okno „Vlastnosti“ pro vybrané soubory (adresáře).
CTRL + P	Tiskne seznam souborů aktivního panelu.
CTRL + F	Zobrazí okno pro vyhledávání souborů.
CTRL + R	Aktualizuje seznam s disky a otevře v panelech kořenové adresáře.
CTRL + O	Zobrazí okno pro nastavení možností aplikace.
F3	Zobrazí okno pro zobrazení aktivního souboru.
F4	Edituje aktivní soubor v textovém editoru.
F5	Kopíruje vybrané adresáře (soubory).
F6	Přesouvá vybrané adresáře (soubory).
F7	Vytvoří nový adresář.
F8	Odstraní aktivní (vybrané) soubor(y) / adresář(e).
ALT + F4	Ukončí aplikaci.

3.8.2 Tlačítka myši

Seznam funkcí tlačítek myši, kterými je možno aplikaci obsluhovat.

Tabulka 2 - Význam tlačítek myši v aplikaci

Tlačítko myši	Popis funkce
LEVÉ	Změní aktivní adresář (soubor).
LEVÉ – DVOJKLIK	Otevře vybraný adresář / Spustí vybraný soubor.
LEVÉ + CTRL	Označí aktivní adresář (soubor).
PRAVÉ	Umožní přejmenování aktivního adresáře (souboru).

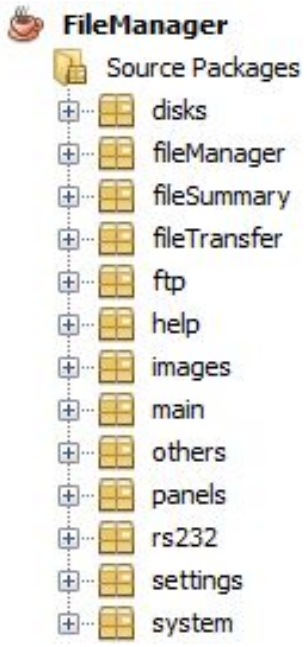
4 Programátorská část aplikace

4.1 Implementace

Na samotném začátku práce bylo třeba zvolit programovací jazyk a nějaké vývojové prostředí. Jako jedna z možností se nabízel jazyk C++ s nadstavbou QT a vývojové prostředí *Microsoft Visual Studio*. Alternativou byl jazyk Java a vývojové prostředí *JDeveloper* nebo *NetBeans*. První pokusy byly v jazyku C++ s QT nadstavbou. QT má asi nejlepší manuál, co znám. Přesto se mi práce zdála zbytečně komplikovaná. Sáhl jsem tedy po multiplatformním Jazyku Java.

Jazyk Java mě zaujal svou jednoduchostí, přehledností a také kvůli své nezávislosti na architektuře. Tím byly splněny mé požadavky, aplikace je spustitelná na více operačních systémech. Tato práce je napsána ve vývojovém prostředí *NetBeans IDE 6.9.1*. Prostředí je open source a ke stažení zdarma na domovské stránce[27].

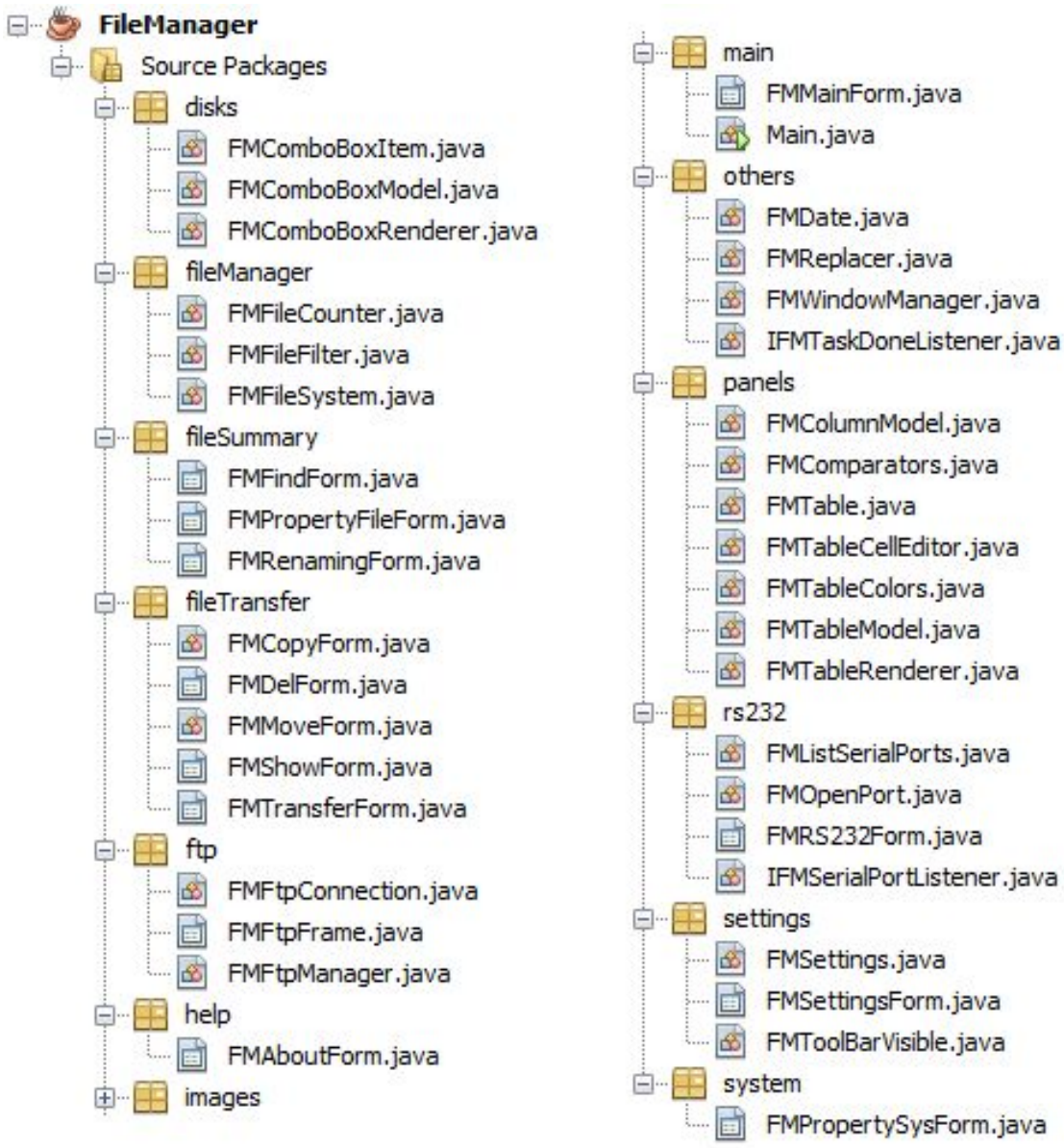
Aplikace obsahuje mnoho tříd, a proto nebudu popisovat všechny jejich metody a atributy. Ty jsou všechny k nalezení jako přiložený zdrojový kód – viz Příloha. Rozhodl jsem se specifikaci rozdělit do balíčků aplikace a popsat pouze základní charakteristiku problémů, které řeší jednotlivé třídy. Následující část práce tedy charakterizuje jednotlivé balíčky a jejich třídy. Seznam balíčků vystihuje Obrázek 18.



Obrázek 18 - Seznam balíčků aplikace[28]

[27] http://netbeans.org/index_cs.html
[28] Zdroj: Vlastní

Seznam tříd v jednotlivých balíčcích charakterizuje Obrázek 19.



Obrázek 19 - Seznam tříd aplikace[29]

Obsahu balíčku *images* není zobrazen, protože balíček neobsahuje žádné třídy. Zahrnuje v sobě pouze ikony formátu **.png*, které aplikace používá pro své zpřehlednění.

Jak můžeme dále vidět, v názvech vlastních tříd je použito prefixu *„FM.“* Prefix umožňuje při „napovídání“ vizuálně oddělit třídy definované mnou a třídy defaultní. Je to z důvodu snadné orientace a přehlednosti.

[29] Zdroj: Vlastní

4.2 Balíček disks

Tento balíček zajišťuje funkci výběru paměťových médií z rozevíracího seznamu úvodního formuláře aplikace. Dále detekci paměťových médií v PC, získání jejich systémových ikon a názvů a nakonec následnému vykreslení v ComboBoxu. Balíček obsahuje třídy:

- **`FMComboBoxItem`** – jak název napovídá, jedná se o třídu, která zastupuje prvek ComboBoxu. Tato třída obsahuje metody pro zjištění názvu paměťového média a také k získání systémových ikon.
- **`FMComboBoxModel`** – třída, u které je využita dědičnost od třídy `DefaultComboBoxModel`. Její konstruktor vytváří objekty třídy `FMComboBoxItem` a naplní jimi model ComboBoxu.
- **`FMComboBoxRenderer`** – i zde je využito dědičnosti a to od třídy `DefaultListCellRenderer`. Metoda s mnoha parametry, kterou jsem překryl vlastní, se nazývá `getListCellRendererComponent()`. Zajistí správné vykreslení prvku ComboBoxu i s nápovědným ToolTip textem.

4.3 Balíček fileManager

Balíček zajišťující hlavní funkce pro operace se soubory. Třída `FMFileCounter` uchovává ve formě atributu informaci o počtu souborů a o počtu adresářů. Dále se zde nachází třída `FMFileFilter` implementující rozhraní `java.io.FileFilter`. Slouží k filtrování souborů a adresářů, které se zobrazují v panelech. Obsahuje metodu `accept(File f)`, která vrací datový typ `boolean` a udává informaci o tom, zda se má se souborem pracovat. V mém případě je tato metoda překryta tak, aby se nepracovalo se soubory a adresáři, které mají atribut *hidden*. Snad nejdůležitější třída aplikace je abstraktní třída `FMFileSystem`. Její veřejné statické metody jsou:

- `boolean runFile(final File file)`
- `void editFile(final File file) throws IOException`
- `static void saveDataInToFile(String path, Object data) throws IOException`
- `Object readDataFromFile(String path) throws FileNotFoundException, Exception`
- `String diskProperty(final File disk)`
- `static long getDirSize(File[] files)`
- `static long getDirSize(File dir)`
- `static FMFileCounter countFile(File[] files)`
- `String sizeFile(long bytes)`
- `File makeDirectory(final File file, final String name) throws IOException`

- `File makeDirectory(final File dir) throws IOException`
- `copyFile(File fromFile, File toFile) throws IOException`
- `copy (InputStream in, OutputStream out) throws IOException`
- `boolean rename(File oldFile, File newFile)`
- `boolean delete(File delFile)`

Význam metod vyplývá z jejich názvů a parametrů. Třída obsahuje také několik privátních metod a dvě abstraktní metody. Ty je nutno naprogramovat při vytvoření konstruktoru této třídy. Jsou to:

- `void bytesWritten(long bytes)`

- `void currentFileSize(long size)`

Tyto metody jsou volány v průběhu kopírovaní datových proudů a také instancí tříd `File`. První posílá již zapsané bajty a druhá aktuální velikost právě kopírovaného souboru, či datového toku. Tyto informace jsou nezbytné pro zobrazování takzvaného *progressBaru*, který uživateli zobrazuje stav dokončení přenosu souborů, či datových toků.

4.4 Balíček fileSummary

Jedná se o souhrnné procesy se soubory jako vyhledávání, hromadné přejmenování a detailní informace o vlastnostech souboru (název, umístění, velikost, datum poslední změny a práva). Balíček obsahuje pouze formuláře, což jsou třídy zděděné od třídy `JFrame`.

`FMFindForm` – tento formulář je navíc i třída abstraktní. Obsahuje totiž abstraktní metodu `goToFile(File findFile)`, kterou je potřeba nadefinovat. Metoda je volána, když výsledek hledání souborů je pozitivní a uživatel zvolil pro nalezený soubor funkci *Jít k souboru*. Formulář potom může přistoupit k hlavnímu oknu a předat mu referenci na nalezený soubor, který se má zobrazit v panelu.

Mimo jiných mnoha metod a atributů třída využívá pro svou činnost vnořenou třídu `FindTask`. Tato vnořená třída implementuje rozhraní `Runnable`, a to z důvodu, že průběh vyhledávání se provádí v novém vlákně. To je důležité zejména proto, že uživateli během procesu vyhledávání souborů udává informace o tom, který adresář a soubor se právě prohledává (zobrazuje *StatusBar* okna) a také průběžnou informaci o tom, který soubor byl již nalezen a vyhověl zadaným kritériím. Instance vnořené třídy `FindTask` se předá jako parametr konstruktoru nového vlákna pro vyhledávání. Toto vlákno se spustí metodou `start()` a zavolá předem nadefinovanou metodu `run()`, která zahájí celý proces.

`FMPropertyFileForm` – třída, která má jako nutný parametr konstruktoru instanci třídy `File`. Může to být i pole těchto objektů. Formulář zobrazuje mnoho

vlastností souborů a adresářů, včetně možnosti změn jejich práv. I tato třída má svoji vnořenou třídu, a to `FileInfoTask`. Ta opět implementuje rozhraní `Runnable`, aby mohla být předána jako parametr novému vláknu. Instance nového vlákna je zde vytvořena z toho důvodu, že např. zjištění velikosti a počtu souborů některého adresáře může trvat delší časový interval. I operační systém *MS Windows* není schopný okamžitě poskytnout informaci o velikosti a počtu souborů např. adresáře „*Windows.*" V dialogovém okně vlastností bude průběžně zobrazovat velikost adresáře, až se proces zastaví a velikost se ustálí. Na podobném principu pracuje i tato třída `FileInfoTask`.

`FMRenamingForm` – formulář pro hromadné přejmenování souborů. Třída obsahuje mnoho metod a atributů pro zobrazení jména, umístění, velikosti, poslední změně a také právech souborů a adresářů. Pro přejmenování souborů používá třídu FMReplacer, které předá původní název souboru + ostatní parametry a pomocí patřičných algoritmů dostane zpět nový název souboru.

4.5 Balíček fileTransfer

Jedná se o balíček, který obsahuje také pouze formuláře. Ty slouží k procesu přenosů souborů a adresářů a také k jejich zobrazení.

Jako hlavní abstraktní třída je `FMTransferForm`. Třída je potomkem třídy `JDialog`. Jeden z atributů této třídy vlastní referenci na instanci třídy `FMFileSystem`. Je to z důvodu možnosti datových přenosů. Třída `FMTransferForm` využívá také statickou metodu třídy `FMFileSystem`, a to `getDirSize(File file)`. Mnohem důležitější je ale využití veřejné metody `copy(File from, File to)`. Ta zajistí fyzický přenos dat. Běží v novém vlákně vnořené třídy `TransferTask`, která implementuje rozhraní `Runnable`. Pro sledování procesu přenosu dat je využito dvou *ProgressBarů*. Třída `FMTransferForm` má atributy s modifikátorem `protected`, a to `File sourceFile` a `File[] destFile` pro možný přístup potomka této třídy. Obsahuje důležité abstraktní metody `transfer()` a `stopTransfer()`, které je při dědění potřeba definovat. Toho využívá třída `FMCopyForm`.

`FMCopyForm` – třída, která je potomkem třídy `FMTransferForm`. Definuje abstraktní metody svého předka a velice jednoduše se potom stará o kopírování souborů a adresářů.

`FMDelForm` – třída zděděná od třídy `JDialog` zajišťující odstranění souborů a adresářů. Pro tuto její schopnost je opět využito nového vlákna formou instance třídy `DelFileTask`.

`FMMoveForm` – tato třída má jako svého předka třídu `FMCopyForm`. Potřebuje totiž obsahovat všechny vlastnosti tříd `FMTransferForm` a `FMCopyForm`. Jako doplňující vlastnost je mazání souborů. Úkolem třídy je totiž zdrojové soubory zkopírovat jako cílové a ty zdrojové následně odstranit.

4.6 Balíček ftp

FMFtpFrame – je formulář, který zajistí přenos souborů mezi serverem a PC. Při konstrukci třídy lze definovat pomocí statické konstanty třídy, o jaké okno se jedná. Možnosti jsou `UPLOAD_FRAME`, nebo `DOWNLOAD_FRAME`. Podle této konstanty pak formulář nastaví vlastnosti okna a také se určí, zda přenos souborů bude probíhat z PC na server, nebo opačně. Jako druhý parametr je třeba předat referenci na objekt(y) třídy `File`. S těmito soubory pak bude třída pracovat. Při uploadu mají tyto soubory zdrojový význam. Při downloadu se pak určí cílový adresář, kam se nakopírují soubory ze serveru. I zde je zapotřebí nového vlákna, které se vytvoří z vnitřní třídy `FtpTask` implementující rozhraní Runnable. Po startu vlákna dojde před přenosem dat k připojení na server. Vytvoří se instance třídy `FMFtpConnection`, které se parametry předá v konstruktoru uživatelsky zadané informace o serveru (uživatelské jméno, heslo a adresa serveru). Tento objekt se předá metodě atributu `FMFtpManager`, která se nazývá `connect()`. Tím dojde k zadání všech přihlašovacích informací a může dojít k samotnému přenosu dat. `FMFtpManager` disponuje metodami `ftpUpload(File file)` a `ftpDownload(File file)`. Tyto metody se opakovaně volají v závislosti na zvoleném parametru typu okna (`UPLOAD_FRAME, DOWNLOAD_FRAME`). Po dokončení všech přenosů dat vlákno zavolá privátní metodu `finish()`. Ta v případě typu okna `DOWNLOAD_FRAME` zavolá metodu všech registrovaných posluchačů (instancí třídy `IFMTaskDoneListener`) s názvem `done()`. Metoda `done()` slouží hlavnímu oknu jako povel k tomu, že se nové soubory již nachází na paměťovém médiu a má provést obnovení (refresh) adresáře.

FMFtpConnection – tento předpis pro objekty uchovává formou atributů přihlašovací údaje k serveru. Vlastní několik privátních metod pro kontrolu a doplnění údajů zadaných uživatelem a jednu pro vytvoření připojení. Neméně důležitá je i veřejná metoda `getOutputStream(File file)`, která vrací referenci objektu třídy `OutputStream`. Tím je zajištěn přístup k datovému toku při uploadu na server. Metoda s opačnou funkcí `getInputStream()` vrací referenci objektu třídy `InputStream`. Ta zpřístupní datový tok pro download souboru.

FMFtpManager – již výše zmiňovaná třída. Obsahuje pouze veřejné metody bez návratového typu `connect(FMFtpConnection conn)`, `ftpUpload(File file)`, `ftpDownload(File file)`. Jejich význam je patrný z názvu metod. Metody pro přenos souborů navíc vrhají výjimku `IOException`. Třída má jako svůj atribut `FMFileSystem` (viz 4.3 Balíček fileManager) a `FMFtpConnection`. Dále obsahuje také dvě abstraktní metody. Ty jsou překryty abstraktními metodami atributu `FMFileSystem` a slouží jako informace o tom, jaké množství dat již bylo přeneseno a jak je velký aktuálně přenášený soubor.

4.7 Balíček help

Balíček obsahuje pouze jednu třídu zděděnou od třídy `JDialog`. Její název je `FMAboutForm`. Význam tohoto dialogového okna je pouze informativní a poskytuje informace o aplikaci.

4.8 Balíček images

Balíček jako jediný neobsahuje žádné třídy. Jeho funkce je pouze shromažďovat ikony, které aplikace používá.

4.9 Balíček main

`Main` – spouštěcí třída aplikace. Obsahuje pouze jednu statickou metodu `main(String[] args)`, která vytvoří hlavní okno celé aplikace. Tato metoda také pomocí objektu třídy `LookAndFeelInfo` změní vzhled celé aplikace ze standardního vzhledu Javy na *Nimbus*. Zde lze navolit i jiné vzhledy, např. *Windows*.

`FMMainForm` – jedná se o nejrozsáhlejší třídu. Je to hlavní okno aplikace. Obsahuje pouze jeden atribut třídy `FMSettings` pro uchování aktuálního nastavení barev a viditelnost ikon nástrojové lišty. Po vytvoření tohoto hlavního okna konstruktor zavolá několik inicializačních metod:

- `initSettings()`
- `initTables()`
- `initDisks()`
- `initToolBar()`
- `setListeners()`
- `centerWindow()`

Metody inicializují nastavení okna. Načte se binární soubor *config.dat*, ve kterém je uložena struktura nastavení formou třídy `FMSettings`. V ní je uloženo barevné rozpoložení aplikace a také viditelnost ikon nástrojové lišty. Pokud soubor neexistuje, je zachycena výjimka `FileNotFound` a nastavení se uvede do defaultního. Dále se nastaví dvě třídy FMTable, které v této aplikaci vystupují jako hlavní panely (formou tabulek). Vytvoří se jejich záhlaví a definují se jejich vlastnosti. Metoda `initDisks()` nastaví oběma comboBoxům modely (viz 4.2 Balíček disks) a naplní je tak prvky, reprezentující paměťová média. Nastaví se také tzv. render, který prvky comboBoxu vykreslí. Také se otevře kořenový adresář prvního paměťového média jako výchozí adresář pro oba panely. Další metoda `initToolBar()` nastaví viditelnost ikon nástrojové lišty podle uživatelem definovaných podmínek. Toto nastavení se načte z atributu třídy `FMSettings`. Metoda `setListeners()` nastaví posluchače všech ovládacích prvků formuláře. Další velké množství metod zajišťuje chování ovládacích prvků, přiděluje klávesové zkratky, mění

focus komponent, vytváří nová okna a celkově se stará o chod aplikace (viz zdrojové kódy v příloze práce).

4.10 Balíček others

Balíček reprezentuje třídy a jedno rozhraní, které se využívají okrajově, ale v několika místech.

`FMDate` – třída nesoucí poslední časovou a datovou změnu souborů. Obsahuje překrytou metodu `toString()`, vracející tuto změnu v patřičném formátu.

`FMReplacer` – třída využívaná k hromadnému přejmenování souborů (viz 354.4 Balíček fileSummary). Její parametrická metoda `run(String origName, int index)` spustí sekvenci privátních metod, které z originálního jména souboru vytvoří nové jméno. Vstupní data jsou zadaná pomocí konstruktoru této třídy.

`FMWindowManager` – je definovaná třída, která má pouze jednu veřejnou metodu `centerWindow()`, určenou k centrování modálních i nemodálních dialogových oken.

`IFMTaskDoneListener` – není třída, ale rozhraní (proto název začíná „I“ – od slova „*Interface*“). Využívá jej mnoho tříd aplikace a slouží k rozpoznání stavu, kdy nějaká úloha dosáhla svého konce. Většinou určuje, kdy dokončilo vlákno nějaké třídy svůj běh. Tato třída má jednu deklarovanou metodu `done()`.

4.11 Balíček panels

Tento balíček interpretuje hlavní panely aplikace, které umožní procházet adresářovou strukturu a spouštět soubory.

`FMColumnModel` – dědí od třídy `DefaultTableColumnModel`. Obsahuje pouze jednu překrytou metodu `getColumn(int columnIndex)`, vracející referenci na objekt třídy `TableColumn`.

`FMTable` – tato třída je potomkem `JTable` třídy. Fyzicky v aplikaci zastupuje tabulky, které jsou využity jako hlavní panely aplikace pro procházení adresářové struktury paměťových médií. Obsahuje všechny metody svého předka + větší množství metod pro práci s barvami panelu, metody sloužící k označování souborů, znovu načtení (refresh) adresářové struktury, rolování panelu (metody vyvolané klávesovými zkratkami *HOME* a *END* – viz 3.8.1 Klávesové zkratky) a nastavování automatické šířky sloupců.

`FMTableCellEditor` – je potomek třídy `AbstractCellEditor` a třída implementující rozhraní `TableCellEditor`. Úkolem třídy je povolit přejmenování (editaci) názvu souborů pomocí pravého tlačítka myši. Stará se také o to, aby při stisku klávesy *ESC* byla editace ukončena.

FMTableColors – třída uchovávající veškeré barevné nastavení panelů aplikace. Objekt dle předpisu této třídy se ukládá do konfiguračního souboru *config.dat*, který zajistí možnost opětovného načtení barev při znovuspuštění aplikace.

FMTableModel – představuje model pro třídu `FMTable`. Předek této třídy je `AbstractTableModel`. Je to třída, která udržuje reference na soubory a adresáře, čili objekty třídy File. Obsahuje datové typy sloupců v hlavních panelech, údaje o (ne)možnosti editace některého ze sloupců a také text záhlaví panelu („*Název*," „*Velikost*," „*Datum a čas*"). Umožňuje přejmenování souborů a také jej zajišťuje. Nastavuje informace o souborech ve sloupcích hlavního panelu.

FMTableRenderer – třída obsahující pouze jednu novou veřejnou metodu a všechny ostatní metody svého předka `DefaultTableCellRenderer`. Také dvě privátní metody pro získání systémové ikony velikosti 16x16 px a nastavení barev hlavního panelu. Pokud systémová ikona není nalezena (např. z důvodů odstranění souborů jinou aplikací před aktualizací okna), je souboru přidělena ikona červeného otazníku. Třída také vykresluje veškeré položky objektu třídy `FMTable`.

4.12 Balíček rs232

Tento balíček v sobě zahrnuje třídy obsluhující sériovou komunikaci.

FMListSerialPorts – třída, která při své konstrukci prozkoumá všechny sériové porty PC, které jsou v danou chvíli dostupné. Reference na porty si postupně uloží v dynamickém poli interpretovaném jako atribut třídy. K získání sériového portu pak můžete využít metody `getPort(String portName)`, nebo `getSerialPorts()`.

FMOpenPort – třída implementující rozhraní `SerialPortEventListener`. Toto rozhraní třídě umožňuje zachytit datový tok od připojeného zařízení. Třída otevře pomocí parametrů předaných konstruktoru sériový port. Obsahuje také veřejnou metodu ke čtení ze sériového portu nebo metodu k zápisu. `IFMSerialPortListener` je rozhraní, které tato třída využívá k tomu, aby zaslala všem posluchačům přijatá data ze sériového portu. Data interpretuje formou datového typu String.

FMRS232Form – dialogové okno formuláře, který zprostředkovává informace mezi vysíláním i přijímáním dat. Při svém vytvoření spouští nové vlákno vnořené třídy `CheckPortsTask`, které detekuje aktuálně viditelné sériové porty v PC a následně jejich názvy zapíše do comboBoxu COM portů.

4.12.1 Sériová komunikace

V jazyku Java je pro chod sériové komunikace nezbytný soubor *RXTXcomm.jar* obsahující třídy pro komunikaci se sériovým zařízením. Dále je zapotřebí knihovna *rxtxSerial*, která se liší podle operačního systému[30]. Aby zkompilovaný soubor aplikace *FileManager.jar* nepotřeboval soubor v relativním adresáři */lib/RXTXcomm.jar*, bylo nutno přidat do kompilačního souboru *build.xml* následující kód [13]:

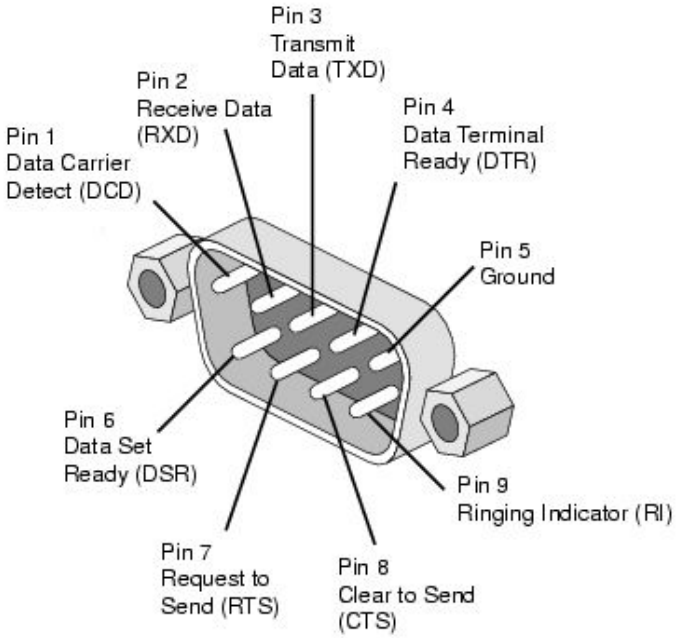
```
<target name="-post-jar">
<jar jarfile="dist/FileManagerComplet.jar">
         <zipfileset src="${dist.jar}" excludes="META-INF/*" />
         <zipfileset src="dist/lib/RXTXcomm.jar" excludes="META-INF/*" />
            <manifest>
                <attribute name="Main-Class" value="main.Main"/>
            </manifest>
      </jar>
</target>
```

Při kompilaci tento kód zajistí vytvoření souboru *FileManagerComplet.jar*, který bude obsahovat výstupní soubor *FileManager.jar* + navíc soubor *RXTXcomm.jar*.

Pro OS MS Windows je nutné přidat ke zkompilovanému souboru knihovnu *rxtxSerial.dll*, nebo ji nakopírovat do adresáře „*C:/Windows/System32/*.“ V operačním sysému Linux je k instalaci *rxtxSerial* možno využít příkazu [14]:

```
sudo apt-get install librxtx-java
```

Aplikaci jsem testoval na verzi RXTX-2.1-7 a neměl jsem s ní žádné problémy.



Obrázek 20 - Standardní konektor pro sériovou komunikaci[31]

[30] Soubory jsou volně dostupné pro všechny OS na http://rxtx.qbang.org/wiki/index.php/Download
[31] Zdroj: Dostupný na WWW http://www.hobbytronics.co.uk/pinout-9-dsub-serial

4.13 Balíček settings

Balíček obsahující třídy, které slouží k uchování nastavení aplikace.

`FMSettings` – třída implementující rozhraní `Serializable`. Toto rozhraní je důležité, pokud chceme danou třídu uložit do souboru jako objekt. V tomto případě se jedná o soubor *config.dat*, který v sobě nese veškerou konfiguraci aplikace. Třída v sobě zapouzdřuje dva atributy, a to `FMTableColors` a `FMToolBarVisible`.

`FMSettingsForm` – formulář pro změny v nastavení aplikace. Při jeho vytvoření přebere nastavení aplikace a nastaví si jej jako výchozí. Po potvrzení formuláře tlačítkem *OK* se mimo jiné zavolá statická metoda `FMFileSystem.saveDataInToFile()`, které je parametricky předán název souboru *config.dat* a také upravený objekt třídy `FMSettings` s novými vlastnostmi.

4.14 Balíček system

Balíček v sobě obsahuje pouze jedinou třídu, a to modální dialogový formulář `FMPropertySysForm`. Jeho charakter je pouze informativní. Díky statické metodě `System.getProperty(String key)` je schopen zobrazit údaje jako jméno uživatele, verze Javy, architekturu a také název operačního systému.

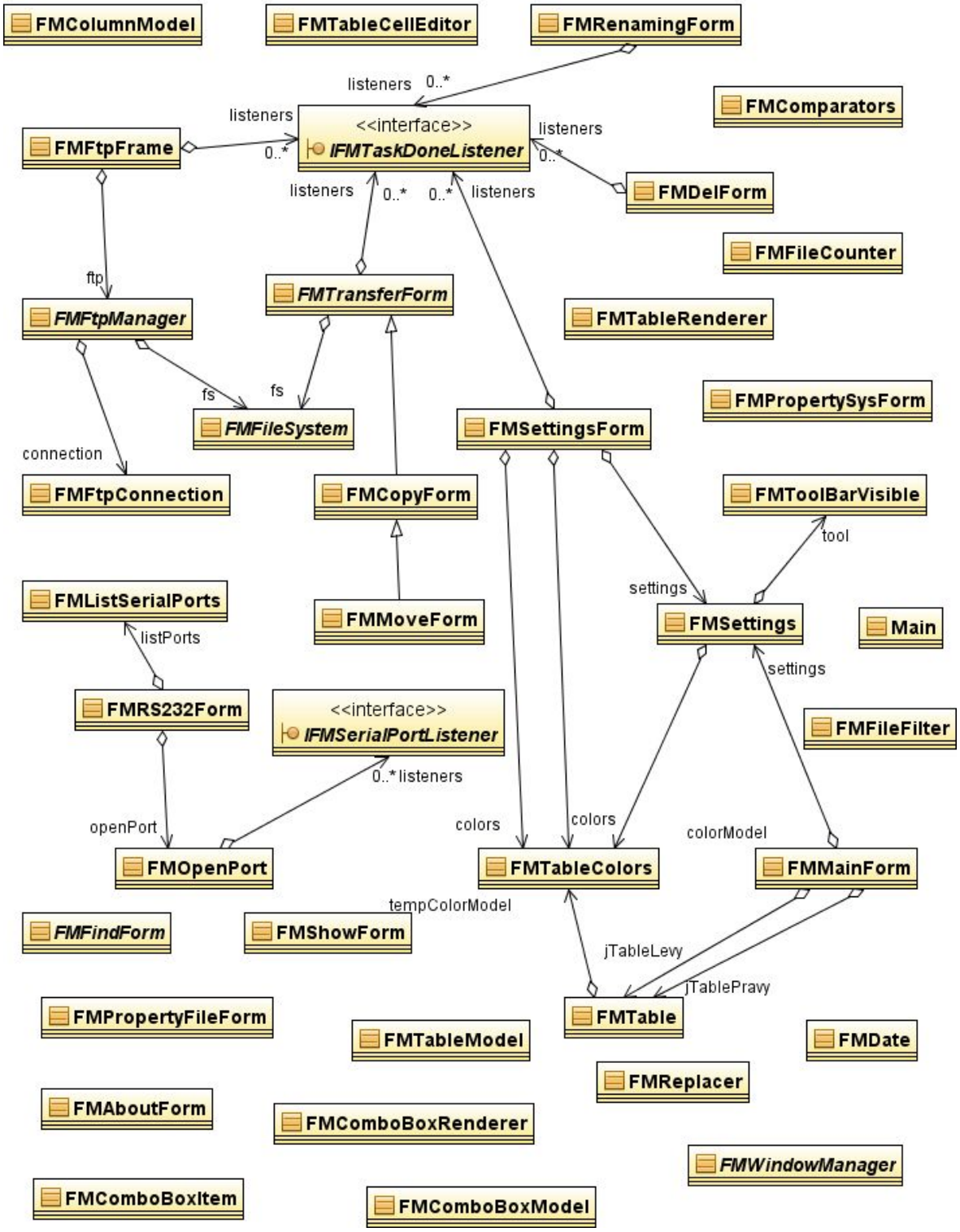
5 Unified Modeling Language (UML)

Pro vizuální vyjádření jsem použil grafický jazyk UML. Tento rozšířený objektově orientovaný jazyk je v dnešní době standardem v oblasti softwarového inženýrství [5]. Jazyk usnadňuje návrh a vizualizaci různých aplikací a je nezbytným prostředkem při tvorbě rozsáhlých systémů. Pod pojmem UML je zahrnuto široké množství diagramů. V této práci jsem použil *Class diagram*.

Diagram tříd, neboli takzvaný *Class diagram*, patří do skupiny diagramů struktur jazyka UML. Používá se k vizualizaci objektově orientovaných systémů. Zobrazuje pohled na třídy objektů, jejich atributy a metody. Znázorňuje také statické vztahy, které mezi nimi existují.

Jak jsem se již zmínil, aplikaci jsem vytvořil v prostředí *NetBeans*. Tento program má schopnost vygenerovat UML *Class diagram* ze zdrojových kódů. Po dlouhém pátrání po nějakém *UML pluginu* jsem zjistil, že nová verze programu *NetBeans* již tuto možnost nepodporuje [6]. Podpora pluginu byla pouze do verze 6.8. Našel jsem sice nabídku různých alternativních programů, ale v mém případě se mi takové řešení zdálo zbytečné a příliš sofistikované. Použil jsem tedy raději starší verzi programu *NetBeans* s podporou pluginu UML.

Obrázek 21 zobrazuje vztahy mezi třídami aplikace. Jsou zde skryty atributy i metody z důvodu přehlednosti vztahů mezi třídami a také rozsáhlému množství informací.



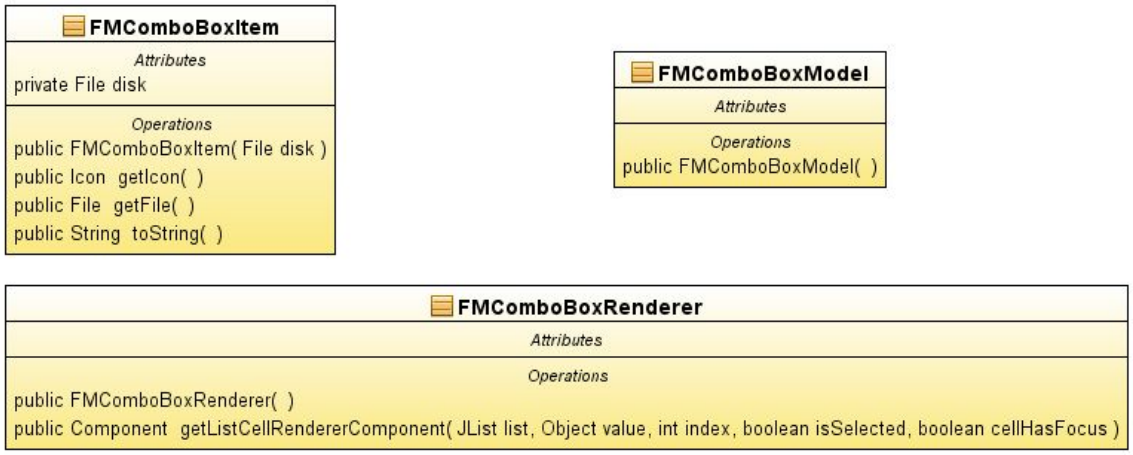
Obrázek 21 - Diagram zobrazující třídy aplikace a vazby mezi nimi[32]

[32] Zdroj: Vlastní

Protože celkový UML *Class diagram* včetně atributů a metod celé aplikace zabere velké množství místa, rozhodl jsem se jej rozdělit na vztahy pouze uvnitř balíčků. Následující část práce zobrazuje UML *Class diagramy* balíčků.

Balíček disks

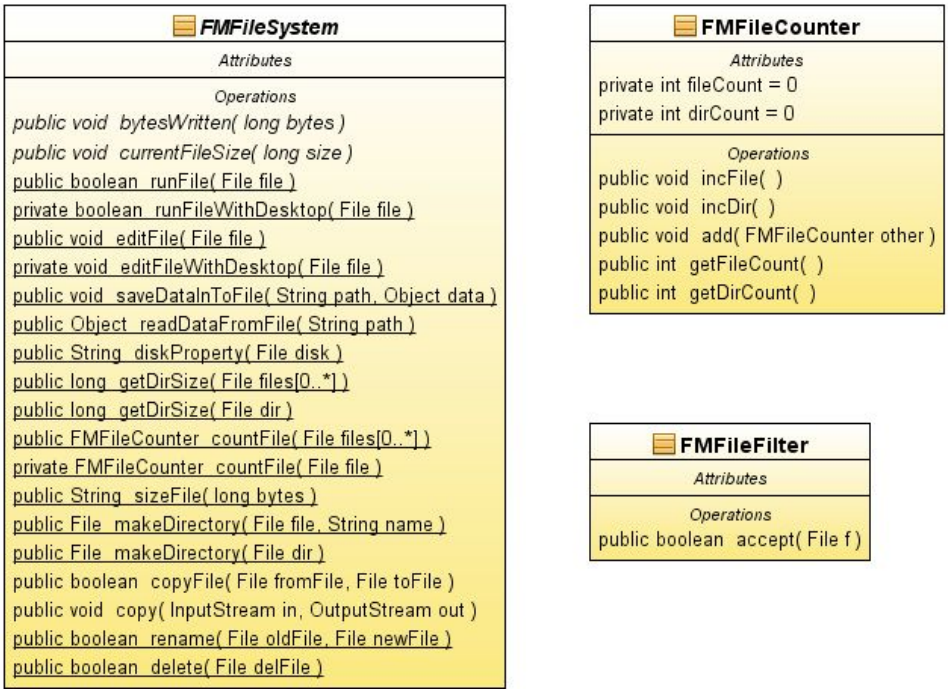
Následující diagram balíčku *disks* zobrazuje třídy a vazby mezi nimi.



Obrázek 22 - Diagram balíčku disks[33]

Balíček fileManager

Následující diagram balíčku *fileManager* zobrazuje třídy a vazby mezi nimi.

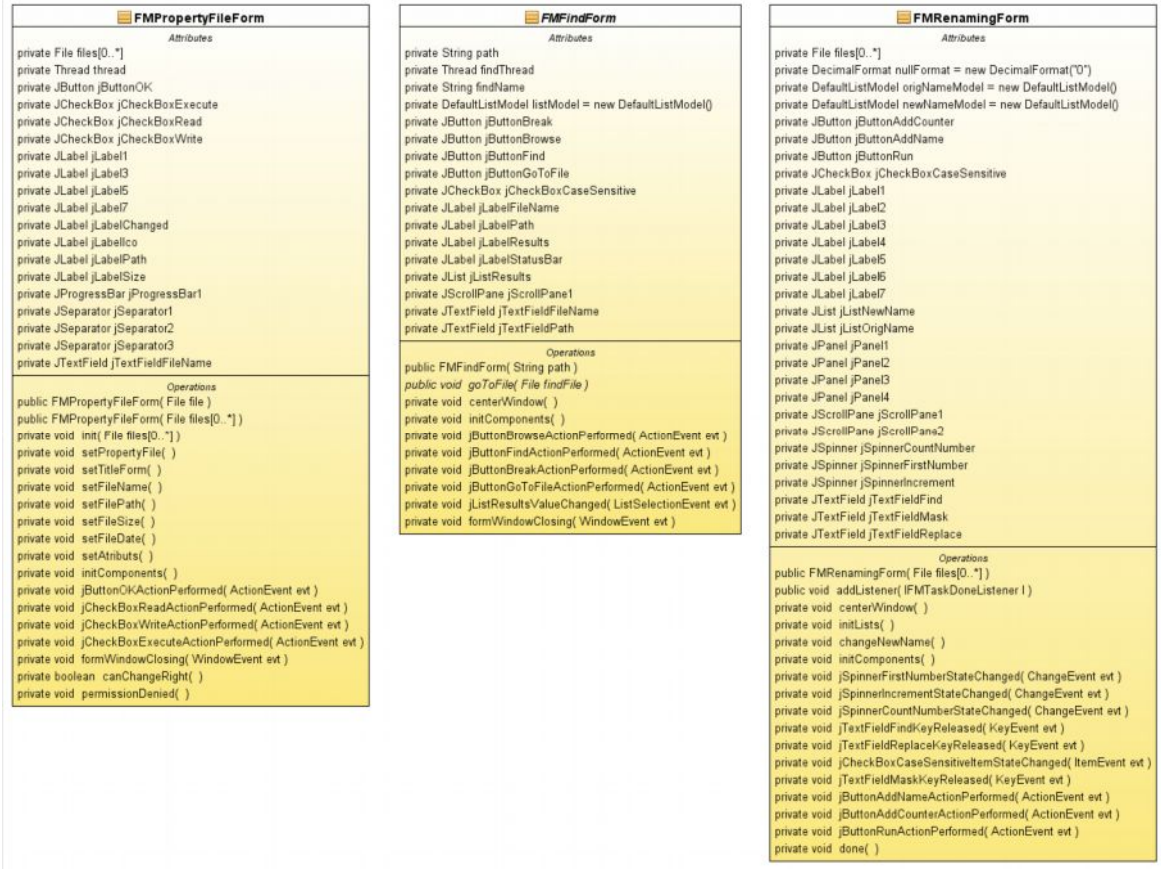


Obrázek 23 - Diagram balíčku fileMnager[34]

[33] Zdroj: Vlastní

Balíček fileSummary

Následující diagram balíčku *fileSummary* zobrazuje třídy a vazby mezi nimi.



Obrázek 24 - Diagram balíčku FileSummary[35]

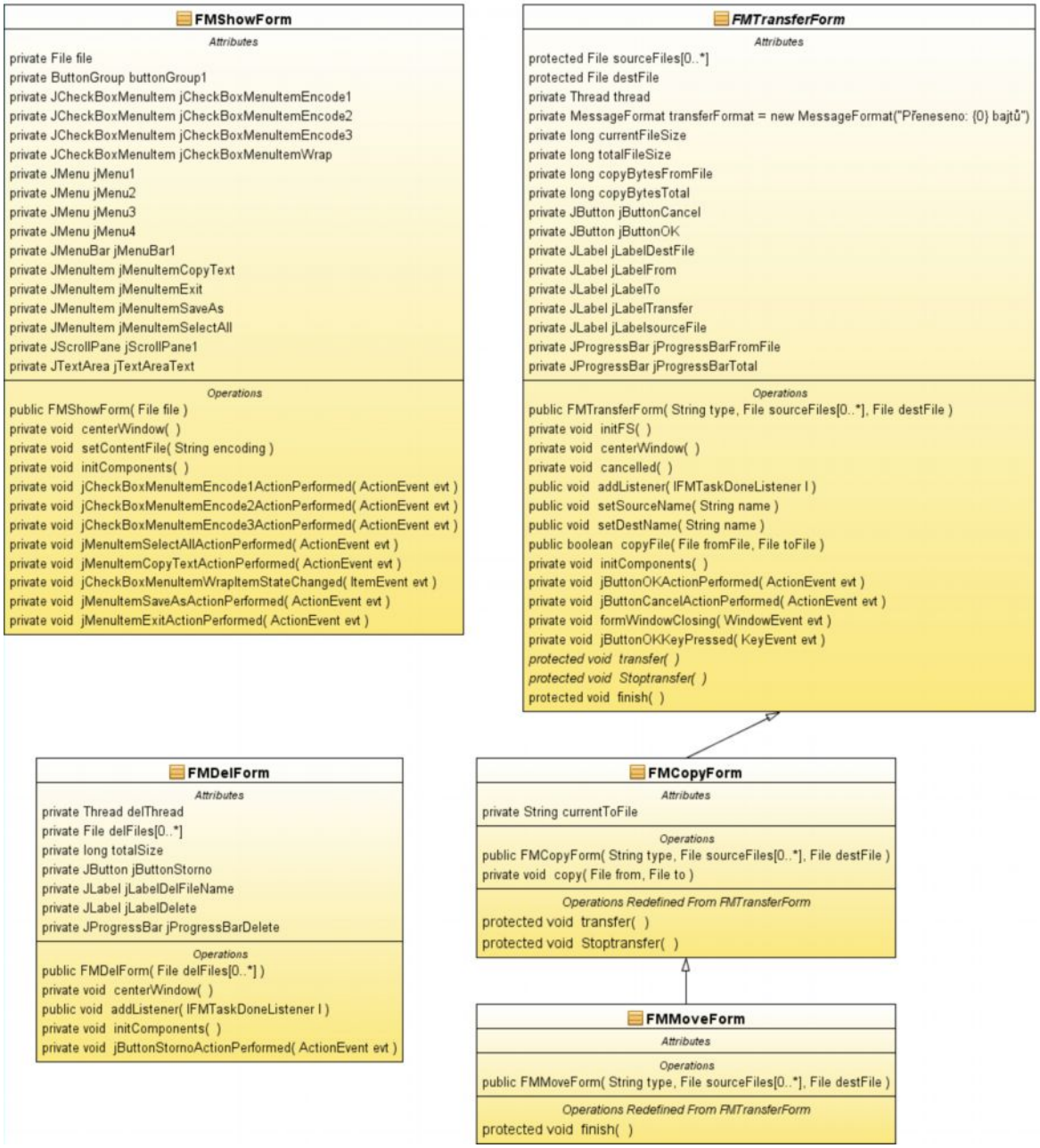
Balíček image

Tento balíček neobsahuje třídy, a proto zde není uveden diagram. Balíček obsahuje pouze ikony použité v aplikaci.

[34] Zdroj: Vlastní
[35] Zdroj: Vlastní

Balíček fileTransfer

Následující diagram balíčku *fileTransfer* zobrazuje třídy a vazby mezi nimi.

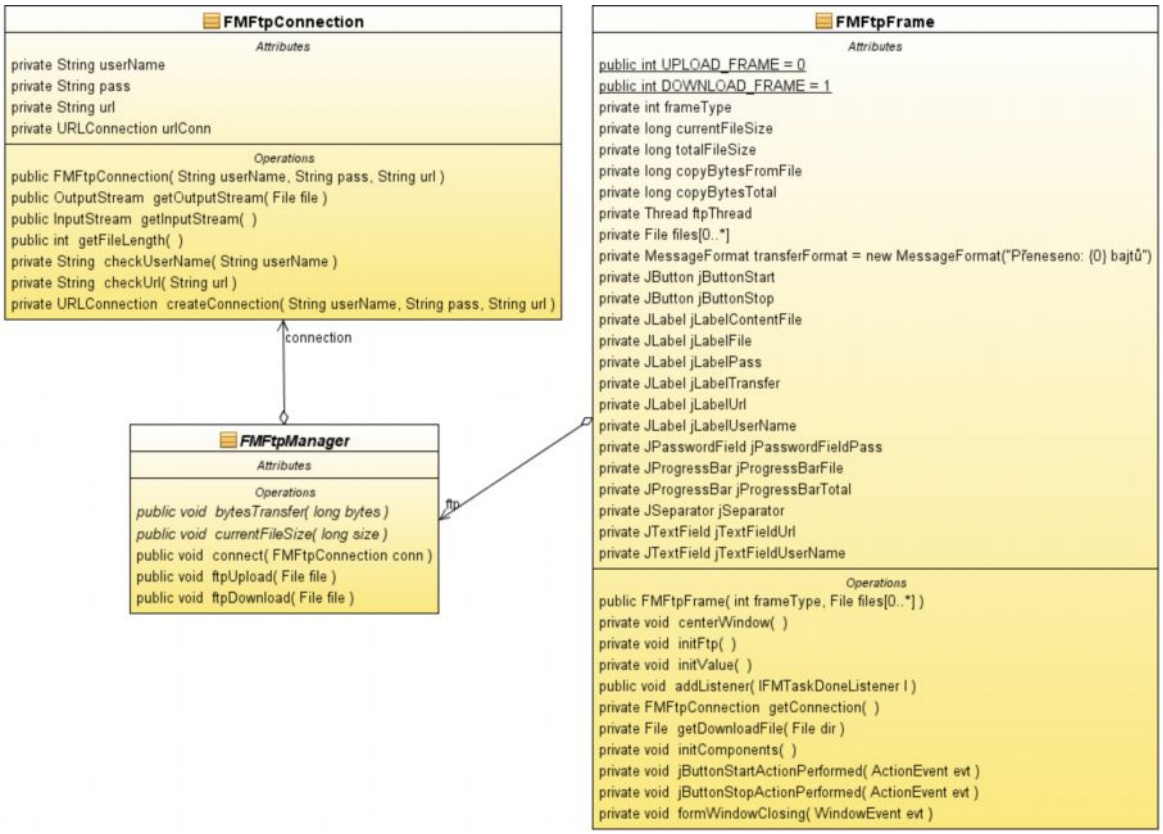


Obrázek 25 - Diagram balíčku fileTransfer[36]

[36] Zdroj: Vlastní

Balíček ftp

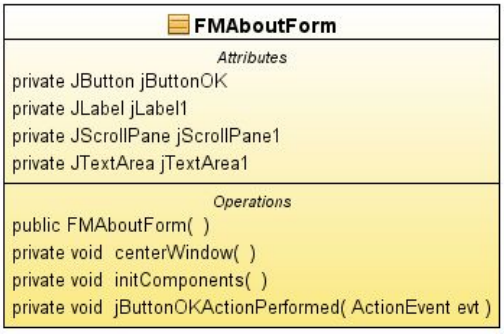
Následující diagram balíčku *ftp* zobrazuje třídy a vazby mezi nimi.



Obrázek 26 - Diagram balíčku ftp[37]

Balíček help

Následující diagram balíčku *help* zobrazuje třídu *FMAboutForm*.

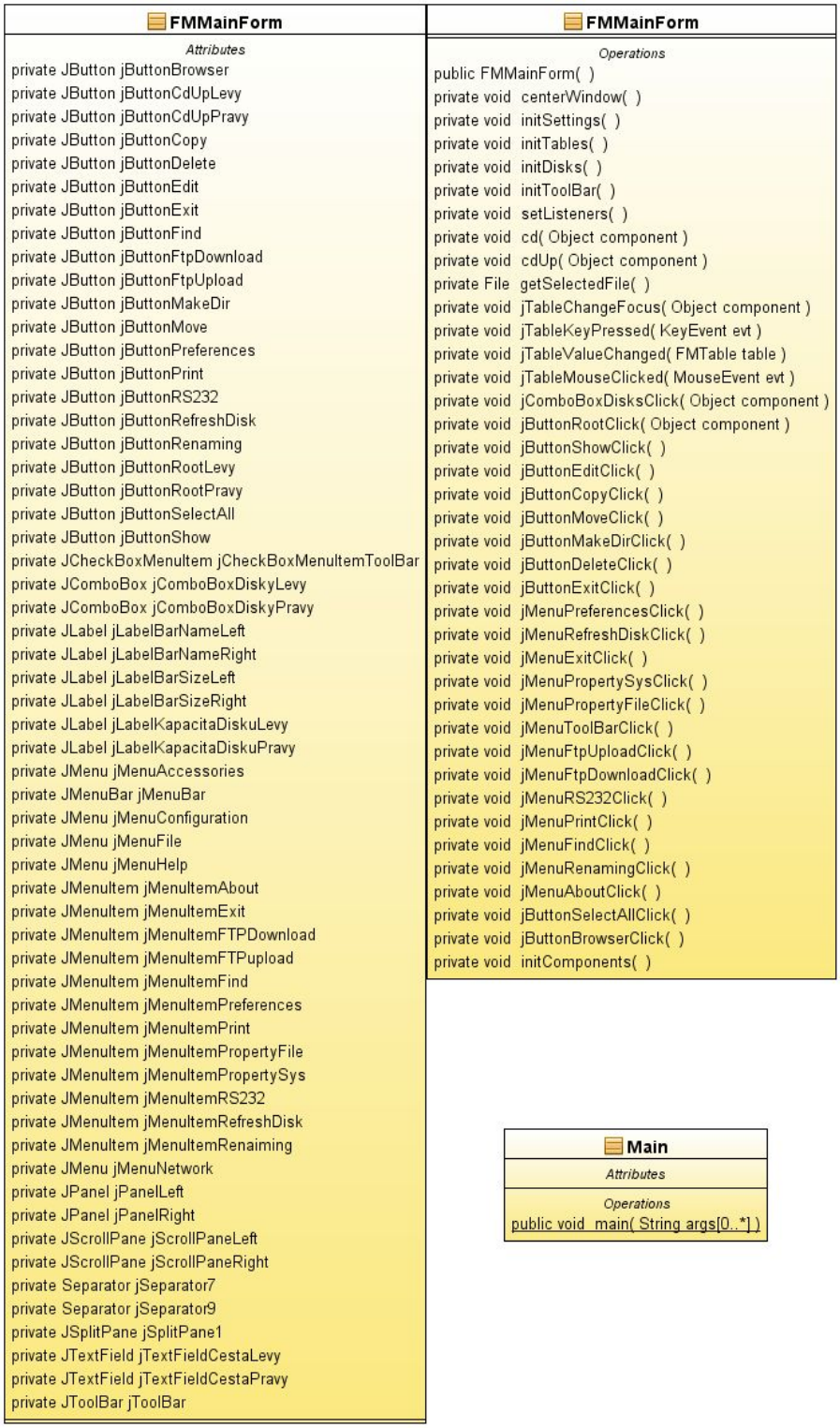


Obrázek 27 - Diagram balíčku help[38]

[37] Zdroj: Vlastní
[38] Zdroj: Vlastní

Balíček main

Následující diagram balíčku *main* zobrazuje třídy a vazby mezi nimi.

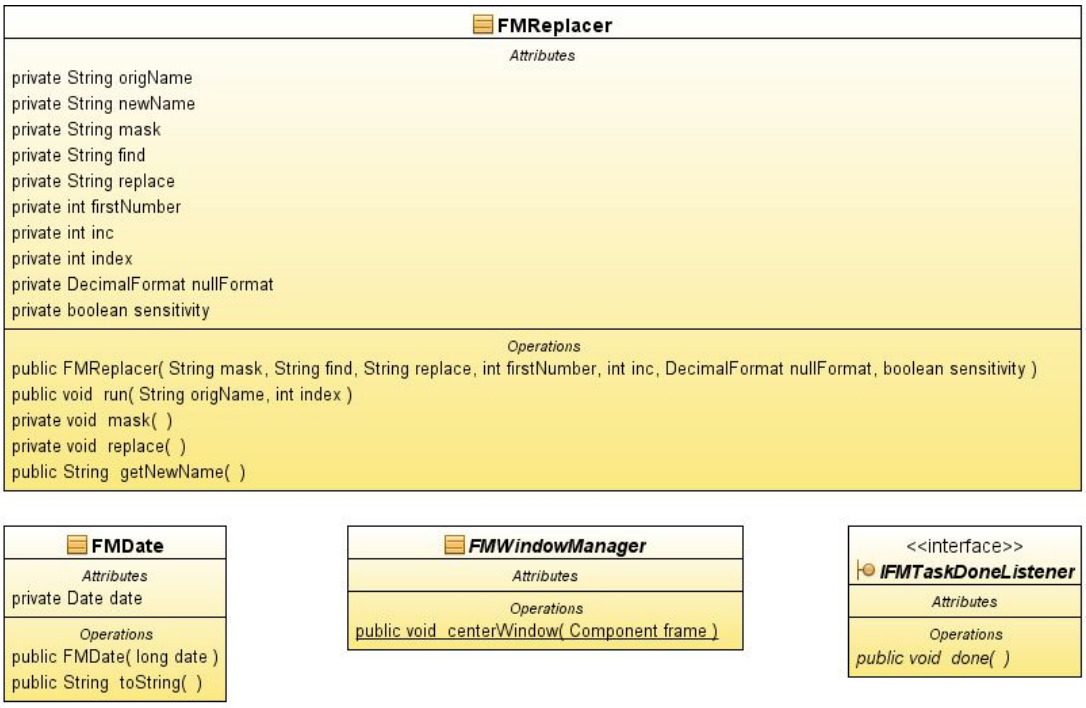


Obrázek 28 - Diagram balíčku main[39]

[39] Zdroj: Vlastní

Balíček others

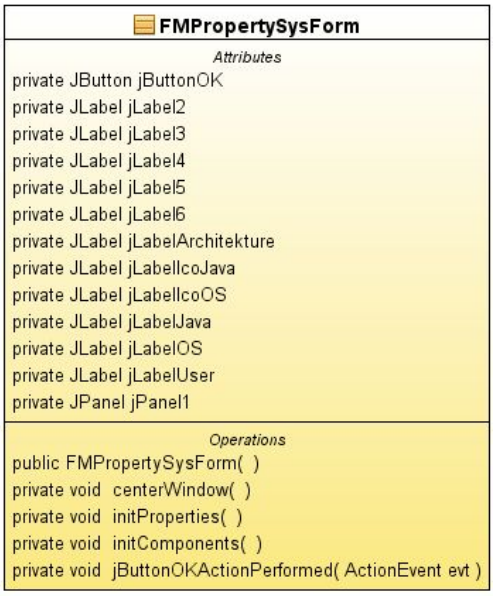
Následující diagram balíčku *others* zobrazuje třídy a vazby mezi nimi.



Obrázek 29 - Diagram balíčku others[40]

Balíček system

Následující diagram balíčku *system* zobrazuje třídu FMPropertySysForm.

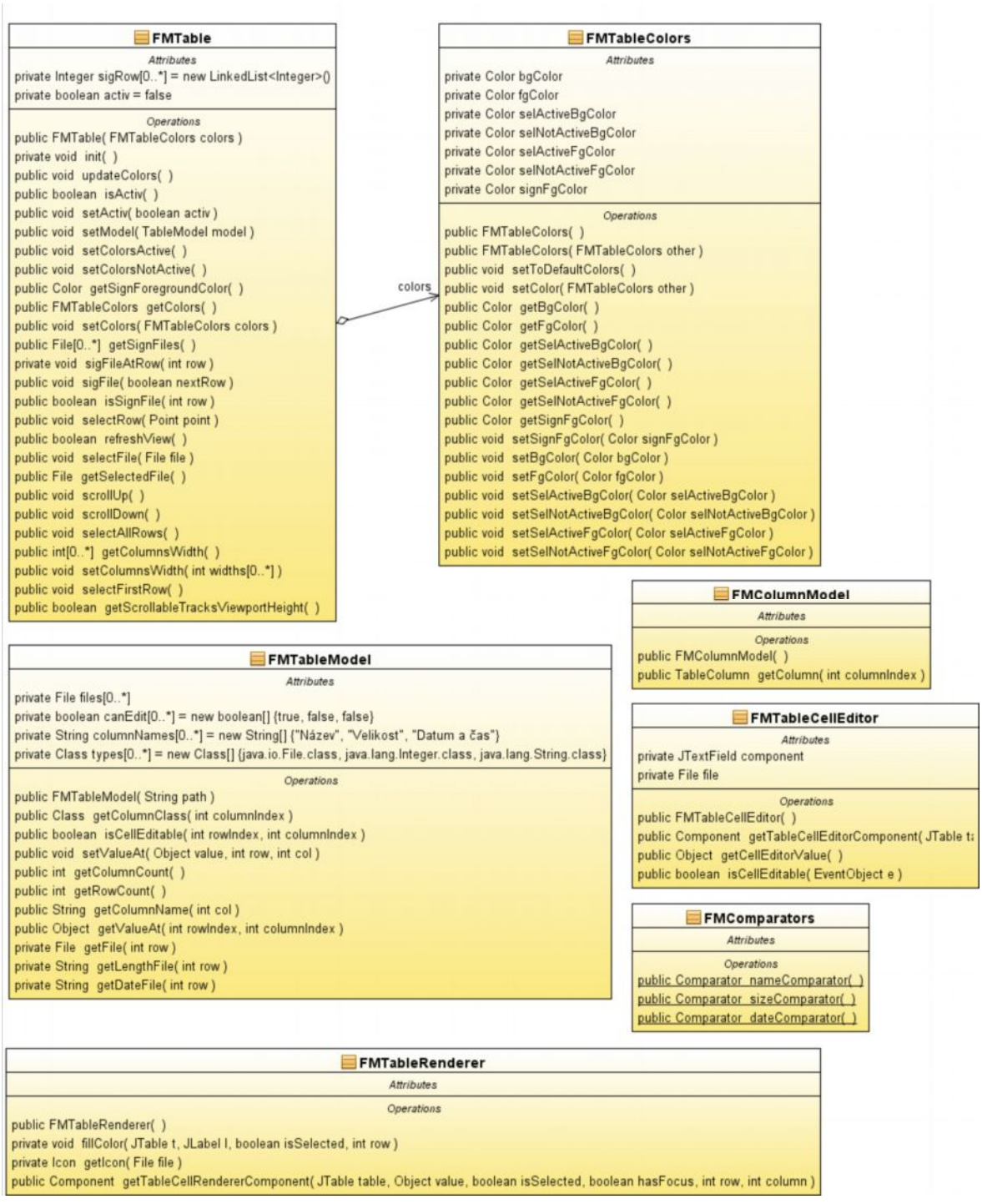


Obrázek 30 - Diagram balíčku system[41]

[40] Zdroj: Vlastní

Balíček panels

Následující diagram balíčku *panels* zobrazuje třídy a vazby mezi nimi.

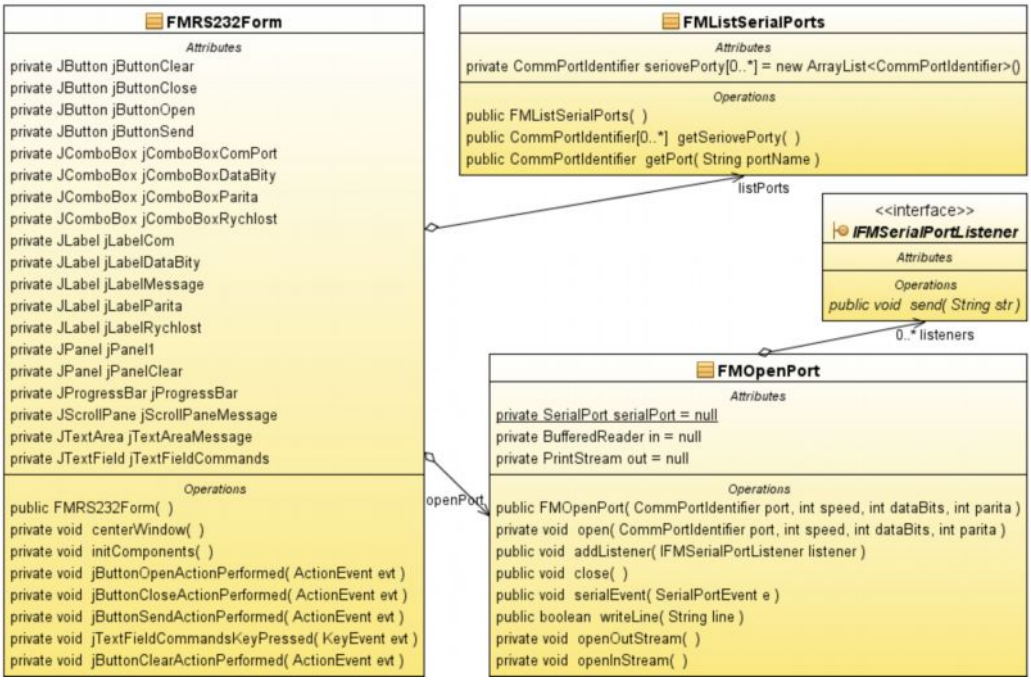


Obrázek 31 - Diagram balíčku panels[42]

[41] Zdroj: Vlastní
[42] Zdroj: Vlastní

Balíček rs-232

Následující diagram balíčku *rs-232* zobrazuje třídy a vazby mezi nimi.

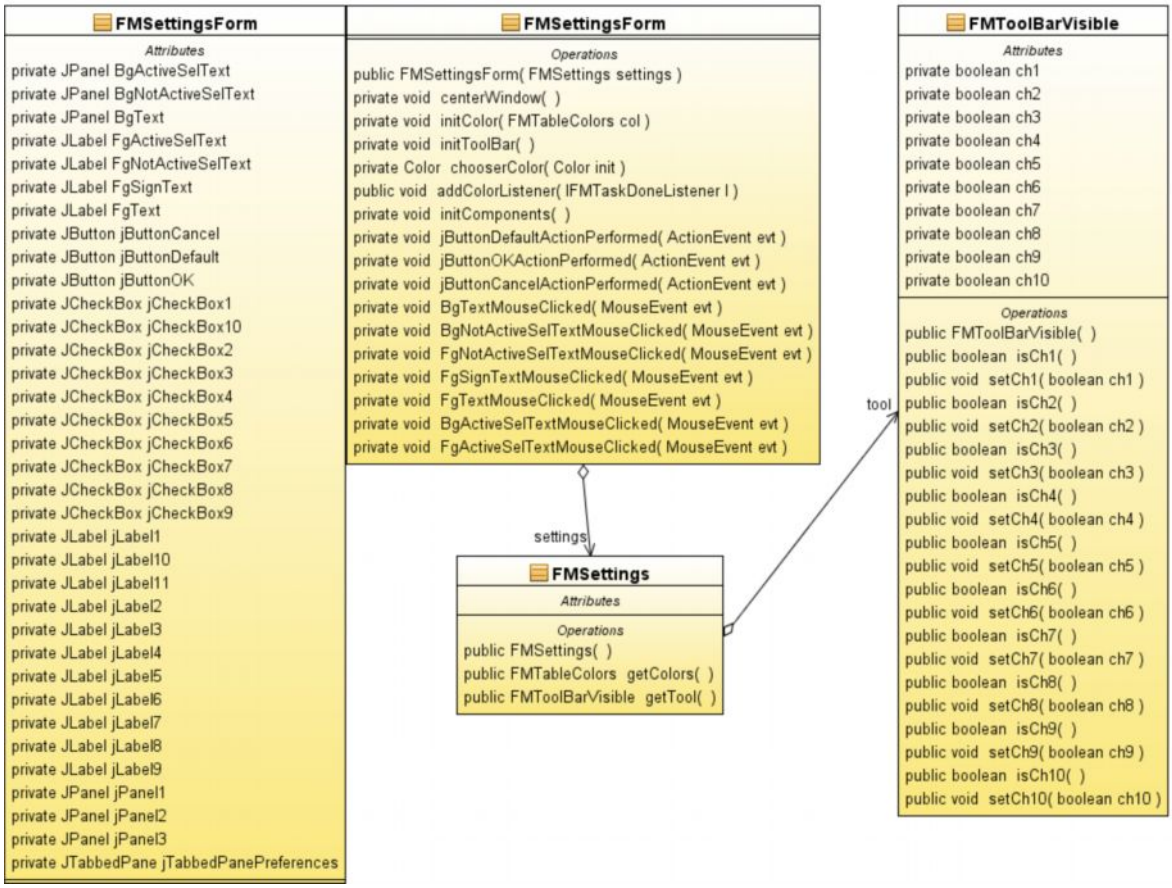


Obrázek 32 - Diagram balíčku rs-232[43]

[43] Zdroj: Vlastní

Balíček settings

Následující diagram balíčku *settings* zobrazuje třídy a vazby mezi nimi.



Obrázek 33 - Diagram balíčku settings[44]

[44] Zdroj: Vlastní

6 Testování aplikace a problémy při vývoji

V průběhu a po dokončení vývoje bylo potřeba aplikaci také testovat. Testování je důležitou součástí vývoje každé aplikace. Aplikaci jsem testoval v několika fázích a na dvou operačních systémech. Jednalo se o:

- MS Windows 7 Professional
- Linux Ubuntu – Karmic koala 9.10

Při testování jsem zjišťoval, zda aplikace správně reaguje na podněty uživatele a v žádném kroku neskončí chybou. Při spuštění zkompilovaného jar souboru již nebylo vidět příkazový řádek s možným výpisem chyb, jakož tomu bylo v prostředí NetBeans. Zjistil jsem ale, že pokud jar soubor spustím z příkazového řádku (v OS Linux z konzole), tak o tuto vlastnost nepřijdu.

Při testování jsem se setkal s několika problémy s rozdílným chováním aplikace v různých operačních systémech. Jedním z nich bylo např. spuštění souboru v programu, který je v operačním systému přidružen příponě spouštěného souboru. Statická metoda `Runtime.getRuntime().exec("rundll32 SHELL32.DLL" + filePath)` totiž v OS Ubuntu nefungovala. Nakonec jsem tento problém vyřešil třídou `Desktop`, která umožňuje spustit aplikaci nezávisle na architektuře.

Jako jeden z dalších problému mě zarazil *focus* panelů (potomků třídy *JTable*). Aktivní panel jsem zjišťoval podle metody `hasFocus`, která vracela datový typ `boolean`. Vše fungovalo v pořádku do té doby, než hlavní okno aplikace vytvořilo jakékoliv nové okno. Po zavření nového okna mi metoda `hasFocus` vracela `false` u obou hlavních panelů aplikace. Na pravý důvod této chyby se mi nepodařilo přijít, tato metoda je standardem Javy. Problém jsem vyřešil tak, že jsem panelům přidal atribut `isActiv`, který měním sám a kontrolovaně.

Když jsem aplikací otevřel adresářovou struktury USB disku, vše se jevilo v pořádku do doby, než jsem disk odpojil za chodu aplikace. Problém tvořilo tzv. rendrování ikon souborů. Aplikace totiž měla referenci na soubor, ke kterému již v danou chvíli neexistovala cesta a snažila se z této cesty načíst ikonu souboru. Problém jsem vyřešil tak, že aplikace v takovém případě zobrazí ikonu červeného otazníku.

Závěr

Tato práce měla za cíl návrh a následné vytvoření aplikace pro jednoduchou správu souborů. Pro vývoj aplikace byl použit jazyk Java a vývojové prostředí NetBeans IDE 6.9.1. Aplikace je přenositelná na platformy Windows a Unix.

Teoretická část práce seznámila čtenáře s jazykem Java, jeho výhodami i nevýhodami. Dále popsala možnosti aplikace a její stále se rozšiřující funkce.

V programátorské části je uveden popis, jakým způsobem aplikace vlastně funguje. Nebylo zde opomenuto ani sériového rozhraní, neboť aplikace využívá komponenty pro sériovou komunikaci. Po vytvoření aplikace došlo k následnému testování, kde se podařilo odstranit známé chyby.

Výsledná aplikace je plně funkční a splňuje všechny cíle ze zadání práce. Je ale nepravděpodobné, že by se stala náhradou za některé jiné komerčně úspěšné programy tohoto typu. Před nasazením do běžného provozu by se musel FTP klient rozšířit o některé další funkce jako např. vytváření (odstraňování) souborů a adresářů. Jako velkou výhodu aplikace bych viděl její nezávislost na architektuře.

Práce byla pro mě opravdu velkým přínosem a měl jsem možnost vyzkoušet si Javu na větším projektu. Seznámil jsem se také s množstvím nových technologií. Jazyk Java je pro mě jako autora velmi zajímavý a nadále bych se rád věnoval jeho učení. Získané zkušenosti bych chtěl rozvíjet jak během dalšího studia, tak i v profesním životě.

Zdroje

[1]Wikipedie: *Java (programovací jazyk)* [online]. Cit. 20. 04. 2011.
Dostupné z WWW:
<http://cs.wikipedia.org/w/index.php?title=Java_(programovac%C3%AD_jazyk)&oldid=6642566>

[2] Programovací jazyk Java: *Co je to Java?* [online]. Cit. 25.4.2011.
Dostupné z WWW: <http://v1.dione.zcu.cz/java/uvod.html>

[3] Wikipedie: *Java EE* [online]. Cit. 28. 04. 2011.
Dostupné z WWW: < http://cs.wikipedia.org/wiki/Java_EE>

[4] HATINA, Petr. *Linuxsoft* [online]. 9.7.2004 [cit. 2011-04-30]. Programování v jazyku Java (1) - Úvod. Dostupné z WWW: <http://www.linuxsoft.cz/article.php?id_article=244>.

[5] Pavus: *Class diagram - diagram tříd* [online]. Cit. 28.04.2011.
Dostupné z WWW: <http://mpavus.wz.cz/uml/uml-s-class-3-3-1.php>

[6] Java.vse: *Instalace NetBeans 6.x včetně podpory pro Subversion a UML* [online]. Cit. 29.04.2011. Dostupné z WWW: <http://java.vse.cz/Java/NetBeansInstalace>

[7] HEROUT, Pavel. *Java : bohatství knihoven.* 3. vyd. České Budějovice : Kopp, 2008. 251 s. ISBN 978-80-7232-368-5.

[8] HEROUT, Pavel. *Java : grafické uživatelské prostředí a čeština*. Dotisk 2. vyd. České Budějovice : Kopp, 2009. 316 s. ISBN 978-80-7232-328-9.

[9] Eckel, B. *Myslíme v jazyku Java*, Grada Publishing, Praha, 2001.

[10] Oracle. *The Java Tutorials* [online]. Cit. 12.5.2010.
Dostupné z WWW: < http://download.oracle.com/javase/tutorial/>

[11] RXTX: *Two way communcation with the serial port* [online]. Cit. 01.05.2011.
Dostupné z WWW:
<http://rxtx.qbang.org/wiki/index.php/Two_way_communcation_with_the_serial_port>

[12] RXTX: *Discovering comm ports* [online]. Cit. 01.05.2011.
Dostupné z WWW: <http://rxtx.qbang.org/wiki/index.php/Discovering_comm_ports>

[13] Java Forums: *Creating a jar including jar libraries* [online]. Cit. 01.05.2011
Dostupné z WWW: <http://www.java-forums.org/netbeans/12582-netbeans-creating-jar-including-jar-libraries.html>

[14] Arduino Forum: *Arduino and 64Bit Ubuntu* [online]. Cit. 27.04.2011.
Dostupné z WWW:
<http://www.arduino.cc/cgi-bin/yabb2/YaBB.pl?num=1239099636/12>

Příloha A – Instalační příručka

Instalace

Spustitelný soubor aplikace se nachází na přiloženém CD v adresáři *install* s názvem *FileManager.jar*. Pro korektní běh aplikace je důležité mít nainstalované Java prostředí *JRE* a také knihovnu *rxtxSerial* pro váš operační systém.

Prostředí *JRE* je možno bezplatně stáhnou z domovské stránky firmy *Oracle*.[45] Knihovnu *rxtxSerial* je možno bezplatně stáhnout z domovských stránek *RXTX wiki*[46] pro všechny operační systémy.

Pro operační systém MS Windows není třeba instalovat knihovnu *rxtxSerial*, nachází se také v adresáři *install*. V případě operačního systému Linux Ubuntu můžete využít místo instalace příkazu:

```
sudo apt-get install librxtx-java
```

Spuštění

Po úspěšné instalaci všech nutných komponent lze aplikaci spustit spouštěcím souborem *FileManager.jar*. Pro spuštění lze použít také příkazové řádky nebo konzole za pomoci příkazu:

```
{cesta k souboru} /java -jar FileManager.jar
```

[45] http://www.oracle.com/technetwork/java/javase/downloads/index.html
[46] http://rxtx.qbang.org/wiki/index.php/Download

Příloha B – CD obsahující

Přiložené CD obsahuje:

- Zdrojové kódy aplikace.
- Instalační příručku.
- Uživatelskou příručku.
- Spustitelný soubor aplikace.