Fakulta informatiky a informačných technológií, STU Bratislava

Šifrovanie

(dokumentácia k projektu z OOP)

Autor: Martin Lipták
Dátum: 26. apríl 2010
Semester: letný

Tabuľka s obsahom

Úvod

Nasledujúce strany sú dokumentáciou k projektu z predmetu Objektovo-orientované programovanie. Časti Inštalácia a Používanie opisujú program z používateľského hľadiska, časť UML diagramy tried z toho technického. Pripojil som aj upresnené zadanie, v závere som zhodnotil, čo z neho a aj z rámcových požiadaviek som naplnil.

Inštalácia

Pred skúšaním programu je nutné najskôr splniť nasledujúce požiadavky

- Java SE 6.0

K používaniu programu nie je nutná jeho inštalácia. Vo väčšine prostredí by sa mal spustiť po dvojitom kliknutí na súbor s programom (*sifrovanie.jar*). Ak sa tak nedeje, môžeme vyskúšať spustenie z konzoly.

$ cd <cesta k programu>
$ java -jar sifrovanie.jar

Požívanie

Okno programu je rozdelené na 3 hlavné časti - vstup, šifrovanie a výstup.

Obrázok 1: Okno programu

Vstup

Na začiatku sa vyberie vstup. Po zvolení sa zobrazia možnosti pre daný typ. Program pracuje s troma typmi vstupov:

1. Textová správa zadaná do okna programu.
2. Súbor. Môže obsahovať akýkoľvek typ údajov – textový, obrázok, dokument, video, ...
3. Prihlasovacie údaje (doména, meno, heslo). Používateľ môže využiť ako bezpečné úložisko svojich najcitlivejších prístupových údajov.

Šifrovanie

Ďalej sa vyberie spôsob šifrovania a zadá sa kľúč. Program pozná 2 typy šifrovaní:

1. Posuvné šifrovanie – každý bajt vstupu sa zväčší o zadanú hodnotu kľúča. Kľúč musí byť v rozsahu od -128 do 127. Známe z histórie (aj keď aplikované na abecedu) ako Cézarova šifra.
2. XOR šifrovanie – na každý bajt vstupu sa vykoná bitová operácia XOR. Ďalším operandom je hodnota kľúča. Zaujímavosťou tejto šifry je, že dešifrovanie sa realizuje rovnako ako pôvodné šifrovanie. Kľúč musí byť v rozsahu od -128 do 127. Funkcia XOR (non-exclusive or) je známa jej častým použitím v kryptografii.
3. Žiadne. Dáta sa nebudú šifrovať, len pretečú zo vstupu na výstup. Vhodné na uloženie správy alebo prihlasovacích údajov do súboru.

Výstup

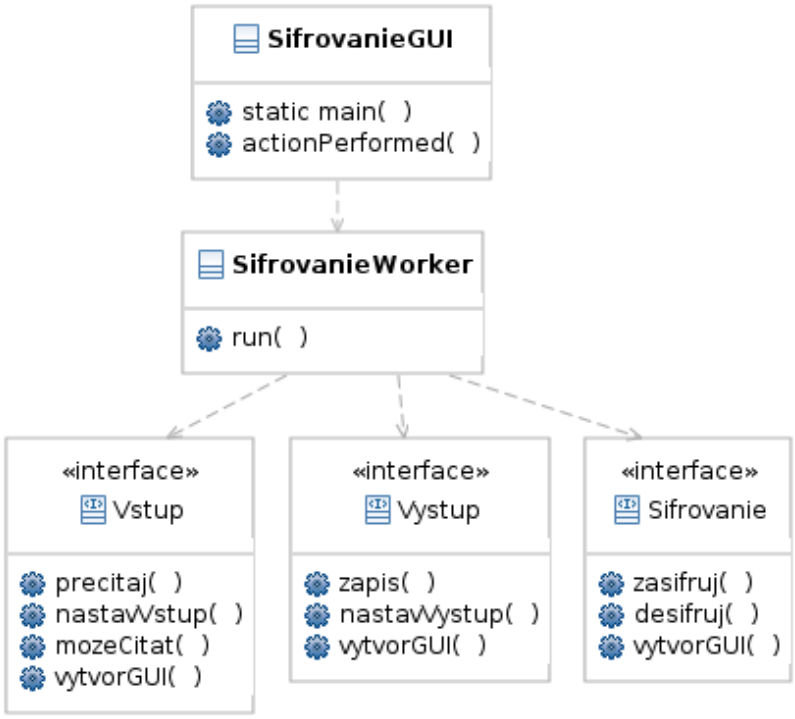
Nakoniec sa vyberie výstup. Po zvolení sa zobrazia možnosti pre daný typ. Program pracuje s troma typmi výstupov:

1. Textová správa vypísaná do okna programu.
2. Súbor uložený na používateľom špecifikované miesto
3. Prihlasovacie údaje (doména, meno, heslo). Využíva sa na zobrazenie dešifrovaných údajov

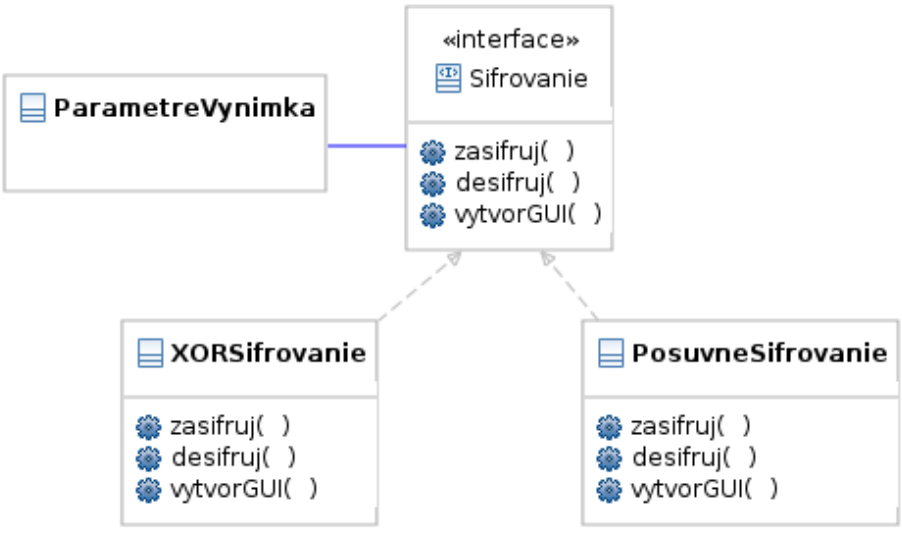
Práca s programom je celkom jednoduchá, intuitívna a používateľsky prívetivá. Každý si snáď všimne jej podobu s princípom rúry známym z unixových operačných systémov – vstup | šifrovanie | výstup.

Pri činnosti programu sa na štandardný výstup vypisujú rôzne informácie užitočné pre ladenie alebo sledovanie jeho priebehu.

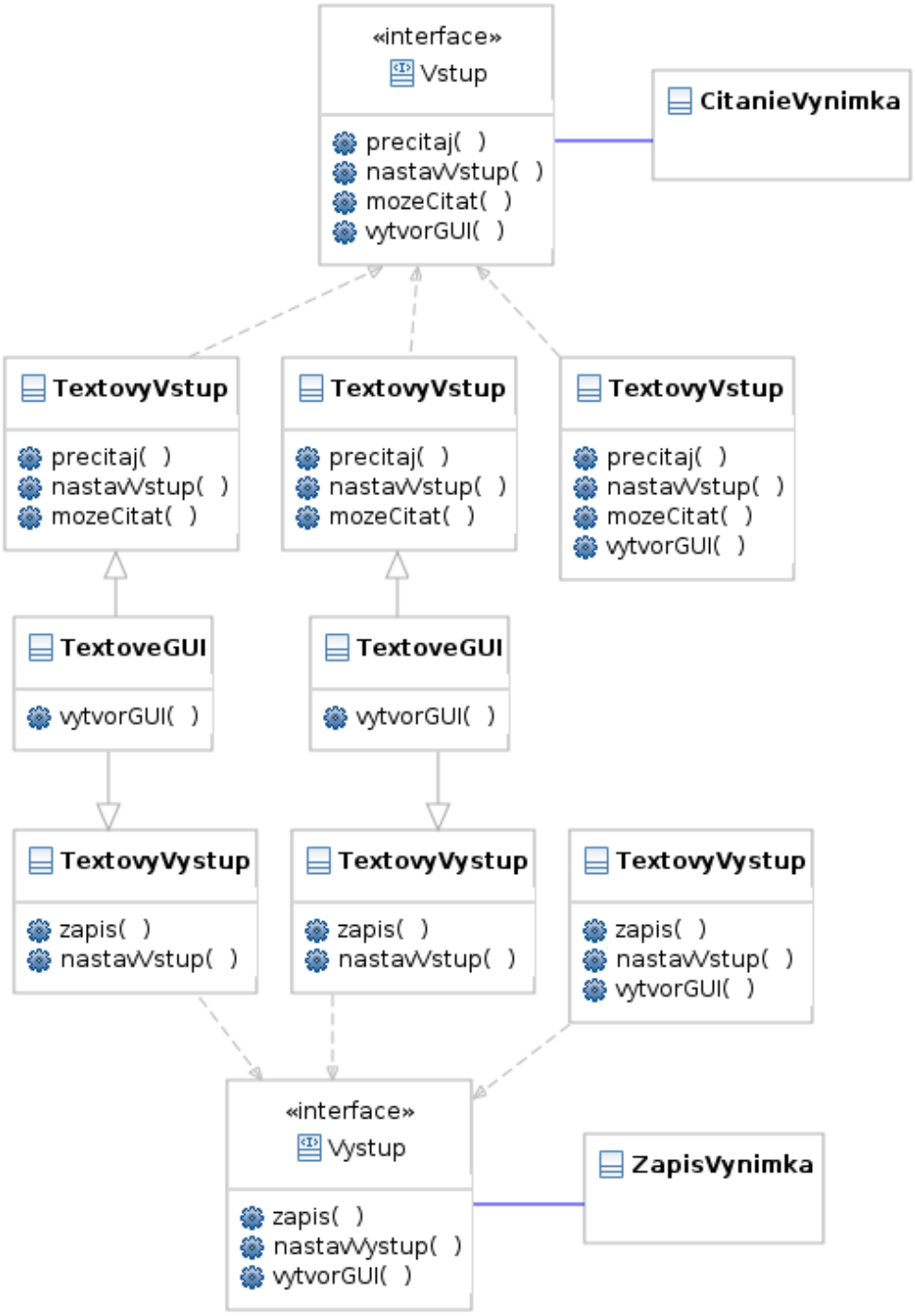
UML diagramy tried



Obrázok 2: Program



Obrázok 3: Typy šifrovaní



Obrázok 4: Typy údajov

Upresnené zadanie

Program bude pracovať s rôznymi typmi dát a šifrovaní. Bude možné si vybrať vstupný typ dát, spôsob šifrovania alebo dešifrovania a výstupný typ dát.

Používateľské rozhranie bude GUI. Používateľ najskôr vyberie vstupný typ dát. Podľa toho sa mu zobrazia možnosti špecifické pre daný typ. Ďalej určí typ šifrovania a či chce data zašifrovať alebo dešifrovať. Zobrazia sa mu voľby pre daný typ šifrovania. Nakoniec zadá typ výstupných dát.

Typy dát

1. Textová správa zadaná priamo do okna v programe, keď sa použije ako vstupný typ. Pri výstupe správu vypíše do okna.

2. Súbor. Do program sa zadá cesta k súboru. Celý súbor sa binárne otvorí a ďalej sa môže po bajtoch zašifrovať. Pri výstupe sa výsledok uloží do súboru.

3. Prihlasovacie údaje na Internet s názvom servera, používateľským menom a heslom.

Typy šifrovaní

1. Posuvné šifrovanie. Kľúčom bude číslo, ktoré určí, o akú hodnotu sa každý bajt zvýši. Pri dešifrovaní bude hodnota od každého bajtu odpočítaná.

2. Substitučné šifrovanie. Kľúčom bude reťazec 26 písmen. Každé písmeno dát sa nahradí písmenom z reťazca podľa poradia v abecede. Používateľa upozorní, že pre binárne súbory algoritmus nemusí byť efektívny.

Výkonnosť program sa môže zvýšiť využitím paralelizmu. Používateľ má možnosť nastaviť počet pracujúcich vlákien programu. Vďaka vhodnému návrhu bude možné kedykoľvek pridať nový spôsob šifrovania alebo nový typ dát.

Záver

Môj program síce nevyužíva návrhový vzor Visitor ani idióm Double Dispatch, ale jeho návrh spĺňa požiadavky zadania na dedenie, polymorfizmus a zapuzdrenie.

Kód som organizoval do jedného balíka. Moje rozhrania predpisujú vlastné výnimky. Grafické používateľské rozhranie som programoval v Eclipse bez návrhára GUI, aby som sa viac naučil a mal jednoduchšie a čistejšie napájanie na triedy „jadra“. V upresnenom zadaní som spomenul paralelizovanie priebehu šifrovania. Nakoniec som si to rozmyslel, ale predsa len som vyčlenil hlavnú činnosť do vlákna, aby sa pri väčšom súbore nezaseklo GUI. RTTI a vhniezdené typy som využil pri programovaní GUI.