Course Project No. 4
Обектно Ориентирано Програмиране
(ООП с Java)
Криптиране на банкови карти с RMI

Acceptable Programming Languages:	**Java**
Deadline:	**Февруари,** ***(on the final exam date)***
Instructor	**Dr. Evgeny Krustev**

Problem Statement:
С оглед защита на информационни системи, използващи номера на банкови карти, да се реализира система, предоставяща възможност за криптиране на данни **[1].** Криптирането предоставя възможност за съхранение на данни в неявен вид и повишава нивото на сигурност на данните и организацията. За **криптиране** обикновено се използват алгоритми от групата на „ Transposition Cipher" http://en.wikipedia.org/wiki/Transposition_cipher или от групата на „Substitution cipher" http://en.wikipedia.org/wiki/Substitution_cipher , при които се извършва отместване на символи на определено място в криптирания текст. Например, при „Substitution cipher" отместване с 5 на даден 16 цифров номер на карта се криптира така:

Номер на карта: 4563 9601 2200 1999
Криптограма: 9018 4156 7755 6444

Системата трябва да реализира и **логика за контрол на достъп**. Потребителите на системата могат да:

1. Криптират номер на банкова карта- задава номер на карта и получава като резултат криптираната стойност;
2. Извличат номер на карта по криптограма – задава криптограма и получава номер на карта.

Системата да се реализира като **RMI приложение**.
За **сървъра графичен потребителски интерфейс със следните изисквания**:

1. Графичният потребителски интерфейс да се реализира с **графична потребителска компонента от JavaFX, която да се използва многократно като Jar файл**.
2. Съхранява информацията за потребители (потребителско име и парола) и техните права в в XML (http://xstream.codehaus.org/);
3. При обръщение от клиент първо изчаква за потребителско име и парола, след което ги проверява за коректност. Ако са некоректни, затваря socket-а;
4. Ако потребител, нямащ права да криптира карти, се опита да го направи, връща грешка; Грешка се издава също при повече от 12 опита за криптиране на вече криптирана карта или при подаване на невалиден номер на кредитна карта т.е.- номерът на картата не започва с 3, 4, 5, 6 и не удовлетворява **формулата на Luhn** за валидност на номер на кредитна карта). При всеки следващ опит за криптиране на вече криптирана кредитна карта да се избере отместване на символите като се увеличава с единица стандартното избрано отместване на символите и така полученото число да се раздели по модул на 16. .В случай, че няма грешка в зададения номер на карта, да се криптира номера на картата и този номер се връща на клиента (криптира)

5. Ако потребител, нямащ права да изисква номер на карта, се опита да го направи, връща грешка; В противен случай, връща номера на картата съответстващ на подадената криптограма (декриптира)
6. Позволява извеждане в текстов файл на таблица на криптираните номера и съответните им банкови карти, сортирана по криптираните номера (една банкова карта може да има няколко криптирани номера)
7. Позволява извеждане в текстов файл на таблица на криптираните номера и съответните им банкови карти, сортирана по банковите карти (една банкова карта може да има няколко криптираните номера)- **да се използват класове от пакета NIO и Stream API**

Забележка: Изборът на конкретен метод за криптиране от групата „ Transposition Cipher" или „Substitution cipher" е по желание на студента. Реализацията на по- сложен метод от тези групи се счита на бонус.

За **клиента графичен потребителски интерфейс със следните изисквания**:

1. Отваря сокет към сървъра като подава потребителско име и парола (през **JavaFX** interface). Администраторът на сървъра може да добавя нови потребители със съответни права за достъп.
2. Предоставя графичен интерактивен интерфейс за криптиране на карта и извличане на номер на карта;
3. Коректно визуализира резултати и грешки.
4. Потребителският вход да се валидира с регулярни изрази

Evaluation:

Your project will be evaluated on the following **general points**:

- **Sophistication/complexity/originality** of the problem being solved/investigated and of the **solution(s)/approaches** considered.
- **Demonstrated ability to extract/analyze** concurrency-related problems/issues from a general problem/area of interest.
- **Clarity of explanations, and for implementations programming skill/quality**. Your report **(in Bulgarian!)** should be well written and free of grammatical and spelling errors. **Programs** must be **well-commented and in a professional style**.
- **Awareness of related work.** Others have considered the same or similar problems before you. Your work does not have to be novel, but you should be able to contextualize your approach. Be sure to explain how each referenced work is *related* to your work. Note that a 5-minute *Google* search will not be adequate; if you are unfamiliar with the required textbooks for the course:
- **Completeness** of the project.

Deliverables: *The files with* :

1. the source code
2. the executable code

3. the instructions for compiling your source code
4. the report explaining the data structures and the algorithm implementation, describe things such as how your code has been tested, limitations of your code, problems encountered, and problems remaining
5. any files used to test the implementation of the program with an explanation about it included in the report.

References:

[1] How Security Can Help Grow Your Business: The Marketing Side of Tokenization, http://merch.bankofamerica.com/documents/10162/50824/How+Security+Can+Help+Grow+Your+Business.pdf

[2] **Текстове, използвани в курса по време на лекции и практически занятия по ООП с Java**