ČESKÉ VYSOKÉ UČENÍ TECHNICKÉ V PRAZE
Fakulta elektrotechnická
Katedra komunikační techniky

Vylepšení systému pro generování šifrových textů

Zpráva k Individuálnímu projektu

Studijní program: Komunikace, multimédia a elektronika
Studijní obor: Multimediální technika

Vedoucí práce: Ing. Tomáš Vaněk, Ph. D.

Miroslav Smutný

Praha 2016

Obsah

1 Úvod

Mým úkolem v tomto projektu a v případné navazující bakalářské práci bylo (je) upravit aplikaci Cypher. Jedná se o program sloužící ke generování zadání k úlohám týkajících se šifrování a jejich odesílání studentům. Jeho původním autorem je pan Doc. Ing. Zdeněk Kouba, Csc. Program dále upravovali Petra Marešová a Jan Kohout. Touto prací jsem se k nim přidal i já.

Konkrétně mým úkolem bylo:

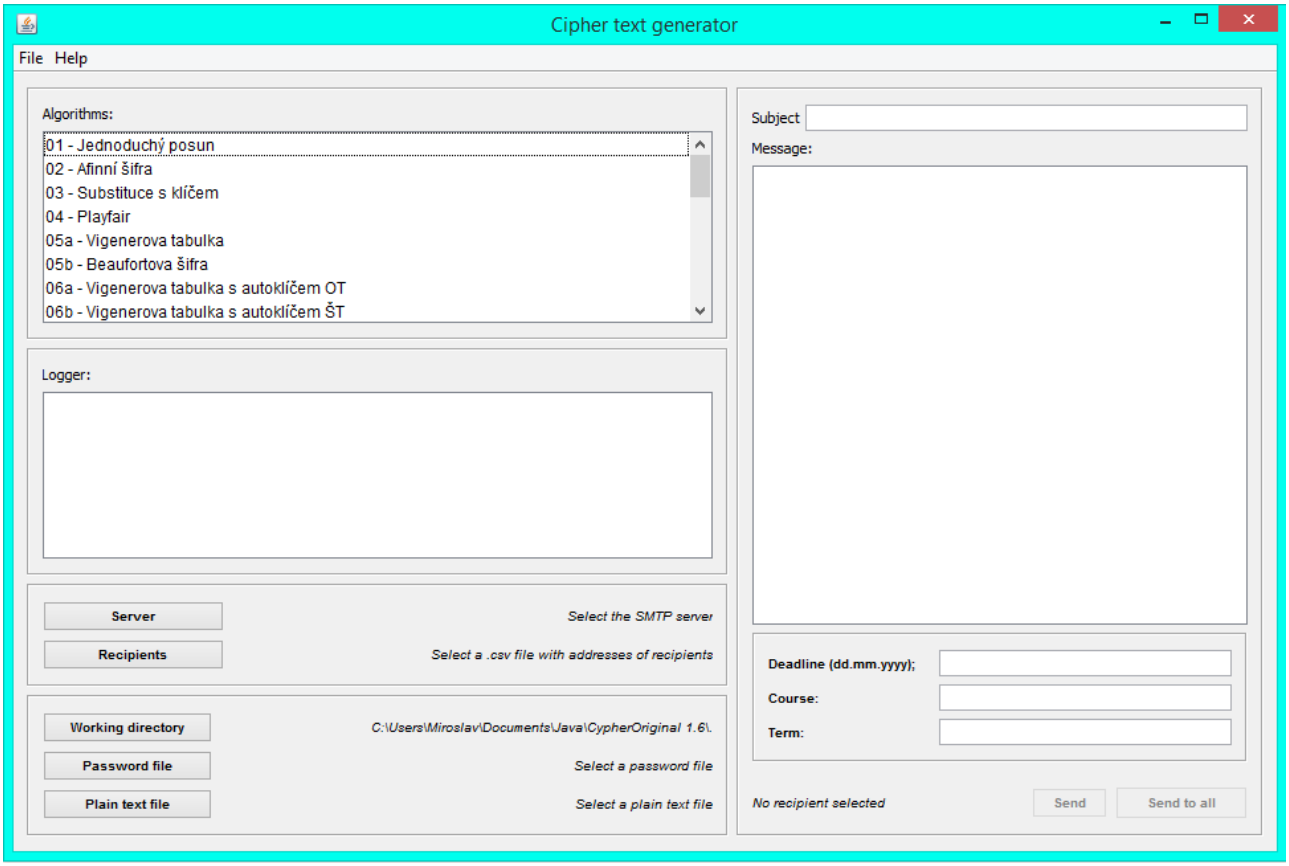
1. seznámit se s aplikací Cypher
2. implementovat export zadání do pdf
3. další drobná vylepšení

Text této zprávy tvoří vedle úvodu, který právě čtete, dále část druhá, kde naleznete poznámku k prvnímu spuštění aplikace (a importu do vývojového prostředí NetBeans), pak zpracování jednotlivých úkolů (části tři až pět), návrh na další vylepšení práce (6) a závěr (7). Za závěrem najdete seznam použité literatury (podle [1]).

2 Poznámky k importu zdrojových kódů a prvnímu spuštění aplikace

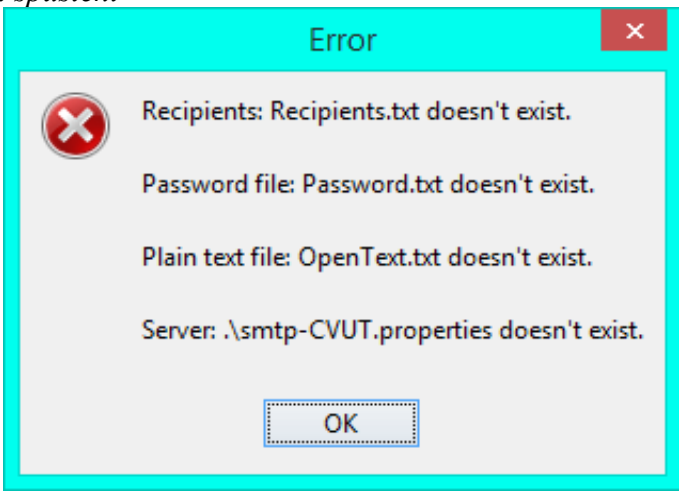
Tvorba programu probíhala ve vývojovém prostředí NetBeans v nejnovější (v době psaní) verzi 8.0.2 [2]. (Verze 8.1 byla sice také dostupná, ale ne jako finální verze, pouze jako release candidate.) Pro práci na projektu jsem použil nejnovější verzi Java Development Kit (JDK), a to konkrétně Java 8 Standard Edition update 66.

Po přidání knihoven appframework, mail, smtp a swingworker do projektu (obsah složky lib nepřidává NetBeans automaticky) bylo možné poprvé spustit aplikaci Cypher.



Obrázek 1: Okno aplikace při prvním spuštění

Protože při prvním spuštění aplikace nebyly v pracovní složce příslušné soubory, zobrazilo se následující dialogové chybové okno. Proto jsem tyto soubory vytvořil a pomocí tlačítek v GUI programu je vložil. Jelikož Cypher si nastavení ukládá do souboru Cypher.properties bylo rovněž možné cesty k těmto souborům přidat manuálně.



Obrázek 2: Chybové okno

Obsah souborů je následující:

Soubor	Obsah souboru
password.txt	passwordheslopasswordheslo
OpenText.txt	Toto je cvičný otevřený text pro účely testování programu Cypher. Toto je cvičný otevřený text pro účely testování programu Cypher. Toto je cvičný otevřený text... Aplikace Cypher požaduje, aby otevřený text měl alespoň 500 znaků, proto se věta opakuje osmkrát.
AdressBook.csv	Mé soukromé a školní adresy (pro účely testování), položky oddělené čárkou, bez hlavičky, ve formátu: jméno,příjmení,paralelka,e-mail.
smtp-SEZNAM.properties	Nastavení smtp serveru k odesílání e-mailových zpráv z mého soukromého e-mailového účtu na Seznamu. Opět nebudu sdělovat podrobnosti k obsahu souboru; je vygenerován automaticky po vyplnění políček v GUI programu Cypher.

Do vývojového prostředí NetBeans bylo dále nutné doinstalovat Swing Application Framework, jelikož byl použit pro tvorbu GUI a podpora pro něj není součástí Netbeans IDE od verze 7.1 [3]. Nicméně je možné použít plugin vyvíjený komunitou uživatelů, který podporu pro tento framework do NetBeans přidá [4]. Nevýhodou tohoto pluginu je to, že poté nejde z NetBeans odinstalovat.

3 Popis aplikace Cypher

Aplikace Cypher je tvořena celkem šesti balíky. Základ aplikace tvoří čtyři balíky cz.cvut.fel.encoding, cz.cvut.fel.cypher, cz.cvut.fel.cyphergui, cz.cvut.fel.conf. Dále tu jsou balíky cz.cvut.fel.cypherhui.resources a cz.cvut.fel.cyphergui.resouces.busyicons. V dalším textu budu část globálně unikátního názvu cz.cvut.fel.* vynechávat a tyto balíky označovat jen encoding, cypher, cyphergui, conf, cyphergui.resources a cyphergui.resources.busyicons. Vedle samotného kódu je k fungování aplikace třeba také několik knihoven. Podívejme se na jednotlivé balíky a knihovny podrobněji. (K získání bližších informací o standardních třídách jsem použil dokumentaci jazyka Java, [5].)

3.1 Balík encoding

V tomto balíku se nácházejí jednotlivé šifry. Každá ze šifer implementuje rozhraní EncodeAlgorithm. Rozhraní EncodeAlgorithm definuje, že každý z algoritmů musí implementovat tyto funkce:

```
String [] getName();
String encode(String plainText, String password);
int getMinTextLength();
int getMaxTextLength();
int getMinPasswordLength();
int getMaxPasswordLength();
```

Seznam algoritmů implementujících toto rozhraní naleznete v sekci 3.1.1.

Dále se zde nachází abstraktní třída SlozenaSifra, která umožňuje sériové spojení dvou šifrovacích metod. Implementuje pouze poslední čtyři funkce rozhraní jako maximum (getMin*) resp. minimum (getMax*) z příslušných hodnot jednotlivých metod. Nicméně téměř žádná ze složených šifer tuto abstraktní třídu nevyužívá (kromě metod 11 až 16 (kombinace s Vigenerovou šifrou)). Možná i proto, že ostatní složené šifry jsou staršího data. Tento nesoulad by možná bylo vhodné odstranit (viz oddíl 6).

Další třídou tohoto balíku je třída Utility obsahující užitečné funkce užívané ve více šifrovacích metodách. Jedná se o generátor náhodných čísel, modulo násobení polynomem, doplnění matice do požadovaných rozměrů, tvorba náhodné permutace a převody znaku na číslo a zpět.

Posledním členem balíku je třída AlgorithmManager, která umožňuje vybrat ze složky (balíky jsou vlastně složky, tečka v názvu balíku je oddělovač podsložek) encode všechny třídy implementující rozhraní EncodeAlgorithm a vytvořit tedy seznam všech dostupných šifrovacích metod, který je poté využívá ve třídě Model k vytvoření HashMap s dostupnými algoritmy.

3.1.1 Seznam algoritmů

Na další stránce je přehledná tabulka všech šifer (kromě složených). Pokud má šifra v tabulce jako minimální i maximální délku hesla uvedenou nulu, znamená to, že heslo nevyžaduje, tedy funkce encode, kterou každá třída implementováním rozhraní EncodeAlgorithm obsahuje, sice přijme jako druhý parametr heslo, ale její algoritmus s ním nepracuje.

Po tabulce následuje princip jednotlivých algoritmů. Pro popis algoritmů jsem užil informace v monografiích [6], [7], kde najdete v případě zájmu více podrobností.

Název	MinTextLength	MaxTextLength	MinPassLength	MaxPassLength
01 – Jednoduchý posun	60	120	0	0
02 – Afinní šifra	80	160	0	0
03 – Substituce s klíčem	150	300	5	15
04 – Playfair	100	200	5	8
05a – Vigenerova tabulka	100	350	5	10
05b – Beaufortova šifra	100	350	5	10
06a – Vigenerova tabulka s autoklíčem OT	200	400	4	10
06b – Vigenerova tabulka s autoklíčem ŠT	200	400	4	10
07 – Transpozice, pevná perioda	40	50	0	0
08 – Sloupcová transpozice	60	300	0	0
09 – Sloupcová transpozice s heslem	100	150	4	10
10 – Dvojnásobná sloupcová transpozice	60	300	0	0

Tabulka 1: Tabulka jednoduchých šifrovacích algoritmů a jejich parametrů

3.1.1.1 Jednoduchý posun

Jedná se o aditivní šifru, kdy šifrovaný text vzniká posunem znaků o několik pozic v abecedě. V této implementaci jsou znaky posunuty o náhodně generované číslo (pro všechny znaky otevřeného textu stejné).

3.1.1.2 Afinní šifra

Afinní šifra je šifra, kterou můžeme zařadit mezi substituční šifry. Každé písmeno se (s použitím modulo aritmetiky) nahradí písmenem vypočítaným pomocí vzorce [7]:

$$y = (ax + b) \bmod 26$$

kde y je písmeno šifrovaného textu, x je písmeno otevřeného textu a a, b jsou celočíselné konstanty.

Aby bylo možné provést zpětné dešifrování (aby zobrazení bylo jednoznačné) je nutné dodržet podmínku, že koeficient a nesmí mít s číslem 26 žádného vyššího dělitele než číslo jedna, tedy

$$D(a, 26) = 1$$

V implementaci v aplikaci Cypher se konstanta a vybírá náhodně z posloupnosti lichých čísel od 3 do 25 kromě čísla třináct (aby byla splněna podmínka zpětného dešifrování) a konstanta b je náhodné (celé) číslo mezi 0 a 25.

3.1.1.3 Substituce s klíčem

Jedná se o substituční šifru, kde šifrovací abeceda začíná (neopakujícími se) písmeny klíče. Za klíčem se abeceda doplní zbylými (nepoužitými) písmeny abecedy.

3.1.1.4 Playfair

Playfair, další substituční šifra, se snaží ztížit možnost prolomení šifry pomocí frekvenční analýzy šifrováním dvojic znaků najednou. K šifrování používá tabulku 5×5 znaků (jeden znak abecedy se tedy substituuje (většinou J pomocí I, nicméně v implementaci v aplikaci Cypher se náhodně volí mezi třemi možnými různými substitucemi: I a J, U a V a V a W)). Můžeme použít heslo, jehož jedinečné znaky vytvoří několik prvních znaků tabulky, zbytek se doplní nepoužitými znaky abecedy.

Při šifrování se otevřený text rozloží do dvojic, pokud by dvojice obsahovala stejná písmena, tak se mezi ně vloží X. X se rovněž vloží na konec, pokud by výsledný počet písmen otevřeného textu byl lichý.

Písmena upraveného otevřeného textu se poté vyhledávají v tabulce. Pokud obě leží ve stejném řádku, každé se nahradí písmenem ležícím o jednu pozici vpravo, pokud leží ve stejném sloupci, nahradí se znakem o jednu pozici dolů. Pokud neplatí ani jedna z předchozích možností, jsou písmena nahrazena písmeny ležícími „na druhé úhlopříčce“, tedy když si dvojici písmen v šifrovací tabulce představíme jako dva vrcholy obdélníku, tak je nahradíme druhými dvěma vrcholy.

3.1.1.5 Vigenerova tabulka

Vigenerova šifra je tzv. polyalfabetická substituční šifra. Písmeno otevřeného textu se nahradí písmenem posunutým o n pozic v abecedě, kde n je pořadí příslušného písmena hesla (číslováno od nuly) v abecedě. Heslo se periodicky opakuje.

Schématicky můžeme šifrování popsat rovnicí (JAVA)

```
cText[i]=(oText[i]+(pass[i%pass.length]-'A'))%26;
```

3.1.1.6 Beaufortova šifra

Jedná se o variantu Vigenerovy šifry, jen písmena hesla nejsou číslována od nuly, ale pozpátku od 26 (tedy jako bychom na heslo aplikovali šifru atbaš).

3.1.1.7 Vigenerova tabulka s autoklíčem

Princip šifrování je totožný s Vigenerovou tabulkou, jen jako klíč (po prvním přiložení dohodnutého klíče místo jeho následného opakování) je použit buď otevřený text (Vigenerova tabulka s autoklíčem OT) nebo šifrovaný text (Vigenerova tabulka s autoklíčem ŠT).

3.1.1.8 Transpozice, pevná perioda

Tento algoritmus spočívá v tom, že otevřený text si přepíšeme do mřížky o pevném počtu řádků (v implementaci se volí náhodně čtyři nebo pět). Pokud otevřený text nestačí k zaplnění celého posledního řádku, doplníme jej znaky X. Poté sloupce mřížky náhodně zpřeházíme a šifrovaný text opět vyčítáme po řádcích.

3.1.1.9 Sloupcová transpozice

Sloupcová transpozice zamíchá písmena otevřeného textu tak, že nejprve jej přepíše do matice po řádcích a vyčítá po sloupcích. Pokud není řádek matice zcela vyplněn, je doplněn znaky X tak, aby všechny sloupce matice byly stejně dlouhé.

3.1.1.10 Sloupcová transpozice s heslem

Tato metoda je podobná obyčejné sloupcové transpozici, jen počet sloupců je dán délkou hesla a pořadí vyčítání jednotlivých sloupců určuje pořadí příslušného písmene hesla v abecedě. Jelikož duplicitní písmena hesla se nevyškrtávají, v případě dvou stejných písmen v heslu se příslušné sloupce vyčítají zleva doprava.

3.1.1.11 Dvojnásobná sloupcová transpozice

Název této metody mluví sám za sebe, na otevřený text se aplikuje sloupcová transpozice a vzniklý text se poté sloupcově transponuje ještě jednou. O velikosti tabulky rozhoduje v implementaci v aplikaci Cypher sám algoritmus tak, aby počet míst, které je nutné vyplnit X byl co nejmenší. Proto je rozměr matice při obou transpozicích stejný.

3.2 Balík cypher

Tento balík tvoří jádro aplikace. Nejdůležitější (a největší) je třída Model, která zajišťuje nebo zastřešuje veškeré činnosti této aplikace – tedy má v sobě uložené veškeré informace týkající se jejího běhu (otevřené soubory s otevřeným textem, hesly, adresami studentů, také získaná data z GUI). Další v pořadí co do velikosti je třída TaskDefinition, která má na starosti veškeré záležitosti konkrétní úlohy (včetně vygenerování samotného výpisu úlohy). Pomyslné třetí místo co do velikosti získala třída CypherSmtpTransport. Ta má na starosti SMTP provoz aplikace. Společně s třídou MailSender se starají o odesílání e-mailů studentům. Neméně důležitou třídou v tomto balíku je třída Controler, která proces šifrování a odesílání zprávy adresátovi ovládá.

Podívejme se nyní na obsah balíku podrobněji.

3.2.1 Třída Model

Třída Model slouží jako taková „studnice informací“ pro celou aplikaci. Při spuštění aplikace CypherGUIView (hlavní okno aplikace, viz dále) při své inicializaci vytvoří instanci třídy Model a poté zavolá metodu initModel, která načte informace z konfiguračního souboru Cypher.properties pomocí instance třídy CypherProperties (viz oddíl 3.4.1) a uloží je do jednotlivých proměných třídy. K tomu slouží velké množství metod pro nastavení a získání hodnoty proměnné, tedy metody set* a get*. O změně některých proměnných (jmen souborů, pracovního adresáře a jménu zrovna vybraného adresáta) informuje třídu CypherGUIView (pomocí metody firePropertyChange – CypherGUIView totiž implementuje PropertyChangeListener).

Při inicializaci třídy Model se rovněž zavolá procedura k inicializaci algoritmů. Tuto inicializaci navrhuji spíše přesunout do třídy AlgorithmManager (viz oddíl 6.1).

Při chodu aplikace je třída Model pomocí metod get* dotazována na jednotlivé informace a při nastavení nových hodnot proměnných v GUI je použito zase metod set*.

Třída Model se stará rovněž o procházení seznamu adresátů pomocí proměnné typu ListIterator<AddressBook.Contact> a metod nextRecipient(), hasNextRecipient() a rewindRecipients().

3.2.2 Třída TaskDefinition

Jak název napovídá, tato třída reprezentuje jednu úlohu vybranou studentovi. Při svém vzniku (v instanci třídy Controler) si z instance třídy Model vybere potřebné informace (ID úlohy (pořadové číslo), vybraného adresáta, jehož úloha se připravuje a seznam algoritmů, které byly studentovi vybrány). Pro každý vybraný algoritmus vytvoří instanci třídy CypherTask (vnitřní třída třídy TaskDefinition, viz dále). O svých aktivitách informuje Logger v GUI (pomocí notifyProgressListeners).

Třída TaskDefinition dále obsahuje funkci

```
getTaskDefinitionProtocol(boolean showPlainText);
```

která vrátí výstupní řetězec, který se odesílá e-mailem, i který se ukládá do textového souboru s řešením. O tom, zda se jedná o text pro studenta nebo pro učitele (i s řešením) rozhoduje parametr showPlainText. Funkce getTaskDefinitionProtocol využívá pro vytvoření jednotlivých částí výstupního řetězce funkce getHeader, getInstructions a getTaskDescription. Poslední funkci volá opakovaně v cyklu pro každou vybranou úlohu. Volání funkce getTaskDefinitionProtocol a uložení textu do souboru řídí třída Controler (viz oddíl 3.2.10).

Vnitřní třída CypherTask reprezentuje jednu úlohu přidělenou studentovi (přidělených úloh může být teoreticky nekonečně mnoho (omezeni jsme jen počtem implementovaných šifrovacích algoritmů)).

Do konstruktoru třídy se předá vybraný šifrovací algoritmus společně s pořadovým číslem úlohy. Podle délky otevřeného textu a hesla požadovaného šifrovacím algoritmem vybere část otevřeného textu a hesla (získaného z instance třídy Model) a zašifruje jej. K získání hesla o požadované délce používá metodu

```
getRandomPassword(int minLenght, int maxLength);
```

ze třídy PasswordBook (viz další oddíl), která, jak název napovídá, vybere náhodně heslo o délce mezi minLenght (sic, opraveno – viz oddíl 5.3) a maxLength.

Ke stejné činnosti u otevřeného textu slouží třída TextParser (oddíl 3.2.6).

Velkou část kódu zabírá práce s řetězením několika algoritmů do složené funkce. Toto zřetězení by bylo lepší přesunout například do třídy SlozenaSifra, jejíž potenciál zůstává v současném stavu aplikace plně nevyužit (více informací v oddíle 6.1). Do ní by bylo vhodné přesunout též obsah třídy CoupoundAlgorithmName.

3.2.3 Třída CompoundAlgorithmName

Třída z pole řetězců s názvy šifer vytvoří řetězec ve tvaru Složený: Alg1 + Alg2.

3.2.4 Třída AddressBook

Tato třída reprezentuje seznam studentů s jejich e-mailovými adresami jako ArrayList instancí vnitřní třídy Contact, tedy jednoho adresáta. Informace načítá z csv souboru, který musí být ve tvaru: Jméno Příjmení Paralelka E-mail. Informace v jednotlivých sloupcích musí být oddělené čárkami.

3.2.5 Třída PasswordBook

Tato třída slouží k vygenerování náhodného hesla z knihy hesel. Kniha hesel je soubor, kde každý řádek je považován za samostatné heslo. Funkce

```
getRandomPassword(int minLenght, int maxLength);
```

vyhledá v knize náhodné heslo, které svojí délkou vyhovuje. Pokud žádné takové ani po tisíci pokusech nenajde, použije poslední nalezené a dokud je kratší než maxLength, tak jej zduplikuje a nakonec ořízne na velikost maxLength.

3.2.6 Třída TextParser

Tato třída slouží k přípravě otevřeného textu. Vybírá z otevřeného textu celé věty (vybranou větu smaže, aby se případně neopakovaly). Konec celé věty nalezne jako nejbližší první výskyt tečky, vykřičníku nebo otazníku v textu. Celé věty skládá za sebe, dokud nezíská dostatečně dlouhý řetězec. Po získání řetězce přeskočí náhodný počet vět (tři až deset), aby pro dvě následující volání po otevřeném textu funkce nevrátila na sebe navazující text, jelikož to by zvyšovalo pravděpodobnost prolomení šifry.

3.2.7 Třída Alphabet

Metodu této třídy ke konverzi písmen využívají obě předchozí třídy, aby převedly hesla a otevřený text obsahující česká písmena s háčky a čárkami na velká písmena bez háčků a čárek. Metoda dále z textu odstraňuje veškerou diakritiku a mezery. Ve výsledném řetězci tedy jsou jen velká písmena anglické abecedy.

3.2.8 Třída MailSender

Tato třída obsahuje vedle metody getInstance(), která buď vrátí existující instanci třídy nebo vytvoří novou, jedinou metodu a to sendMessage. Tato metoda se užívá k vytvoření a následnému odeslání e-mailu studentům. K samotnému odeslání vytvořené zprávy užívá volání metody sendMail ze třídy CypherSMTPTransport (viz následující oddíl).

Díky tomu, že odesílaná zpráva je ve formátu MIME (Multipurpose Internet Mail Extendions), je možné použít text s diakritikou nebo odeslat přílohu. Nicméně přestože odesílání přílohy bylo implementováno, tak nefungovalo tak, jak mělo (přiložený soubor byl poškozen), a pro odesílání pdf přílohy bylo nutné provést drobnou opravu (více viz oddíl 4.2.6).

3.2.9 Třída CypherSmtpTransport

Tato třída zpracovává samotné odesílání e-mailů studentům. Načte parametry SMTP spojení s pomocí třídy SmtpProperties (z balíku conf, viz oddíl 3.4.3) a poté odešle zprávu s použitím funkcí z JavaMail API (viz oddíl 3.7.2).

3.2.10 Třída Controler

Třída Controler řídí tvorbu zadání na nejvyšší úrovni. Jakmile dojde z GUI pokyn k odeslání zadání, zavolá se příslušná metoda (sendToOne nebo sendToAll). Každá z metod nejprve vytvoří spojení pomocí instance třídy CypherSmtpTransport. Poté zavolají proceduru processOneRecipient (sendToAll samozřejmě v cyklu).

Procedura processOneRecipient načte parametry e-mailu z instance třídy Model a poté vytvoří instanci třídy TaskDefinition. Od ní získá řetězec textu úkolu pro studenta a odešle jej. Poté získá text pro učitele a uloží jej do souboru.

Příklad výpisu (e-mail studentovi):

```
PREDMET SEMESTR   Zadání č. xxx

Jméno: xxx xxx (paralelka)   Datum zadání: dnesni datum
 Datum odevzdání: xxxxx

Pokyny:
Text pokynů

Úloha 1:
GBGBWRPIVABGRIRAGRKGCEBRYLGRFGBIACEBTENZHPLCUREGBGBWRPIV
ABGRIRAGRKGCEBRYLGRFGBIACEBTENZHPLCURE
```

Soubor pro učitele (s názvem ve tvaru cislo_zadani-jmeno-studenta-paralelka.txt) obsahuje to samé, jen před šifrovaný text úlohy je přidán otevřený text a parametry použité šifry.

```
(…)

Úloha 1:
TOTOJECVINOTEVENTEXTPROELYTESTOVNPROGRAMUCYPHERTOTOJECVI
NOTEVENTEXTPROELYTESTOVNPROGRAMUCYPHER
(Algoritmus: 01 - Jednoduchý posun [a=13])

GBGBWRPIVABGRIRAGRKGCEBRYLGRFGBIACEBTENZHPLCUREGBGBWRPIV
ABGRIRAGRKGCEBRYLGRFGBIACEBTENZHPLCURE
```

Po dokončení komunikace je spojení po SMTP protokolu ukončeno.

V kódu funkce processOneRecipient byla chyba, která způsobila, že poslednímu studentovi v adresáři nebylo možné odeslat e-mail se zadáním. Po zpracování předposledního totiž aplikace chybně zahlásila, že se jednalo o posledního a převinula adresáty zpět na začátek. Tuto chybu jsem opravil (viz oddíl 5.1).

Třída Controler, stejně jako třída TaskDefinition, o svých aktivitách informuje pole Logger v GUI pomocí funkce notifyProgressListeners.

3.2.11 Třída CypherException

Instance této výjimky je vyhozena při chybě v aplikaci Cypher. Vyhození výjimky je zpravidla ještě doplněno zápisem do logu programu (do textového pole Logger v GUI aplikace).

3.3 Balík cyphergui

Tento balík obsahuje grafické rozhraní aplikace. Vedle toho rovněž obsahuje definici rozhraní ProgressListener, které třídy Controler a TaskDefinition z balíku cypher využívají k zápisu do logu v GUI programu. K tvorbě GUI byl využit Swing Application Framework, jehož podpora byla pozastavena, takže je nutné zastaralou knihovnu v budoucnu vyměnit (viz oddíl 6.2).

3.3.1 Třída CypherGUIApp

Třída obsahující funkci main, jejímž voláním se spouští aplikace. Připraví instanci třídy CypherGUIView a zobrazí ji. Poté z její instance zavolá funkci showErrorLog, která v případě, že se v logu nějaká chybová hlášení nacházejí, zobrazí dialogové okno, jaké se zobrazilo při prvním spuštění aplikace (Obrázek 2).

3.3.2 Třída CypherGUIView

Hlavní okno aplikace (viz Obrázek 1). V levé části okna se nachází (odshora) seznam dostupných algoritmů (jList), které je možné vybrat. S podržením tlačítka Shift lze vybrat algoritmů více. Pod ním se nachází textové pole Loggeru. Zde se objevují zprávy o činnosti tříd TaskDefinition a Controler.

Ve druhém panelu odspodu se nacházejí dvě tlačítka. Jedno pro nastavení a výběr SMTP serveru z nabídky, druhé pro výběr csv souboru s adresáty.

Spodní panel vlevo obsahuje další tři tlačítka. Jedno pro výběr pracovního adresáře (kam budou ukládány soubory s řešením pro učitele), druhé a třetí pro výběr souboru s hesly a s otevřeným textem. Pokud při vybírání souboru dojde k chybě a je vyhozena výjimka CypherException (například třídě Model se nelíbí formátování souboru, nebo žádný soubor nebyl vybrán), zobrazí se chybové dialogové okno.

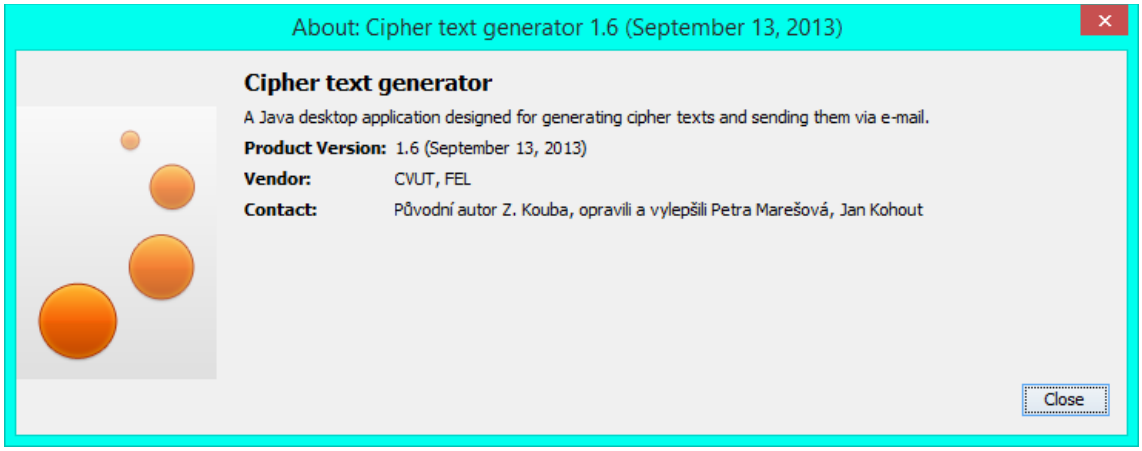
V pravé části okna jsou textová pole pro předmět a pro samotný text e-mailu (pokyny k luštění šifer). Pod nimi jsou další textová pole pro datum odevzdání, kód předmětu a semestr (ZS, LS). Kdykoliv uživatel opustí psaní v některém z textových polí (klikem jinam), dojde k načtení hodnoty z pole a uložení do třídy Model. To se mi zdá zbytečně zdlouhavé, jelikož třída Model informace rovněž zapisuje do textových souborů s nastavením, což může zdržovat. Rychlejší by bylo nechat uživatele vpisovat do polí a vyčíst z nich text až při stisku tlačítka pro odeslání (Send nebo Send to all). Nicméně na druhou stranu uživatel v případě náhlého pádu aplikace o nic nepřijde.

Stisk tlačítka pro odeslání jednomu adresátovi (respektive všem) obsluhují metody sendToOneHandler a sendToAllHandler. Ty zkontrolují, zda všechna potřebná pole jsou vyplněná (pokud je prázdný předmět a/nebo text zprávy, zobrazí se varovné dialogové okno), změní kurzor na čekající (z balíku cyphergui.resouces.busyicons) a zavolají metodu sendToOne respektive sendToAll z třídy Controler (viz oddíl 3.2.10). Poté změní kurzor zpět na původní.

Okno CypherGUIView rovněž naslouchá změnám instance třídy Model (implementuje rozhraní PropertyChangeListener (z balíku java.beans.*)). Kdykoliv se něco v instanci třídy Model změní (a je zavolána metoda firePropertyChange), CypherGUIView na ni vhodně zareaguje. K reakci je určena metoda propertyChange. Podle řetězce propertyName (event.getPropertyName()) se funkce rozhoduje, jaká vlastnost byla změněna. Reakcí většinou bývá změna textu v nějakém prvku GUI (jLabel vpravo od tlačítka) aby odrážel novou skutečnost (jméno vybraného souboru).

Okno rovněž obsahuje ve své horní části dvě menu (jMenu) každé s jednou položkou. Menu File obsahuje položku Exit (vypnutí aplikace) a menu Help položku About (zobrazení informací o programu – instance třídy CypherGUIAboutBox).

3.3.3 Třída CypherGUIAboutBox

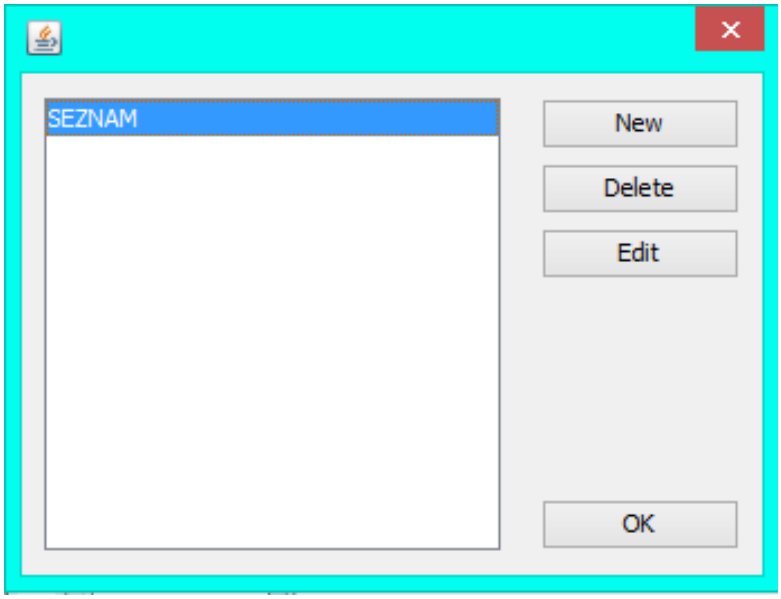


Obrázek 3: Dialogové okno O aplikaci (About)

Okno, které se zobrazí po stisku položky menu About. Text v okně je načítán ze souboru CypherGUIAboutBox.properties ve třídě cyphergui.resouces.

3.3.4 Třída ServerDialog

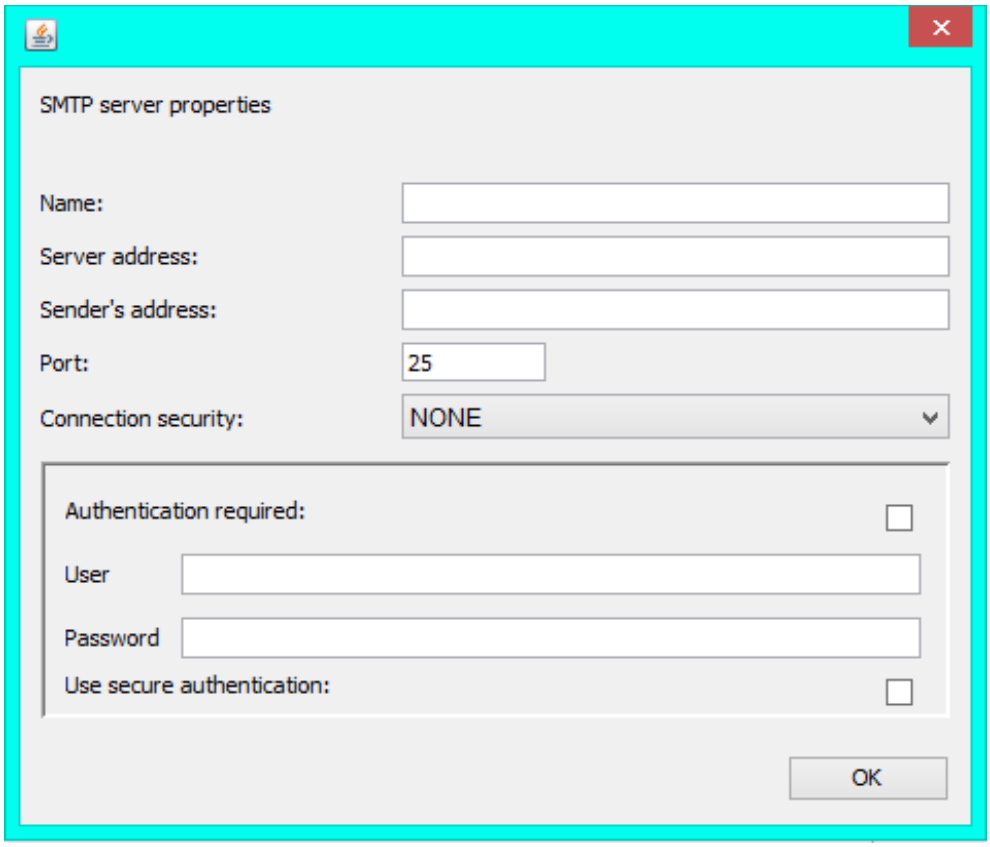
Instance této třídy vzniká již při inicializaci hlavního okna. Při kliknutí na tlačítko Server dojde pouze ke zviditelnění okna jako na obrázku 4.



Obrázek 4: Dialogové okno pro výběr SMTP serveru

V seznamu jsou dříve nastavené servery, které mohou být využity k odesílání e-mailů studentům. Uživatel si jeden ze serverů může vybrat (a výběr potvrdit tlačítkem OK) nebo vytvořit nové spojení, smazat jedno ze stávajících a nebo stávající upravit. Při vytváření nového a editaci stávajícího nastavení serveru se využívá okno PropertiesDialog.

3.3.5 Třída PropertiesDialog

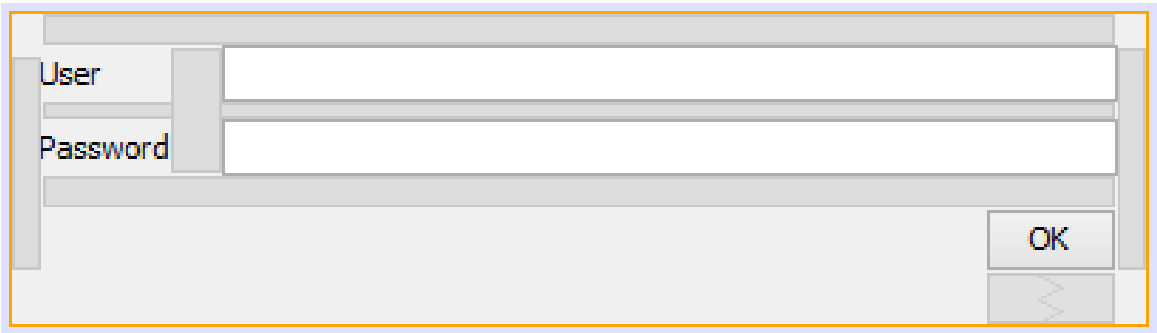


Obrázek 5: Dialogové okno třídy PropertiesDialog

Třída PropertiesDialog slouží k nastavení SMTP serveru pro odesílání e-mailů. Hodnoty v políčkách jsou načítány z instance třídy SmtpProperties (také do ní ukládány) a položky drop-down menu jsou načítány z třídy ConnectionSecurity (obě z balíku conf, viz oddíly 3.4.3 a 3.4.4).

3.3.6 Třída AuthenticationDialog

Tato třída reprezentuje dialogové okno k zadání přihlašovacího jména a hesla, nicméně tato třída není nikde použita.



Obrázek 6: Grafický návrh okna AuthenticationDialog

3.4 Balík conf

Balík obsahuje třídy starající se o zachování nastavení mezi jednotlivými běhy aplikace.

3.4.1 Třída CypherProperties

Při spuštění aplikace se z této třídy načítá nastavení. Třída pracuje tak, že na vyžádání dodá informaci uloženou v souboru cypher.properties. Informace jsou v tomto souboru uložené jako řetězce pod klíči specifikovanými touto třídou.

Jejich seznam:

```
private static final String CP_FILE_NAME =
     "cypher.properties"; //jméno souboru, kam se
                          //vlastnosti ukládají
private static final String CP_DEFAULT_SMTP_CONFIG_DIR
     = "."; //kam se uloží
     //konfigurační soubor SMTP serveru, když
//v souboru CP_FILE_NAME konfigurace není nalezena
public static final String CP_SMTP_CONFIG_DIR =
     "smtp.config.dir";
private static final String CP_SMTP_SELECTED_SERVER =
     "smtp.selected.server";
private static final String CP_WORKING_DIR =
     "cypher.working.directory";
private static final String CP_PLAIN_TEXT_FILE =
     "cypher.plain.text.file";
private static final String CP_RECIPIENTS_FILE =
     "cypher.recipients.file";
private static final String CP_PASSWORD_BOOK_FILE =
     "cypher.password.book.file";
private static final String CP_COURSE = "cypher.course";
private static final String CP_DEADLINE =
     "cypher.deadline";
private static final String CP_TERM = "cypher.term";
private static final String CP_TASK_DEFINITION_ID =
     "cypher.task.definition.id";
```

Čtení a zápis vlastností do souboru zajišťuje instance třídy Properties (z balíku java.util.*). Třída CypherProperties poskytuje ke každé vlastnosti get* a set* metodu.

Vždy, když se v GUI programu nastaví nějaká z těchto vlastností, informace se uloží ve třídě Model a ta poté zavolá příslušnou metodu set* z této třídy, která ji uloží do souboru.

3.4.2 Třída SmtpAccountName

Třída, která má na starosti převod mezi názvem SMTP serveru (nastavení zobrazené v GUI, například podle obrázku 4 řetězec SEZNAM) a jménem souboru uchovávajícího toto nastavení (například smtp-SEZNAM.properties). Obsah tohoto souboru vytváří třída SmtpProperties.

3.4.3 Třída SmtpProperties

Do instance této třídy se ukládají data nastavení ze třídy PropertiesDialog pro odesílání e-mailů.

Ve svém konstruktoru si načte z instance třídy Model adresář, kde má hledat soubory s nastavením e-mailového účtu. Poté prohledá tento adresář a uloží si názvy všech dostupných souborů, které splňují podmínku, že se jedná o soubor s nastavením (o správnosti názvu rozhoduje třída SmtpAccountName). Poté si ze třídy CypherProperties načte jméno vybraného serveru.

Načítání a ukládání vlastností a tvorbu nového nastavení řeší funkce (předáváním instance třídy java.util.Properties):

```
public Properties getAccountProperties(
      String accountName);
public void updateAccount(String smtpAccountName,
      Properties props);
public void createNewAccount(String accountName,
      Properties props)
```

Tuto instanci v případě ukládání dat vytváří PropertiesDialog. Do souboru se ukládají pod tímto názvem. Kvůli tomu, že instanci třídy Properties vytváří jiná třída, jsou následující vlastnosti označeny jako public:

```
public static final String CYPHER_SMTP_HOST =
      "cypher.smtp.host";
public static final String CYPHER_SMTP_AUTH =
      "cypher.smtp.auth";
public static final String CYPHER_SMTP_PORT =
      "cypher.smtp.port";
public static final String CYPHER_SMTP_ENCRYPTION =
      "cypher.smtp.encryption";
public static final String CYPHER_SMTP_USER =
      "cypher.smtp.user";
public static final String CYPHER_SMTP_PASSWORD =
      "cypher.smtp.password";
public static final String CYPHER_MAIL_FROM =
      "cypher.smtp.from";
public static final String
      CYPHER_MAIL_TRANSPORT_PROTOCOL =
            "cypher.transport.protocol";
public static final String CYPHER_SECURE_AUTHENTICATION
      = "cypher.secure.authentication";
```

Heslo k účtu ukládá třída PropertiesDialog do souboru zaheslované. K šifrování používá funkce encryptPassword a decryptPassword z této třídy. Tyto funkce šifrují heslo šifrou DES (Data Encryption Standard), aby nebyly „human readable“. K samotnému šifrování je použita třída DesEncryptor.

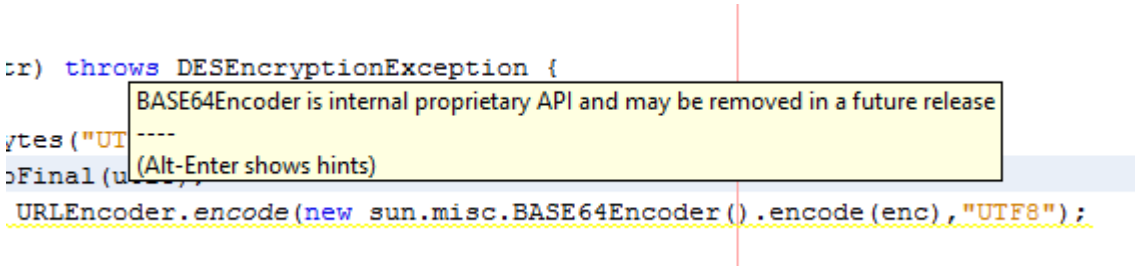
3.4.4 Třída ConnectionSecurity

Třída obsahuje výčtový typ ConnectionSecurityEnum, jehož obsah definuje jaké šifrování se použije při odesílání e-mailu. ConnectionSecurityEnum může nabývat tří hodnot: none (žádné šifrování), startTls (šifrování STARTTLS) a sslTls (SSL/TLS). Třída rovněž poskytuje dvě metody pro převod enum na řetězec a naopak.

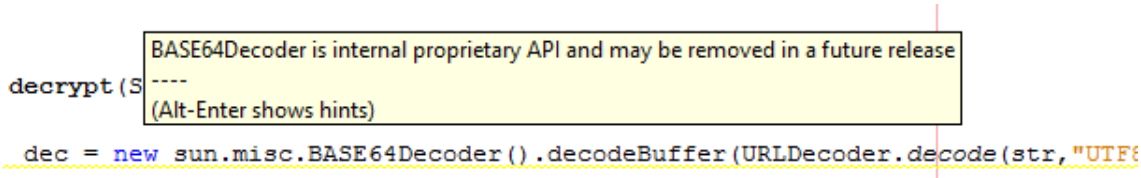
3.4.5 Třída DesEncryptor

Třída, která se využívá z bezpečnostních důvodů k zašifrování a dešifrování hesla k e-mailu, aby nebylo snadno čitelné „každému na očích“ zapsané v konfiguračním souboru SMTP-xxx.properties (kde xxx je název spojení (server)).

Znění třídy bylo podle komentáře ve zdrojovém kódu získáno z adresy http://www.exampledepot.com/egs/javax.crypto/DesString.html, ale tato adresa již není funkční. Nicméně téměř totožný kód nalezneme například na jedné stránce věnující se demo kódům [8].



Obrázek 7: Výřez z vývojového prostředí s varováním o BASE64Encoder



Obrázek 8: Výřez z vývojového prostředí s varováním o BASE64Decoder

Vývojové prostředí hlásí varování (viz obrázky 7 a 8), že třídy Base64Encoder a Base64Decoder z balíku sun.misc.* jsou interní proprietární API a v některém z následujících vydání Javy by už nemusely být přítomny. Tudíž je nutné je nahradit (viz oddíl 6.3).

3.5 Balík cyphergui.resources

Tento balík obsahuje soubory vlastností (s příponou .properties) s veškerými textovými řetězci oken grafického prostředí. Tento a následující balík je vytvořen knihovou appframework (o knihovnách více v oddíle 3.7) pro usnadnění překladu GUI do jiného jazyka.

3.6 Balík cyphergui.resources.busyicons

Balík obsahuje rozfázovaný pohyb (otáčení) kurzoru (soubory png) používaného pokud je počítač zaneprázdněný (například při šifrování a odesílání e-mailu studentům).

3.7 Importované knihovny

3.7.1 Swing Application Framework

Swing Application Framework, nebo Appframework [9], je knihovna, která slouží k usnadnění práce s tvorbou GUI malých a středně velkých aplikací. Má za úkol zjednodušit opakovaně prováděné operace při tvorbě GUI s použitím předpřipravených knihoven. Každá třída aplikace, která chce využívat výhody Swing Application Frameworku musí dědit buď ze třídy Application nebo SingleFrameApplication (ze které dědí třída CypherGUIApp).

Appframework definuje posloupnost volání metod při vytváření GUI. Jako první uživatel spustí metodu main, kde dojde k zavolání metody launch. Následuje metoda initialize (nepovinná) a startup. Po metodě startup je zavolána (opět nepovinná) metoda ready. Při zavírání aplikace uživatel nejprve zavolá metodu exit a poté je automaticky zavolána nepovinná metoda shutdown.

Pomocí ResourceMap se mohou objekty dotazovat na obsah souboru s názvem „JmenoTřídy.properties“ umístěného ve složce resources (tedy například soubor ServerDialog.properties ve složce cyphergui.resources). Účelem tohoto je usnadnit lokalizaci GUI do jiného jazyka snadným přepisem konfigurčního souboru místo hledání kde všude ve zdrojovém kódu je nutné přepsat řetězec.

Další věc, kterou Appframework přináší je anotace @Action. Názvy metod s touto anotací jsou přeneseny do instance třídy ActionMap a prvky GUI se pomocí ní opatří akcí – schématicky například:

```
prvek.setAction(instanceActMap.get("delej_tohle")));
```

Přisouzením akce lze prvku GUI rovněž přidat vlastnosti jako ikonu nebo popisek. V příslušném souboru resources totiž může být uvedeno něco takového:

```
delej_tohle.Action.text = tohle
delej_tohle.Action.icon = Icon.png
```

Poté, co se vývojáři Appframeworku nemohli dohodnout na pokračování vývoje a nestihli termín vydání Javy 7, byl projekt postupně zastaven [10]. V současné době je Swing Application Framework podporován jen komunitou uživatelů (plugin do NetBeans, Better Swing Application Framework). Aktuálně používanou verzi knihovny (1.03, poslední veřejně dostupná verze Appframeworku) je nutné nahradit (minimálně pomocí Better Swing Application Frameworku, viz oddíl 6.2).

3.7.2 Swing worker

Knihovna nutná pro správné fungování knihovny Swing Application Framework. SwingWorker umožňuje to, aby během vykonávání složitého úkolu „nezamrzlo“ GUI tím, že úkol spustí na pozadí [11]. (Například když se při vykreslování okna musí načítat obrázky. To nějakou dobu může trvat a během této doby nebude GUI odpovídat. Pokud se načítání obrázků bude provádět na pozadí, načte se GUI bez obrázků a ty se poté doplní.)

Swing worker zajišťuje správu vláken, informování volajícího objektu o tom, že zpracovávání vlákna bylo dokončeno a komunikaci vláken mezi sebou. I když je SwingWorker nyní součástí Java SDK (od verze 6), knihovna Appframework požaduje SwingWorker jako knihovnu (třídy v knihovně Swingworker a ty, které jsou součástí Java SDK nejsou úplně totožné, i když mají stejný úkol).

3.7.3 JavaMail

Knihovna JavaMail umožňuje vytvořit v jazyce Java aplikaci schopnou odesílání a čtení e-mailů. Obsahuje JavaMail API a řadu poskytovatelů jednotlivých protokolů vytvořených společností Sun [12].

Javamail API přináší především abstraktní třídy, ze kterých uživatel může vytvořit třídy řešící určité komunikační protokoly, nicméně obsahuje i třídy, které umožňují vytváření zprávy podle standardu RF822 a MIME.

Sun microsystems vybavil JavaMail API dalšími třídami, které implementují určitý protokol (ve verzi knihovny 1.4.1 použité v aplikaci Cypher to jsou SMTP, POP3 a IMAP) a není tedy potřeba, aby je sám tvůrce aplikace vytvářel z tříd Mail API.

3.7.4 SMTP

Tato knihovna obsahuje specifikaci poskytovatele protokolu SMTP, který je již obsažený v knihovně JavaMail, tudíž je zbytečné, aby se tato knihovna k aplikaci přidávala.

Druhou možností je použití osekané knihovny mailapi, která obsahuje pouze JavaMail API, a poté by tato knihovna byla potřebná. Navíc by se snížila náročnost aplikace na úložný prostor. Ale tento požadavek je v dnešní době téměř nepodstatný vzhledem k tomu, že ono snížení by bylo maximálně v řádu stovek kB.

4 Export do pdf

4.1 Export do pdf – teoretický rozbor

Pdf (portable document format) [13] je formát, který vznikl ve společnosti Adobe v devadesátých letech minulého století k tomu, aby umožnil tvorbu, prohlížení a tisk dokumentů mezi různými zařízeními bez změny formátování. V současné době je PDF volný formát, spravovaný ISO (norma ISO 32000-1:2008).

4.1.1 Objekty v pdf dokumentu

Podle normy [14] se PDF soubor skládá z objektů. V pdf se může nacházet osm typů objektů (booleovské hodnoty, čísla, řetězce (Strings), jména (Names), pole (Arrays), slovníky (Dictionaries), proudy (Streams) a objekt null).

Booleovské objekty jsou takové, které reprezentují hodnotu pravda, nepravda, v pdf se používají klíčová slova „true“ a „false“.

Čísla mohou být jak celá, tak reálná, ve způsobu práce s nimi není rozdíl.

Řetězec je posloupnost bytů zapsaná jako řada písmen v závorce nebo řada hexadecimálních hodnot ve špičaté závorce (<>).

Jméno je unikátní posloupnost znaků začínající lomítkem.

Pole je jednorozměrná sada objektů, které mohou být jakéhokoliv typu.

Slovník reprezentuje tabulku dvojic ve tvaru jméno-hodnota. Celá tabulka musí být uzavřená do dvojité ostré závorky (<</jmeno (hodnota1) /jmeno2 /hodnota2 /jmeno3 5>>, v tomto příkladu je hodnota1 typu řetězec, hodnota2 typu jméno a hodnota3 číslo pět). Slovník je obvykle používán k uložení atributů nějakého složitějšího objektu (třeba stránky).

Proud je posloupnost bytů před kterou je umístěn slovník definující vlastnosti proudu (povinná je pouze délka, pak tam může být například informace, jaký filtr použít pro dekódování, parametry filtru, nebo i cesta k externímu souboru, kde hledat data). Samotná posloupnost bytů je ohraničená klíčovými slovy „stream“ a „endstream“.

Null je nulový objekt. V dokumentu by měl být jen jeden objekt tohoto typu.

4.1.2 Struktura souboru

Samotný soubor se skládá z hlavičky, těla, kde jsou uložené jednotlivé objekty, tabulky s křížovými odkazy na nepřímé objekty (offset v bytech kde v souboru se nachází, viz norma [14], oddíl 7.5.4) a paty souboru (viz tamtéž, oddíl 7.5.5). Tělo dokumentu tvoří katalog, slovník, který obsahuje odkazy jak na jednotlivé stránky dokumentu a další informace, jako například data týkající se zobrazení dokumentu (na jaké stránce se má dokument otevřít, přiblížení stránky, rozložení stránek, …).

4.1.3 Grafika a text v pdf

Jakákoliv data se nacházejí v tzv. content streamu. Jedná se o objekt typu proud, kde se nachází data (např. obrázky, řetězce) a instrukce, jak je zobrazit (souřadnice ve stránce, kam mají být umístěny, u grafiky např. velikost, průhlednost, u textu použitý font). Ve specifikaci pdf je pět standardních (vestavěných) fontů (Courier, Helvetica, Times, Symbols a Zapf Dingbats (také symboly)). První tři jsou celkem ve čtyřech variantách (plain, kurzíva, tučný, tučná kurzíva). Vedle toho si tvůrce dokumentu může přidat i vlastní fonty, pokud mu ty vestavěné nestačí.

4.1.4 Možnosti exportu do pdf v javě

Pro export do pdf v javě existuje několik možností. Od mnou použitého open source Apache PDFBoxu [15], přes dvojlicencovaný iText [16], až po užití placených Adobe PDF Library SDK [17] (s použitím Java Native Interface, protože je napsaná v jazyce C) nebo PDFNet SDK [18].

Při řešení tohoto projektu jsem se rozhodoval mezi iText a Apache PDFBoxem. Největší výhodou PDFBoxu oproti iText je jeho licence. V případě iText (pokud bych použil iText ve varintě zdarma) bych musel vydat celý kód aplikace se stejnou licenci, jakou má iText (AGPL). Což bych velmi rád udělal, pokud bych byl jediným autorem programu. Ale v tomto případě, kdy je autorů víc a nemám od nich souhlas k volnému šíření zdrojového kódu, nemohu použít AGPL licenci. Apache licence, kterou má PDFBox, naopak dovoluje využití i pro aplikace s uzavřeným zdrojovým kódem.

4.1.4.1 iText

Výhodou knihovny iText je její jednoduchost použití podpořená perfektní dokumentací včetně rozsáhlé knihovny příkladů.

Knihovna iText je poskytována se dvěma různými licencemi. První je komerční licence a druhá je licence AGPL (Affero General Public Licence) pro použití zdarma.

4.1.4.2 Apache PDFBox

Apache PDFBox je open source knihovna pro vytváření a upravování souborů ve formátu pdf. Rovněž umožňuje vyjímání dat z těchto dokumentů. I když nenabízí takovou řadu funkcí a na stránkách projektu nenajdeme tolik příkladů jako u iTextu, vzhledem k licenci je vhodnější.

Při práci s pdf soubory umožňuje PDFBox dvojí přístup, nabízí dva modely. První z nich, COS model, přistupuje k tvorbě pdf na nižší úrovni a umoňuje úpravu jednotlivých objektů v dokumentu. Druhý, PD model poskytuje vyšší míru abstrakce, poskytuje programátorovi možnost se odpoutat od konkrétní specifikace formátu pdf. Nicméně v případě potřeby lze mezi oběma modely přecházet.

4.2 Export do pdf – samotné řešení

4.2.0 Úvod a přípravné fáze

Pro export do pdf jsem zvolil PDFBox ve verzi 2.0.0 RC3 (Release Candidate), jelikož finální verze 2.0.0 nebyla ještě zatím vydána a verze 1.8.11 nepodporuje v textu Unicode (nešly zobrazit háčky). Pro podporu háčků je rovněž nutné do dokumentu vkládat vlastní font, jelikož ty vestavěné ve standardu pdf je neumí. Rozhodl jsem se pro font DejaVu Mono[1]. Ten bylo nutné po stažení rozbalit do nově vytvořeného adresáře font. Do adresáře font jsem také přidal font Open Sans.[2]

V přípravné fázi programování jsem stáhl a přidal do vývojového prostředí potřebné knihovny. Konkrétně se jednalo o soubory pdfbox-2.0.0-RC3.jar, fontbox-2.0.0-RC3.jar[3] a dále commons-logging-1.2.jar[4].

1 domovská stránka: http://dejavu-fonts.org/wiki/Main_Page, přímý odkaz ke stažení: http://sourceforge.net/projects/dejavu/files/dejavu/2.35/dejavu-fonts-ttf-2.35.zip

2 dostupný například ze stránky opensans.com, přímý odkaz: http://www.opensans.com/download/open-sans.zip

3 dostupné ze stránky https://pdfbox.apache.org/download.cgi#20x

4 dostupná ze stránky http://commons.apache.org/proper/commons-logging/download_logging.cgi, knihovnu je nejprve nutné rozbalit z archivu commons-logging-1.2-bin.zip

Samotný postup řešení exportu jsem rozdělil celkem do osmi kroků.

Prvním krokem bylo vytvořit třídu PDFMaker (zatím bez užitečné aktivity).

Dále bylo nutné upravit GUI a třídu Controler tak, aby bylo možné zvolit export do pdf a zavolat příslušnou funkci z třídy PDFMaker (jinou možností by bylo k jejímu volání vytvořit pomocnou třídu s funkcí main). V souvislosti s tím jsem upravil třídy Model a CypherProperties, do kterých se ukládají informace (Model – při běhu aplikace, CypherProperties – mezi dvěma spuštěními aplikace).

Při testování odesílání cvičného souboru přišel testovací soubor poškozený, takže bylo nutné upravit třídu MailSender.

Dále jsem upravil třídu TaskDefinition, aby uměla vytvářet výstup jako pole řetězců (znaky \n v původním výstupním řetězci jsou při tvorbě PDF ignorovány, celý text by byl na jednom řádku).

Posledním krokem bylo vybavit užitečným kódem třídu PDFMaker, aby vytvářela pdf soubor s korektními daty.

4.2.1 PDFMaker (základní funkce)

Vytvořil jsem třídu PDFMaker, která obsahovala konstruktor, ve kterém této třídě předávám instanci třídy TaskDefinition. Třída PDFMaker dále obsahovala funkci

```
public File makePDF(String fileName, boolean
showPlainText)
```

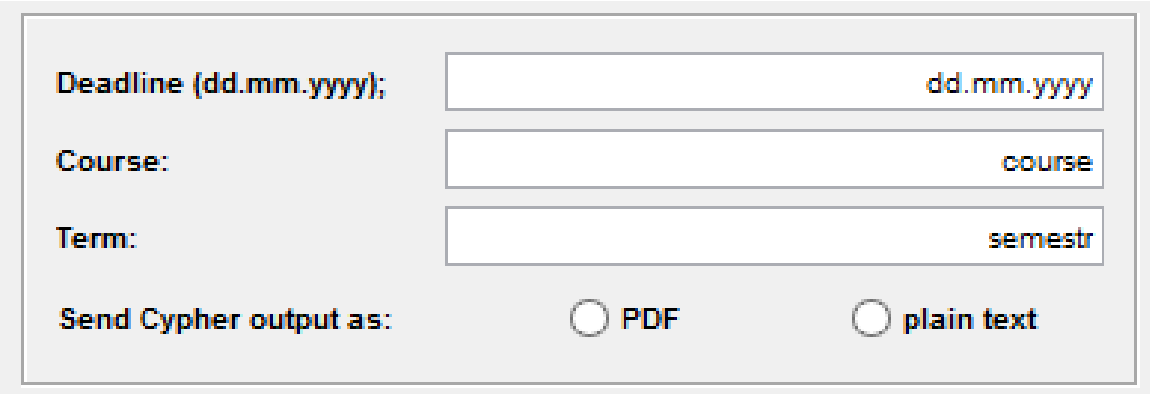
která vytvořila jednoduchý pdf soubor (s textem Hello World) podle kuchařky umístěné na stránkách PDFBox [19] (ze začátku jsem používal knihovnu PDFBox ve verzi 1.8.10, až poté jsem kód upravil na nejnovější verzi pro podporu Unicode).

4.2.2 Úprava GUI

Upravit stačilo třídu CypherGUIView. Jelikož GUI bylo vytvářeno pomocí GUIBuilderu, upravil jsem ho také tak.

Přidal jsem celkem čtyři komponenty. Dva jRadioButtons (jPDFSelButton a jPlainTextSelButton), které jsem přidal do skupiny jButtonGroup a ještě jLabel s popiskem: „Set Cypher output as:“. Tento text, stejně jako popisky jRadioButtons a použité fonty (Dialog Bold pro jLabel a Dialog Plain pro popisky radiobuttons, oba fonty ve velikosti 10pt) se díky knihovně SwingWorker přidaly automaticky do souboru CypherGUIView.resources.

Skupinu jRadioButtons a jLabel jsem umístil do spodního panelu vpravo, hned pod trojici jLabels a jTextFields Deadline, Course a Term.



Obrázek 9: Výřez z okna aplikace – výsledný panel vpravo dole

Do funkcí sendToOneHandler a sendToAllHandler jsem téměř na začátek (hned za kontrolu, zda jsou všechna potřebná pole vyplněná) přidal příkaz

```
model.setPDFOutput(CypherOutputButtonGroup.
     isSelected(jPDFSelButton.getModel()), true);
```

který do instance třídy Model uloží, zda je vybrán export do PDF (jak jsem psal v oddílu 3.2.1, třída Model slouží jako „studnice informací“, obsahuje mimo jiné veškerá data, která uživatel vloží do GUI). Z třídy Model si tuto informaci vyzvedne poté třída Controler, která podle toho zareaguje (viz další oddíl).

Při inicializaci GUI, se zase z instance třídy Model načítá informace, jaký radio button má být při spuštění stisknutý (i když jsou ve stejné skupině, je nutné nastavovat oba).

```
CypherOutputButtonGroup.
     setSelected(jPDFSelButton.getModel(),
          model.isPDFOutput());
CypherOutputButtonGroup.
     setSelected(jPlainTextSelButton.getModel(),
          !model.isPDFOutput());
```

4.2.3 Controler

V této třídě jsem upravil proceduru processOneRecipient (puntičkářsky řečeno: z procedury proccessOneRecipient (sic) jsem vytvořil proceduru processOneRecipient, kam jsem doplnil tvorbu pdf). Celé znění této procedury najdete v dodatku.

Dále jsem do funkce getFileName přidal rozhodování, zda je vybrán výstup do pdf a podle toho se do jména souboru pro učitele přidává přípona .pdf nebo původní .txt.

4.2.4 Model

Do třídy Model byly přidány navíc dvě proměnné.

```
private boolean PDFOutput;
private String PDFAttachmentFilename;
```

K proměnné PDFOutput jsem vytvořil dvě metody isPDFOutput, která vrátí true, pokud je vybrán výstup do pdf, a false v opačném případě, a metodu setPDFOutput, která tuto proměnnou nastaví. K proměnné PDFAttachmentFilename jsem také přidal funkci getPDFAttachmentFilename a setPDFAttachmentFilename.

```
public boolean isPDFOutput()
public void setPDFOutput(boolean PDFOutput,
     boolean updateProperty)
public String getPDFAttachmentFilename()
public void setPDFAttachmentFilename(String filename,
     boolean updateProperty)
```

V inicializační části třídy Model jsem přidal načítání těchto proměnných ze třídy CypherProperties.

```
setPDFOutput(CypherProperties.getInstance().
     isPDFOutput(), false); //updateProperty: pokud
//by zde bylo true, hodnota, na kterou se PDFOutput
//nastavuje, by se zbytečně uložila do souboru
//Cypher.properties (odkud ji právě čtu)

if(isPDFOutput())
     setPDFAttachmentFilename(CypherProperties.
          getInstance().
               getPDFAttachmentFilename(), false);
```

4.2.5 CypherProperties

Do této třídy jsem přidal definici čtyř řetězců. První dva definují pod jakými řetězci jsou dané informace uloženy v souboru Cypher.properties, druhé dva definují výchozí hodnotu těchto informací.

```
private static final String
     CP_IS_PDF_OUTPUT="cypher.is.PDF.output";
private static final String
     CP_PDF_ATTACHMENT_FILENAME=
          "cypher.pdf.attachment.filename";
private static final String
     CP_DEFAULT_PDF_OUTPUT_SELECTION="true";
private static final String
     CP_DEFAULT_PDF_ATTACHMENT_FILENAME="Task.pdf";
```

Dále jsem podle vzoru ostatních funkcí vytvořil čtveřici funkcí k získání a nastavení jejich hodnoty.

```
public void setPDFAttachmentFilename(String filename)
public String getPDFAttachmentFilename()
public void setPDFOutput(boolean option)
public boolean isPDFOutput()
```

Po těchto operacích byla aplikace opět spustitelná. Po zvolení exportu do pdf nicméně pouze vypsala na konzoli a odeslala pouze cvičný soubor. Jak jsem zjistil, přiložený cvičný soubor přišel v e-mailu poškozený, takže bylo nutné upravit ještě funkci odesílající e-maily.

4.2.6 MailSender

Možnost přidávání přílohy do e-mailu byla již naprogramována, nicméně, jak jsem zjistil, nefungovala tak, jak měla, protože soubor dorazil poškozený. Původní kód

```
if (attachedFiles != null) {
  for (File file : attachedFiles) {
    try {
      mbp = new MimeBodyPart(
                    new FileInputStream(file));
      mbp.setFileName(file.getName());
      mp.addBodyPart(mbp);
    } catch (FileNotFoundException ex) {
      logger.log(Level.SEVERE, null, ex);
    }
  }
}
```

jsem nahradil tímto

```
if (attachedFiles != null) {
  for (File file : attachedFiles) {
    try {
      mbp = new MimeBodyPart();
      mbp.attachFile(file);
      mp.addBodyPart(mbp);
    } catch (IOException ex) {
      logger.log(Level.SEVERE, null, ex);
    }
  }
}
```

Poté již přiložený soubor dorazil v pořádku.

4.2.7 TaskDefinition

Pro správné vytvoření pdf bylo nutné rozdělit každý řádek do samostatného souboru. Přišlo mi jednodušší vytvořit například ArrayList řetězců rovnou ve funkci TaskDefinition, než jednotlivé řádky složitě parsovat z jednoho dlouhého řetězce podle výskytu \n (těch mohlo být i víc za sebou).

Proto jsem vytvořil speciální „PDF“ verzi každé funkce z této třídy, která zpracovává textový výstup aplikace Cypher (prázný řádek nahrazen pomocí null). Vznikly tedy tyto funkce (celé najdete v příloze):

```
private ArrayList <String> getHeader4PDF()
private ArrayList<String> getInstructions4PDF()
private ArrayList<String>
     getTaskDescription4PDF(CypherTask task,
          boolean showPlainText)
```

Poslední funkce výstupy předchozích private funkcí spojila do jednoho ArrayListu.

```
public ArrayList<String>
     getTaskDefinitionProtocol4PDF(
          boolean showPlainText)
```

4.2.8 DocumentCursor

Do PDF Fokumentu nelze psát „standardním způsobem“, jak jsme zvyklí, protože řádky v dokument streamu mohou následovat v libovolném sledu. „Standardní způsob“ jsem se snažil emulovat použitím instance třídy DocumentCursor, kterou jsem za tím účelem vytvořil, proto se před třídou PDFMaker nejprve podívejme na ni.

Konstruktor třídy DocumentCursor je dvojí. První přijímá pouze dvojici parametrů – velikost stránky (tedy vlastně souřadnice pravého horního rohu, jelikož je stránka v PDF souboru měřená od levého dolního rohu). Druhý konstruktor rovněž přijme velikost a řádkování (relativně vůči velikosti) použitého fontu (při použití prvního konstruktoru je nutné poté nastavit metodou setLineHeight výšku řádku (leading*size)).

```
public DocumentCursor(int xMax, int yMax)
public DocumentCursor(int xMax, int yMax, int size,
     double leading)
```

Okraje stránky jsou zajišťovány konstantou (v budoucnu bude možné tuto hodnotu nastavit v GUI).

```
private final int MARGIN=80;
```

I když DocumentCursor obsahuje i proměnnou x, není s ní pracováno (ačkoliv je možné získat délku vytištěného řádku), nicméně jsou používány metody jak getXPos i getYPos, od kterých ve třídě PDFMaker začíná výpis řádku.

Vedle dalších metod set* a get* je jedna z nejdůležitějších funkcí v této třídě

```
public void newline()
```

která odřádkuje, tedy posune pozici kurzoru o řádek dále a pak dvojice funkcí

```
public boolean isAtTheEndOfPage()
public void resetCursor()
```

V případě, že funkce isAtTheEndOfPage vrátí true, přidá se do dokumentu nová stránka a kurzor se resetuje, tedy nastaví na začátek nové stránky.

Neméně důležitou funkcí je funkce

```
public int getMaxLen()
```

která se používá k porovnávání zda se řádek na šířku stránky vejde nebo ne.

4.2.9 PDFMaker (finální verze)

Jelikož jsem měl tuto třídu již rozpracovanou, posledním úkolem, který zbýval, bylo doprogramovat funkci makePDF.

Výsledná funkce makePDF funguje takto:

Nejprve vytvoří nový PDDocument. Do něj vloží vybraný font DejaVuSansMono a uloží ho do proměnné font.

```
PDType0Font font=PDType0Font.load(document,
     new File("DejaVuSansMono.ttf"));
```

Poté vytvoří content stream první stránky přidané do dokumentu pomocí funkce (jelikož se jedná o první stránku, tak jako předchozí content stream jsem vložil null)

```
private PDPageContentStream writeInNewPage (
     PDDocument doc, PDPageContentStream previous)
```

První přidanou stránku extrahuje a použitím funkce getMediaBox získá objekt typu PDRectangle, který mimo jiné obsahuje rozměry stránky [20]. S pomocí nich vytvoří instanci třídy DocumentCursor.

Poté již získá ArrayList jednotlivých řádků od instance třídy TaskDefinition příkazem

```
content=taskDef.getTaskDefinitionProtocol4PDF(
                    showPlainText);
```

a poté je v cyklu zapíše do pdf souboru (pokud jeden ze řádků je null, tak pouze odřádkuje kurzor). Jelikož jsou však některé řádky delší, než se vejde na stránku, je nutné je rozdělit. Buď v mezeře (pokud tam nějaká je) nebo po počtu znaků daných funkcí getMaxLen z instance třídy DocumentCursor.

K samotnému zápisu do pdf dokumentu slouží další funkce

```
private void writeLine(DocumentCursor curs,
     String line, PDPageContentStream contentStream,
     PDFont font, int fontSize)
```

Na závěr své činnosti funkce writePDF uzavře content stream poslední zapsané stránky, uloží dokument pod názvem fileName, uzavře ho a vrátí ho jako instanci třídy File, která je poté ve třídě Controler buď odeslána e-mailem nebo zahozena (soubor je již uložen, není nutné jej ukládat znovu).

5 Další drobná vylepšení

5.1 sendToOneHandler

Ve třídě CypherGuiView z balíku cyphergui jsem ve funkci sendToOneHandler přesunul volání, zda je k dispozici další adresát, nebo ne. V původní verzi totiž tato funkce po zpracování předposledního adresáta posune iterátor (takže ukazuje na poslední) a zeptá se třídy Model, zda tam je nějaký další. Jelikož po posledním adresátovi samozřejmě již žádný není, odpoví třída Model záporně a funkce zobrazí dialogové okno, že již byli zpracováni všichni adresáti a převine iterátor na začátek seznamu adresátů, takže poslední adresát nemůže dostat zadání. Po přesunu volání hasNextRecipient na začátek již všechno funguje tak, jak má.

5.2 Switch

Ve stejné třídě jsem dále ve funkci propertyChange nahradil konstrukci s několika vnořenými if-else na switch, u kterého je možné použití řetězce od Javy 7 [21]. Implementace switche pro řetězce pracuje s unikátními integery pro každý řetězec. Ty se vypočítají pomocí hashovací funkce. Switch se tedy může zdát pomalejší než if-else, protože se musejí tyto hashe počítat, nicméně většinou bývají již jednou vypočítané hashe uložené v mezipaměti, takže při druhém běhu již zpomalení nepoznáme a naopak bude rychlejší (switch s integery je mnohem rychlejší oproti if-else). V případě shody hashe se teprve porovnávají řetězce (s použitím if). Vícenásobný if byl tedy převeden na výpočet hashů (častokrát pouze podívání se do paměti, kde je již vypočítaný hash), switch a jeden if.

5.3 Oprava překlepu v parametru minLenght

Překlep v názvu parametru funkce getRandomPassword ze třídy PasswordBook byl opraven na minLength.

5.4 Knihovna smtp

Z přidaných knihoven byla odstraněna knihovna smtp, jelikož její třídy jsou již obsaženy v knihovně javaMail.

6 Návrhy na další zlepšení

6.1 Balík encoding

Složené algoritmy v tomto balíku jsou implementovány dvěma rozdílnými způsoby. Myslím, že je nutné tuto „schizofrenii“ v implementaci složených šifer odstranit. Problémem je, k jaké metodě se přiklonit. Zda k té starší (název složené šifry jako pole stringů, podle stringu vyhledat příslušné jednoduché algoritmy a zřetězit je) nebo dědění z abstraktní třídy SlozenaSifra a v příslušné zděděné třídě implementovat funkci pro kódování. Podle mého názoru je lepší použít metodu docenta Kouby, jelikož jeho algoritmus je univerzální a funguje i pro zřetězení více šifrovacích metod než dvou.

Na druhou stranu je kód docenta Kouby nevhodně umístěn ve třídách TaskDefinition (vytvoření složené šifry) a Model (načtení šifer). Podle mě by bylo lepší, kdyby třídy Model ani TaskDefinition nemusely rozlišovat mezi složeným a jednoduchým algoritmem. Pro ně stačí si šifru představit jako černou skříňku, které předám otevřený text a heslo (obojí o vhodné délce) a dostanu šifrovaný text. To, že uvnitř si text postupně předává více šifer je vůbec nemusí zajímat, natož aby se o zmíněné předávání samy staraly, jako v případě třídy TaskDefinition.

Toto zřetězování by bylo možné přesunout například do třídy SlozenaSifra (implementací metody encode, jelikož i třída SlozenaSifra implementuje rozhraní EncodeAlgorithm). V metodě init třídy TaskDefinition by tedy poté byl pouze kód, který po zjištění potřebných parametrů vybrané metody (délka otevřeného textu, délka hesla) zavolá funkci encode a tím získá šifrovaný text bez ohledu na to, zda se jedná o složenou či jednoduchou šifru.

Do zmiňované třídy SlozenaSifra je také nutné přesunout z balíku cypher obsah třídy CompoundAlgorithmName pro tvorbu názvu složené šifry (implementace metody getName). Po vyřízení názvu složené šifry odpadá nutnost, aby návratový typ funkce getName v rozhraní EncodeAlgorithm byl pole řetězců.

Třída AlgorithmManager by si také možná zasloužila úpravu. Nově by se starala o vytvoření seznamu algoritmů použitelného přímo třídou Model. V současném stavu totiž AlgorithmManager vyhledá ve složce třídy algoritmů. Jejich seznam pošle třídě Model a ta vytvoří HashMap instance třídy a jména algoritmu. Jelikož toto je jediné volání funkce getAlgorithms(), nevidím důvod, proč by návratová hodnota nemohla být rovnou více „Model-friendly“, tedy aby s ní třída Model neměla další „zbytečnou“ práci. Tento HashMap by tedy mohl být vytvářen již ve třídě AlgorithmManager.

6.2 Balík cypher

Třídy PDFMaker a DocumentCursor, které jsem umístil do tohoto balíku, bych chtěl v budoucnu upravit, aby bylo možné nastavit font, jeho velikost a řádkování přímo v GUI programu.

6.3 Balík cyphergui

Některé třídy z balíku cyphergui by bylo vhodné přepsat, aby nebyly závislé na knihovně Swing Application Framework, jelikož ta již od roku 2009 není podporovaná [22]. Nejjednodušší je použít nástupce, zpětně kompatibilní (ve verzi 1.9.x) Better Swing Application Framework, který je spravovaný původními tvůrci Swing Application Frameworku. Nicméně do budoucna by bylo nejlepší přepsat kód GUI tak, aby nebyl nutný žádný framework.

6.4 Balík conf

Zde je nutné nahradit Base64Encoder a Base64Decoder ve třídě DESEncryptor, jelikož se jedná o proprietální API z balíku sun.misc.*. Použít je možné například třídu Base64 z balíku java.util.* [23]

7 Závěr

V rámci práce na projektu jsem se seznámil s aplikací Cypher a naprogramoval výstup aplikace do pdf. V navazující bakalářské práci je nutné ještě výstup do pdf vylepšit a přidat možnosti uživatelského nastavení jak exportu do pdf, tak i parametrů jednotlivých šifer. Dalším úkolem v bakalářské práci bude zrealizovat návrhy na zlepšení uvedené v oddílu 6.

Dalším úkolem v bakalářské práci bude implementace nových šifer (RSA a Baby Rijndael).

8 Seznam použité literatury

[1]: TICHÁ, Ludmila PhDr., Mgr. Zdeňka CIVÍNOVÁ, Mgr. Michaela MARYSKOVÁ, Mgr. Ilona TRTÍKOVÁ, Mgr. Lenka NĚMEČKOVÁ: *Jak psát vysokoškolské závěrečné práce*. Ústřední knihovna ČVUT, 2009, akt. 2014.

[2]: *Welcome to NetBeans* [online]. Oracle Corporation. 2015 [cit. 2015-10-31]. Dostupné z: https://netbeans.org/index.html

[3]: *SAF Support* [online]. Oracle Corporation. 2012 [cit. 2016-02-06]. Dostupné z: http://wiki.netbeans.org/SAFSupport

[4]: *Swing Application Framework Support* [online]. Oracle Corporation. 2012 [cit. 2016-02-06]. Dostupné z: http://plugins.netbeans.org/plugin/43853/swing-application-framework-support

[5]: *Overview (Java Platform SE 8)* [online]. Oracle Corporation. 2016 [cit. 2016-02-08]. Dostupné z: https://docs.oracle.com/javase/8/docs/api/overview-summary.html

[6]: MARTIN, Keith M.: *Everyday Cryptography.* Oxford university press, 2012. ISBN 978-0-19-969559-1

[7]: KLIMA, Richard E. a Neil P. SIGMON: *Cryptology: Classical and Modern with Maplets*. CRC Press, 2012. ISBN 978-1-4398-7241-3

[8]: *Encrypting a String with DES: DES* [online]. Demo source and support. rok neuveden [cit. 2016-02-08]. Dostupné z: http://www.java2s.com/Code/Java/Security/EncryptingaStringwithDES.htm

[9]: O'CONNER, John. *Using the Swing Application Framework*. 2007 [cit. 2016-02-08]. Dostupné z: http://www.oracle.com/technetwork/articles/javase/index-141957.html

[10]: *Swing Application Framework - Wikipedia, the free encyclopedia* [online]. . 2014 (poslední modifikace) [cit. 2016-02-09]. Dostupné z: https://en.wikipedia.org/wiki/Swing_Application_Framework

[11]: *Worker Threads and SwingWorker (The Java Tutorials)* [online]. Oracle Corporation. 2015 [cit. 2016-02-09]. Dostupné z: https://docs.oracle.com/javase/tutorial/uiswing/concurrency/worker.html

[12]: *JavaMail 1.4.3 Release Notes* [online]. Oracle Corporation. [cit. 2016-02-10]. Dostupné z: http://www.oracle.com/technetwork/java/notes-2-149762.txt

[13]: *Soubory PDF, formát Adobe PDF | Adobe Acrobat DC* [online]. Adobe Systems Incorporated. 2016 [cit. 2016-02-10]. Dostupné z: https://acrobat.adobe.com/cz/cs/products/about-adobe-pdf.html

[14]: ISO 32000-1:2008 Document management - Portable Document Format - Part1: PDF 1.7. ICS: 35.240.30.First Edition 2008-07-01. International Organization for Standartization. Dostupné také z: http://wwwimages.adobe.com/www.adobe.com/content/dam/Adobe/en/devnet/pdf/pdfs/PDF32000_2008.pdf

[15]: *Apache PDF Box | A Java PDF Library* [online]. The Apache Software Foundation. 2015 [cit. 2016-02-11]. Dostupné z: https://pdfbox.apache.org/

[16]: *iText* [online]. iText Group NV. 2016 [cit. 2016-02-11]. Dostupné z: http://itextpdf.com/

[17]: *Adobe PDF Library SDK* [online]. Adobe Systems Incorporated. 2015 [cit. 2015-11-29]. Dostupné z: http://www.adobe.com/devnet/pdf/library.html

[18]: *PDFTron: PDF Components and PDF Tools* [online]. PDFTron Systems Inc.. 2015 [cit 2015-11-29]. Dostupné z: https://www.pdftron.com/pdfnet/index.html

[19]: *Apache PDFBox | Cookbook - Document creation* [online]. The Apache Software Foundation. 2015 [cit 2015-02-10]. Dostupné z: http://pdfbox.apache.org/1.8/cookbook/documentcreation.html

[20]: *java - How to generate multiple lines in PDF using Apache pdfbox* [online]. Uživatel mkl, Stack exchange. 2016 [cit. 2016-02-14]. Dostupné z: http://stackoverflow.com/questions/19635275/how-to-generate-multiple-lines-in-pdf-using-apache-pdfbox

[21]: *Java - Why can´t I switch on a String - Stack Overflow* [online]. Uživatel erickson, Stack exchange. 2016 [cit. 2016-02-07]. Dostupné z: http://stackoverflow.com/questions/338206/why-cant-i-switch-on-a-string

[22]: *Better Swing Application Framework* [online]. Oracle Corporation. 2014 [cit. 2016-02-06]. Dostupné z: https://kenai.com/projects/bsaf/pages/Home

[23]: *Base 64 (Java SE 8)* [online]. Oracle Corporation. 2016 [cit. 2016-02-08]. Dostupné z: http://docs.oracle.com/javase/8/docs/api/java/util/Base64.html