Система за криптиране на банкови карти с многонишков сървър

Системата реализира **многонишков сървър** и **клиентска част**. Комуникацията между тях се осъществява чрез сокети.
Системата позволява потребителите да *криптират номер* на дадена кредитна карта(като това се случва по зададен алгоритъм) или да изискват номер на дадена карта по кода й(*декриптират*). Нови потребители могат да бъдат създавани само от администраторите.

- Класът **User** представя потребител с *име*, *парола* и *правото му на достъп*(клас Access).

- Класът **Access** представя право на достъп - дали *е администратор*, дали *може да криптира номер на карта*, или *да декриптира карта*.

- Класът **UserDataBase** представлява съвкупност(*ArrayList<User>*) от всички потребители(*симулира база от данни*).

- Класът **UserManager**

Този клас е *статичен** и улеснява работата с "*база данни*" от потребители(клас *UserDataBase*) предоставяйки следните функционалности:

1) *Извлича потребител по дадени име и парола*
2) *Проверява дали съществува потребител по дадено име*
3) *Създава нов потребител*
4) *Запазва текущата база от данни в файл(XML форматирана с XStream)*

UserManager съдържа 2 полета - поле от клас *UserDataBase* и *файл*. Инициализира се(*initiazlize(String filename)*) с име на файл, които съдържа xml записи на потребители(използва се XStream, за да се извлекат)******. Тези потребители се заделят в базата данни от потребители(полето *UserDataBase*).

- Класът **CreditCard** представя кредитна карта като съвкупност(*ArrayList<String>*) от кодовете й.

- Класът **CreditCardDataBase** представлява съвкупност(*ArrayList<CreditCard>*) от всички карти(*симулира база от данни*).

- Класът **CreditCardManager**

Този клас е *статичен** и улеснява работата с "*база данни*" от кредитни карти(клас *CreditCardDataBase*) предоставяйки следните функционалности:

1) *Проверява даден номер дали е валиден номер за кредитна карта(според алгоритъма на Luhn).*
2) *Проверява дали карта с даден номер или код съществува в базата данни.*
3) *Извлича кредитна карта от базата данни по даден номер или код.*
4) *Запазва текущата база в файл(XML форматирана с XStream)*
5) *Да криптира или декриптира дадена карта.*
6) *Да извежда картите в текстов файл(табулирано) сортирани по номер или код.*

CreditCardManager съдържа 3 полета - поле от клас *CreditCardDataBase*, *файл*, *алгоритъм* по които да се криптират картите(имплементира *interface CryptAlgorithm*). Инициализира се(initiazlize(String filename, CryptAlgorithm algorithm)) с име на файл, които съдържа xml записи на карти(изпозлва се XStream, за да се извлекат)******. Тези карти се заделят в *базата данни* от карти(полето *CreditCardDataBase*).

- **interface CryptAlgorithm** има 3 функции:

getStandartOffset() - връща стандартното отместване на алгоритъма.
encrypt(String creditCardNumber, int offset) - криптира номер на кредитна карта по дадено отместване.
decrypt(String creditCardCode, int offset) - декриптира код на кредитна карта по дадено отместване.

- Клас **Algorithm** служи за да се имплементират различни алгоритми в него.(*closure, callback конструкция*).

- Клас **Client** изгражда *графичен интефейс за клиента*, чрез които може лесно до комуникира(да праща заявки към сървъра за *логване*, *криптиране*, *декриптиране*, *създаване на нови потребители*) със сървъра. Включва и *валидация* на потребителския вход.

- Класът **ThreadPooledServer** изгражда *многонишков* сървър. Той се грижи за това всеки клиент да се "обслужи" в отделна нишка от клас*ClientHandlerRunnable*. (бонус: thread-safe)

- Класът **ClientHandlerRunnable** осигурява комуникацията между клиента и сървъра. Използвайки функционалностите на мениджърите(*UserManager* и *CreditCardManager*), *ClientHandlerRunnable* осъществява *логването* и *създаването* на потребители, *криптирането* ,и *декриптирането* на кредитни карти.

- Класът **GraphicServer** изгражда *графичен интефейс за ThreadPooledServer-и*.

Предоставя *лог*(какво се случва със сървъра)(бонус) и възможност да *извежда кредитните карти в текстов файл*(табулирано) *сортирани по номер или код* с помощта на *CreditCardManager*(трябва да се инициализира преди да се стартира сървъра(бонус може много сървъри)).

*статични класове в Java няма, но може да се симулират като се дефинира final клас с private конструктор и само статични методи.
**може и празен файл(или несъществуващ)

*Използвана литература:
- Javadoc
- XStream doc (http://xstream.codehaus.org/tutorial.html)
- http://stackoverflow.com/