**Криптиране на банкови карти с RMI**

Автор: Димитър Панайотов

Проектът предоставя възможността за криптиране на банкови карти посредством Remote Method Invocation (RMI). Всеки потребител, влязъл в системата, с изключение на тези без нужните права, може да криптира и декриптира номер на карта.

* **Създаване на сървъра и клиента**

Сървърът се стартира като се създаде регистър на локален порт. Той ще приема и обработва заявките на клиентите. Клиент се свързва към сървъра като вземе stub registry и по него се открие Remote обектът, който отговаря за операциите, които клиентът може да изпълнява.

Когато сървърът се стартира се извършва проверка за несъществуващ или празен файл, който ще съхранява потребителските данни (име, парола, ниво на достъп и уникален потребителски ключ). При несъществуващ такъв се създава файл по подразбиране, наречен users.xml в директорията, от която е стартирано приложението, с единствен потребител - admin.

При стартиране на клиента на потребителя се представя екран за вход или регистрация с потребителско име и парола.

* **Функционалност на сървъра**

При стартиране на сървъра се показва прозорец, предоставящ информация за състоянието на сървъра (работещ или не) и два бутона – единият за смяна на състоянието на сървъра, другият за съхраняване на номера на банкови карти.

Съхраняването на банкови карти се осъществява по два начина – сортирани по номерата на картите или по криптираните номера. При избиране на един от двата начина се отваря диалогов прозорец, предоставящ възможност на потребителя да избере име на файла, в който да бъдат съхранени банковите карти. В случай, че е избран вече съществуващ файл той може да бъде презаписан или не.

* **Функционалност на клиента**
  + Вход и регистрация

При стартиране на клиента на потребителя се показва прозорец, чрез който той може да влезе в системата с потребителското си име и парола или да се регистрира съответно с избрани такива. В случай, че е избрал да се логне се проверява дали името и паролата отговарят на изискванията, дали съществува в записите с потребители и ако съществува се сравнява въведена парола с тази от записа. При съвпадение се преминава към потребителския екран, а при некоректно въведени данни се показва прозорец описващ съответната грешка.

* + Потребителски екран

В зависимост от правата му, на потребителя се представят различни екрани. В случай, че той е обикновен потребител, екранът предоставя единствено възможност за криптиране и декриптиране на банкови карти. Ако има права на администратор се показва администраторски панел със същите възможности като на обикновения потребител плюс възможността за добавяне и премахване на други потребители.

* **Изключения**
  + AlreadyTakenException – при регистрация е избрано потребителско име, което вече е заето от друг потребител
  + InvalidCardNumberException – въведеният номер на банкова карта за криптиране не отговаря на едно от изискванията – да започва с 3,4,5,6 и да удовлетворява формулата на Luhn за валидност на номер на кредитна карта
  + InvalidCredentialsException – въведеното потребителско име и/или парола не отговарят на изискванията. Името може да съдържа само малки букви, цифри, символите ‘\_’ и ‘-‘ и трябва да бъде с дължина между 3 и 16 символа. Паролата може да съдържа същите тези символи, но дължината ѝ трябва да бъде между 5 и 20 символа. Изключението също се хвърля и при добавяне на потребител с невалидни права, различни от Administrator, User, Regular.
  + NoPermissionException – при права на потребителя забраняващи му да извършва дадено действие (например криптиране на банкова карта от потребител без това право)
  + Обработка на изключенията – в зависимост от значението на изключението за продължаване работата на програмата се избира дали да бъде показано съобщение за грешка, информиращо потребителя за случилото се, или да бъде показан детайлен диалогов прозорец съдържащ съобщение и stacktrace.
* **Файлове**

Форматът на файловете използван в програмата е .xml   
Потребителите се съхраняват във файл с наименование users.xml със следната структура:

<?xml version="1.0"?><users><user><username>потребителско име</username><password>парола</password><permission>ниво на потребителя </permission><KEY>уникален идентификационен номер</KEY></user>...</users>

Банковите карти със съответните им криптирани номера се съхраняват в xml файл с име избрано от потребителя със следната структура:

<?xml version="1.0" ?><root><entry><string>некриптиран номер на банкова карта</string><list><string>криптиран номер 1</string><string>криптиран номер 2</string>...</list></entry>…</root>

* **Използвани библиотеки**

XStream – всички операции с файлове използват тази библиотека като, за да удовлетворява нуждите на програмата са дописани методи за запис и четене на потребители от файл (обект от тип Users съдържащ списък от потребители) и запис и четене на карти от файл (обект от тип Map<String, List<String>>).

* **Имплементация**
  + Application – Предоставя операциите са стартиране, спиране и достъп до сървъра. Също така и стартирането на клиента.
  + CreditCard – Имплементира CardValidation. Помощен клас за валидация на номер на карта. При задаване на номер на карта се форматира по подходящ начин, за да бъде валидиран.
  + XStreamConverter – Дава достъп до операциите нужни за запис и четене на xml файлове с потребители и банкови карти.
  + Client – Показва екрана за вход и регистрация на потребители. Пуска нишка, която през определено време проверява дали има връзка със сървъра.
  + StartClient – Стартира клиента като достъпва Application.startClient() и задейства функционалността на Client.
  + Server – Съдържа главната логика на приложението. Всички действия на клиента биват обработвани от сървъра.
    - getAllUsers() – връща всички потребители, съдържащи се във файлът зададен от Application.FILE\_NAME
    - hasEntry(String username) – ако има потребител с потребителско име като това зададено от параметъра го връща, иначе връща null
    - saveUsers(Users users) – записва потребителите зададени като параметър във файла зададен от Application.FILE\_NAME
    - encrypt(String key, String number, int displacement) – използва уникалния ключ на потребителя (key), за да криптира номер на банкова карта (number). За всяко криптиране след първото добавя отместване (displacement) с 1 над текущия символ в ключа. Добавя картата в Map<String, List<String>> от карти и съответните им криптирани номера и увеличава броят на криптиранията на картата в Map<String, Integer>
    - decrypt(String number) – Проверява в Map-a на криптираните карти дали даденият номер отговаря на вече криптирана карта. Ако да го връща, иначе – използва стандартно отместване, за да декриптира картата и я добавя в Map-a.
    - storeCard(int sortingOption) – Запазва всички карти от Мап-а на криптираните карти в избран от потребителя файл със сортиране в зависимост от параметъра (0 = запазване на картите сортирани по номер на банковите карти; 1 = запазване на картите сортирани по номер на криптираните номера)
  + StartServer – Стартира сървъра като зарежда инстанция на регистър на Application.PORT и закача remote инстанция на обектът, който ще бъде използван за изпълняване на функциите на клиента.
  + Permission – Enumerable клас, описващ нивата на достъп на всеки потребител. Administrator – encrypt + decrypt + add / remove user; User – encrypt / decrypt; None – няма права да изпълнява никакви действия
  + User – Потребител с име, парола, ниво на достъп и 16-знаков alphanumerical уникален идентификационен номер, генериран чрез randomUUID, който се използва за криптирането на номер на банкова карта.
  + Users – Колекция от потребители за по-лесно запазване и четене от файл.
  + AdminScreen – Само администратор има достъп до него. Съдържа панел за добавяне на потребител, премахване на потребител и криптиране на банкова карта. При добавяне на потребител данните се валидират и ако са коректни се добавя нов потребител във файла users.xml. При изтриване се избира един потребител, различен от текущия, който да бъде изтрит, изтрива се и потребителите се презареждат в таблицата.
  + UserScreen – Общ за потребителите и администраторите. Съдържа единствено опцията за криптиране и декриптиране на номера на карти. При криптиране картата се валидира и ако номерът е коректен се връща криптиран номер. При декриптиране първо се проверява дали този номер не отговаря на карта. Ако е така се връща номера на картата, иначе се изпълнява декриптиране със стандартно отместване на символите.