Course Project No. 1
Обектно Ориентирано Програмиране
(ООП с Java)
Токенизация на банкови карти с многонишков сървър

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 „***токенизация***“.[1- 2] Токенизацията представлява изоморфно изображение на номер на **16 цифров номер на карта** към **число**. Например:

Номер на карта: 4563960122019991
Токен: 1234243434269991

Да се реализира **токенизация на зададен номер на банкова карта** при следните изисквания:

- броят на цифрите на токена и на номера на банковата сметка да е еднакъв
- последните 4 цифри на токена и на номера на банковата карта да съвпадат
- първите 12 цифри на токена да са произволно генерирани и да не съвпадат със съответните цифри от номера на банковата карта
- първата цифра на токена да е различна от цифрите 3, 4, 5, 6, които се използват от основните брандове на банкови карти
- сумата от цифрите на токена не трябва да е кратна на10

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

1. Регистрират токен- задава номер на карта и получава като резултат дали е регистриран токен т.е извършва се токенизация на банковата карта. Издава се съобщение за грешка, ако не е регистриран токен (проверява се дали номерът на картата е валиден т.е. дали започва с 3, 4, 5, 6 и удовлетворява **формулата на Luhn** за валидност на номер на кредитна карта). Ако номерът на кредитната карта е валиден, потребителят получава новосъздадения токен.
2. Извличат номер на карта по токен- задава токен и получава номер на карта. Издава се съобщение за грешка, ако не е регистриран токен

Системата да се реализира като приложение на **многонишков сървър- клиент**.
За **сървъра графичен потребителски интерфейс със следните изисквания**:

1. Графичният потребителски интерфейс да се реализира като **графична потребителска компонента от JavaFX, която да се използва многократно като Jar файл**
2. Съхранява съотношението номер на карта <-> токен в XML сериализация (http://xstream.codehaus.org/);
3. Съхранява информацията за потребители (потребителско име и парола) и техните права в XML сериализация (http://xstream.codehaus.org/);
4. При регистрация на токен се извършва токенизация на банковата карта.. Създаденият токен трябва да е с уникална (неповтаряща се) стойност измежду всички токени. Проверява се дали номерът на картата е валиден т.е. дали започва с 3, 4, 5, 6 и удовлетворява **формулата на Luhn** за валидност на номер на

кредитна карта) Ако е не е регистриран токен, връща false, в противен случай – връща true;
5. При обръщение от клиент първо изчаква за потребителско име и парола, след което ги проверява за коректност. Ако са некоректни връща грешка;
6. Ако потребител, нямащ права да регистрира токени, се опита да го направи, връща грешка;
7. Ако потребител, нямащ права да изисква номер на карта, се опита да го направи, връща грешка;
8. Позволява извеждане в текстов файл на таблица на токените и съответните им банкови карти, сортирана по токените (една банкова карта може да има няколко токена)
9. Позволява извеждане в текстов файл на таблица на токените и съответните им банкови карти, сортирана по банковите карти (една банкова карта може да има няколко токена)- **)- да се използват класове от пакета NIO и Stream API**

За **клиента графичен потребителски интерфейс със следните изисквания**:
1. Отваря сокет към сървъра като подава потребителско име и парола (през JavaFX interface);
2. Извиква опции за регистрация на токен 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] Implementing Tokenization Is Simpler Than You Think, http://www.firstdata.com/en_us/insights/implementing-tokenization-white-paper

[2] 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

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