**Course Project N4**

**Обектно Ориентирано Програмиране  
(OOП с Java)**

**Криптиране на банкови карти с RMI**

Проекта се състои от сървърна част и клиентска част. Сървърната част отговаря за всички дейности по криптиране и декриптиране на банкови карти, добавяне и премахване на потербители, проверка за коректност на потребителско име, парола, номер на банкова карта и следи всеки потребител да не може да извършва дейности, за които не притежава права.

Клиента предоставя възможност за връзка със сървъра и извършване на криптиране и декриптиране на номера на банкови карти, както и добавяне на потребители със съответните права. Грижи се за визуализиране на възникнали грешки и изключения по подходящ начин - визуализиране на прозорец, който уведомява потребителя за възникналия проблем.

Класовете в проекта са разделени в шест пакета. В пакета bg.uni\_sofia.fmi.oopjava.client се намира клас, който намира отдалечения обект и също така съдържа методите, които трябва да може да извиква клиента използвайки отдалечения обект. В bg.uni\_sofia.fmi.oopjava.enums се намира enum, който съдържа видовете права на потребителите. В bg.uni\_sofia.fmi.oopjava.exceptions са всички специфични изключения, които хвърлят методите на клиента и сървъра. В bg.uni\_sofia.fmi.oopjava.server се намира Remote Interface, неговата имплементация и клас с методи за регистриране и премахване на обекта от регистъра.В bg.uni\_sofia.fmi.oopjava.utils се намира класът, който описва потребител, клас който съдържа методи за запис и четене на XML файлове и клас с методи за записване на номерата на картите. Пакетът bg.uni\_sofia.fmi.oopjava.windows съдържа всички панели и прозорци на приложението.

За криптиране е използван „Substitution cipher”, които прибавя едно и също число към всяка цифра от номера на банковата карта. По време на работата на сървъра при всяко криптиране на номер на карта, криптограмата се запазва в TreeMap<String, ArrayList<String>>. Ключове са номерата на банковите карти, представени като символни низове, а стойности са ArrayList<String>, тъй като всеки номер на карта може да има по повече от една криптограми. Номерата и криптограмите могат да се записват в текстови файл като за сортиране се използва свойството на TreeMap да държи двойките <ключ, стойност> сортирани по ключ. За сортиране по криптограми се съставя нов TreeMap<String, ArrayList<String>>, в който ключове са криптограмите, а стойности са съответните им номера на карти. За запис във файл се използва класът Formatter.

За декриптиране на криптограма, която не е била криптирана преди това и не присъства в TreeMap<String, ArrayList<String>> се използва стандартното отместване. В противен случай се обхожда структурата от данни, която пази номерата и съответните им криптограми и се извежда номера съответстващ на криптограмата.

Контролът на потребителския вход се извършва чрез регулярни изрази. За потребителско име и парола се приемат символни низове състоящи се от малки и големи латински букви, цифрите от 0 до 9 и знакът '\_'. За сравнение с регулярния израз се използва методът matches() на класът String.

Данните за всеки потребител се пазят в отделен XML файл, чието име съвпада с потребителското име. Файлът се намира в папката на проекта. Данните се съдържат в клас User и се състоят от потребителско име, парола и масив от изброимия тип Permissions за описване на правата на потребителя. Данните се записват и четат от XML файла с помощта на два статични метода, които използват библиотеката XStream.

За тестване на кодът е създаден предварително XML файла на администратора, тъй като само той има права да добавя нови потребители. С този потребител (потребителско име: admin, парола: admin) е тествана валидацията на потребителския вход и изход с регулярни изрази и добавянето и премахването на потребители. За тестване на криптиране и декриптиране на номера на карти е използван примерния номер на карта от заданието на проекта и негови вариации. Използвана е формулата на Luhn за проверка за валидност на номер на карта. Тествани са и проверките за грешки чрез въвеждане на некоректни данни - несъществуващи потребители или такива с грешни пароли, невалидни номера на карти, опит за извършване на криптиране/декриптиране или добавяне/премахване на потребител от потребител без нужните права.

При тестване първо се стартира класа ServerWindow, като сървърът се стартира автоматично. След това се стартира класът LogInWindow, който дава възможност да се въведат потребителско име и парола, които за нуждите на теста са admin/admin, jack/jack, georgi/georgi, petar/petar.