

# “Майже В МММ”

Marie Maimeskul  
Michael Kobelev  
Vadym Kochmar  
Maryna Tiutiun

## Terms of reference

September 25, 2019

## Overview

The system being developed is a bank system, capable of performing different operations with *money* (here and next, by money meaning some measure units that have equivalent in sheets of paper, design of paper is to be decided). The main components of the system - is a server that processes all transactions and clients - *ATMs* (real or virtual clients that perform operations). For the simplicity of project - software for bank branches is not developed and all users are assumed to already have active accounts within the system and cards for them. Also for simplicity, only one ATM will be developed.

## Users

The users of the system are current clients of the bank and hold active accounts and active cards to access them. The card contains QR code with card details and card number.

(The design is a subject for changes):



## Domain knowledge (banking)

### *Accounts*

There are three types of accounts: checking, deposit and credit.

**Checking**: simple account from which cash can be withdrawn easily.

**Deposit**: a client can open a deposit for a certain sum and in case the money is withdrawn before the term, all benefits are discarded and there's commission for that 1%. For simplicity, only 1-year accounts are available.

**Credit**: a client can open a credit account with credit limit. The cash can be withdrawn and replenished easily, though if dept is not payed till the end of the next month (i.e. subsequent to the month, when the limit was withdrawn). For example, a client has 2500 units on his credit account (1500 his money + 1000 limit), then he spends 2000 units in June (-500 his money + 1000 limit). He has to replenish the account to at least 500 units till the 31st August (inclusively). If not, every day of delay is 1% of the negative balance(in this situation 5 units a day). Also, a client cannot have negative overall credit account.

## *Cards*

A card - is identifier of the account and user. Forgotten pin code makes the account inaccessible (made for simplicity).

A card represents one and only one account. Any type of account can possess its own physical card.

## Functional requirements

1. **PIN-code**: Users can change their PIN-codes to new. For simplicity, if PIN is forgotten - there is no access to the card (meaning that the client needs to apply to office branch).
2. **Balance**: users can view the balance on the account.
3. **Cash withdrawal**: if possible, users can withdraw cash money.
4. **Transfer**: if possible, users can transfer money to another account, provided the number of recipient's account is known.

## Technical requirements

1. **Server:** Server is a C# application that manages all the requests concerning operations with account. The server is connected to MS SqlServer relational database.
2. **Client:** is a physical machine (ATM), running Raspbian OS, virtually consists of two parts: Low-level library (written in Python, using system and C libraries), responsible for giving money and Java application that handles all interactive user flow.
3. **API:** Server and Client communicate with each other, using predefined REST API:

<i>Request</i>	<i>Response</i>
<code>POST /api/startSession/ { }</code>	<code>{ "Ok" - required bool "Errors" - not required list }</code>
<code>POST /api/login/ { "cardNum" - required string "pin" - required string }</code>	<code>{ "ok" - required bool "accessToken" - required string "errors" - not required list }</code>
<code>GET /api/balance/{Card_num} { }</code>	<code>{ "ok" - required bool "errors" - not required list "cardNum" - required string "balance" - required string }</code>
<code>PUT /api/changePin/ { "cardNum" - required string "oldPin" - required string "newPin" - required string }</code>	<code>{ "ok" - required bool "allowed" - required bool "errors" - not required list }</code>

<pre> }</pre>	<pre> }</pre>
<pre> POST /api/withdraw/ {   "cardNum" - required string   "amount" - required double }</pre>	<pre> {   "ok" - required bool   "allowed" - required bool   "txnId" - required int   "errors" - not required string }</pre>
<pre> POST /api/confirmWithdraw/ {   "cardNum" - required string   "txnId" - required string   "finished" - required string   "errors" - not required string }</pre>	<pre> {   "ok" - required bool   "errors" - not required list }</pre>
<pre> GET /api/cardExists/{cardNum}</pre>	<pre> {   "ok" - required bool   "errors" - not required list   "cardNum" - required string   "isValid" - required bool }</pre>
<pre> POST /api/transfer/ {   "cardNumFrom" - required string   "cardNumTo" - required string   "amount" - required double }</pre>	<pre> {   "ok" - required bool   "errors" - not required list }</pre>

4. **Security:** PIN-code is not known to (and can't be known by) the developers of the system. Once assigned, a pin number is hashed several times and stored in DB. When user enters PIN, it is hashed in the same manner and on the server only hashes are compared to each other.

5. **Issuing cash.** The cash issuing differs on virtual and real ATMs. In case of physical ATM (to be decided, whether it will be implemented), the machine can issue one banknote at a time. Because the hardware is not the main part of the project, the ATM can withdraw not the correct amount of money (but the correct amount will be withdrawn from the account). No more than 8 banknotes can be issued within one transaction. The information about the remaining number of banknotes is stored on server and when technical support replenishes the ATM he manually increases the balance on the server. In case of virtual ATM, issuing cash is not available.

## Areas of responsibility

Area	Subtask	Assignee	Deadline
Server	DB connection, DB structure (models) and API to make queries.	Marina Tiutun Vadym Kochmar	21.10.2019
	Implementing REST API. Configuring session and authentication flow.		21.10.2019
Client	GUI	Marie Maimeskul Michael Kobelev	28.10.2019
	Communicating with Server		28.10.2019
	Communicating with giving money machine	Michael Kobelev	28.10.2019
ATM	Low-level library for giving out money.	Michael Kobelev	22.09.2019
	Camera settings.		22.09.2019
	ATM box.	Marie Maimeskul	04.11.2019
System design	Terms of reference	Michael Kobelev	29.09.2019
	REST API structure		29.09.2019
	UML class diagrams	Marie	14.10.2019

	UML activity diagrams (user activity).	Maimeskul	14.10.2019
	ER model of domain knowledge and DB.	All together	29.09.2019

## Technical Documentation

### Diagrams

Diagram 1 - ER-model (Domain knowledge)

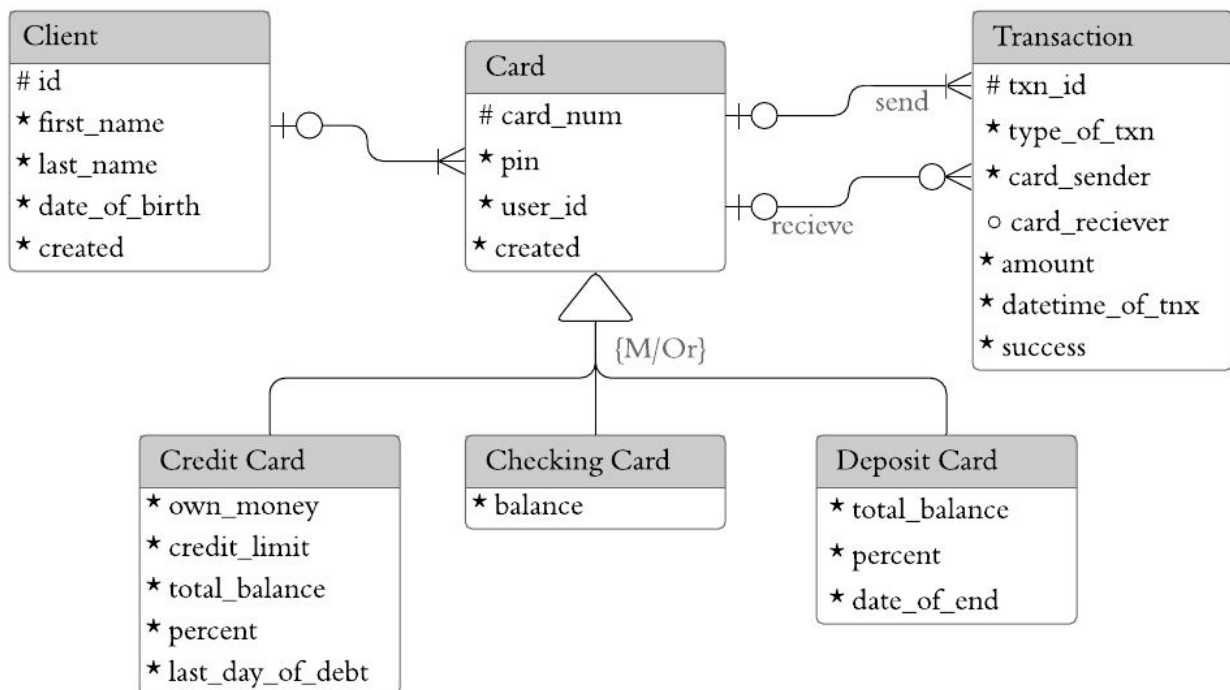


Diagram 2 - UML diagram of activity/user interaction

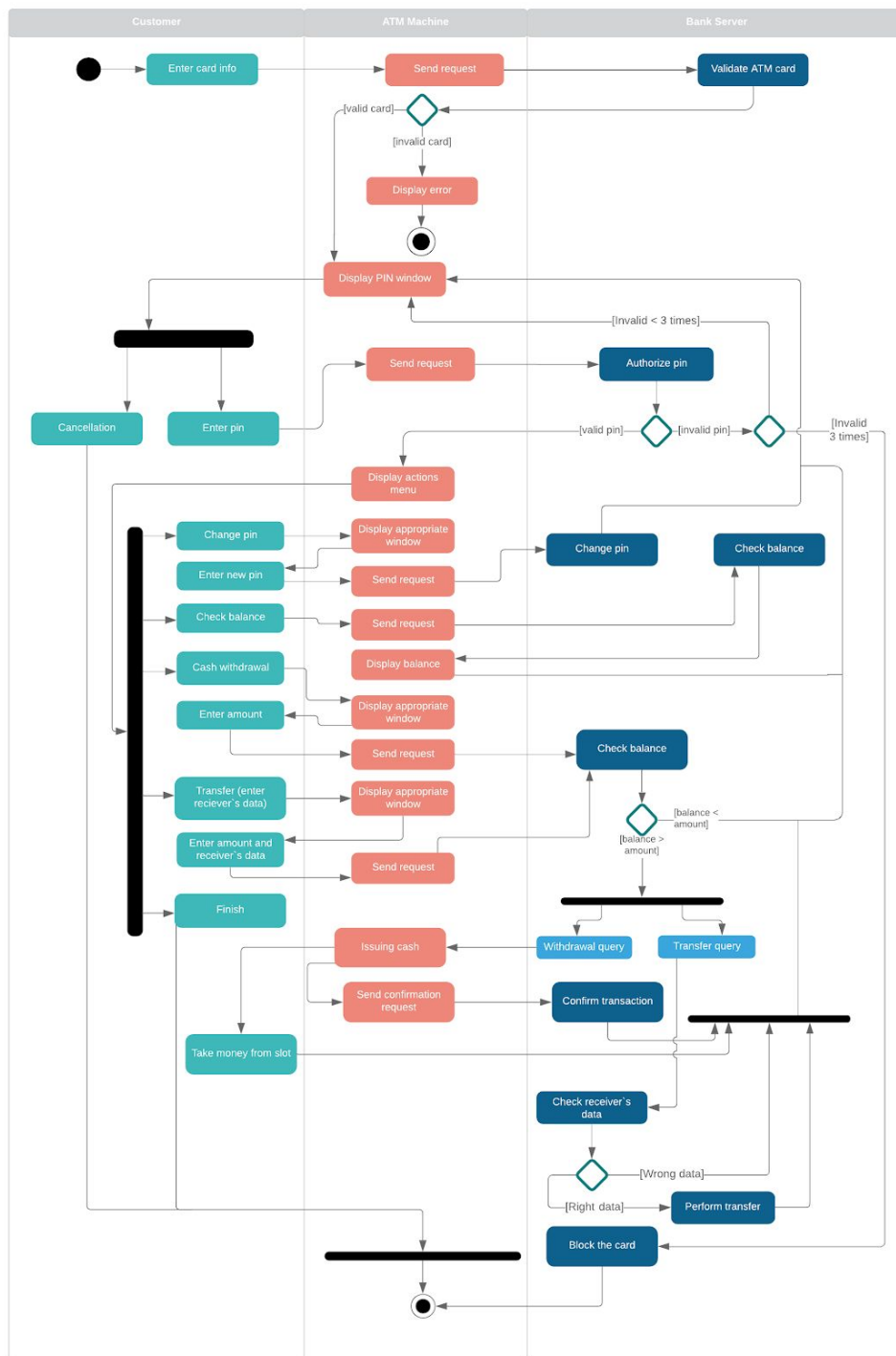




Diagram 3 - UML-diagram of classes on server

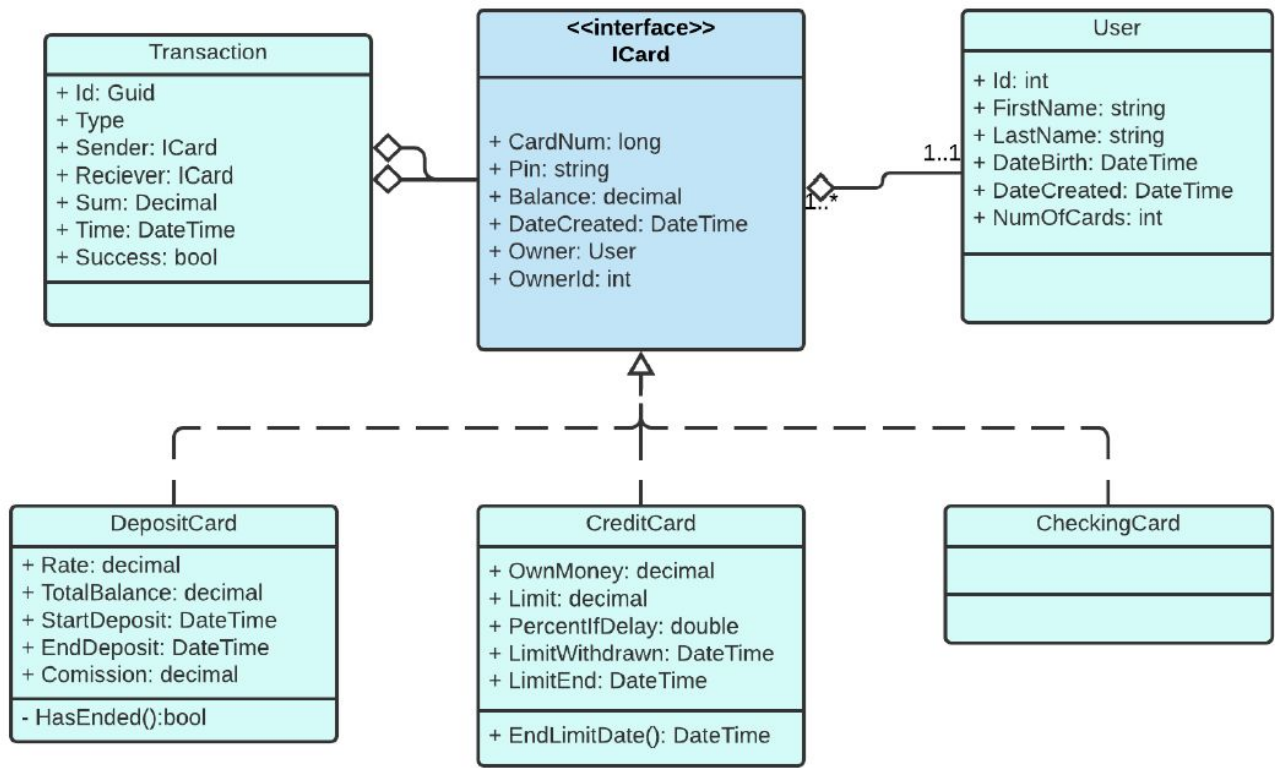
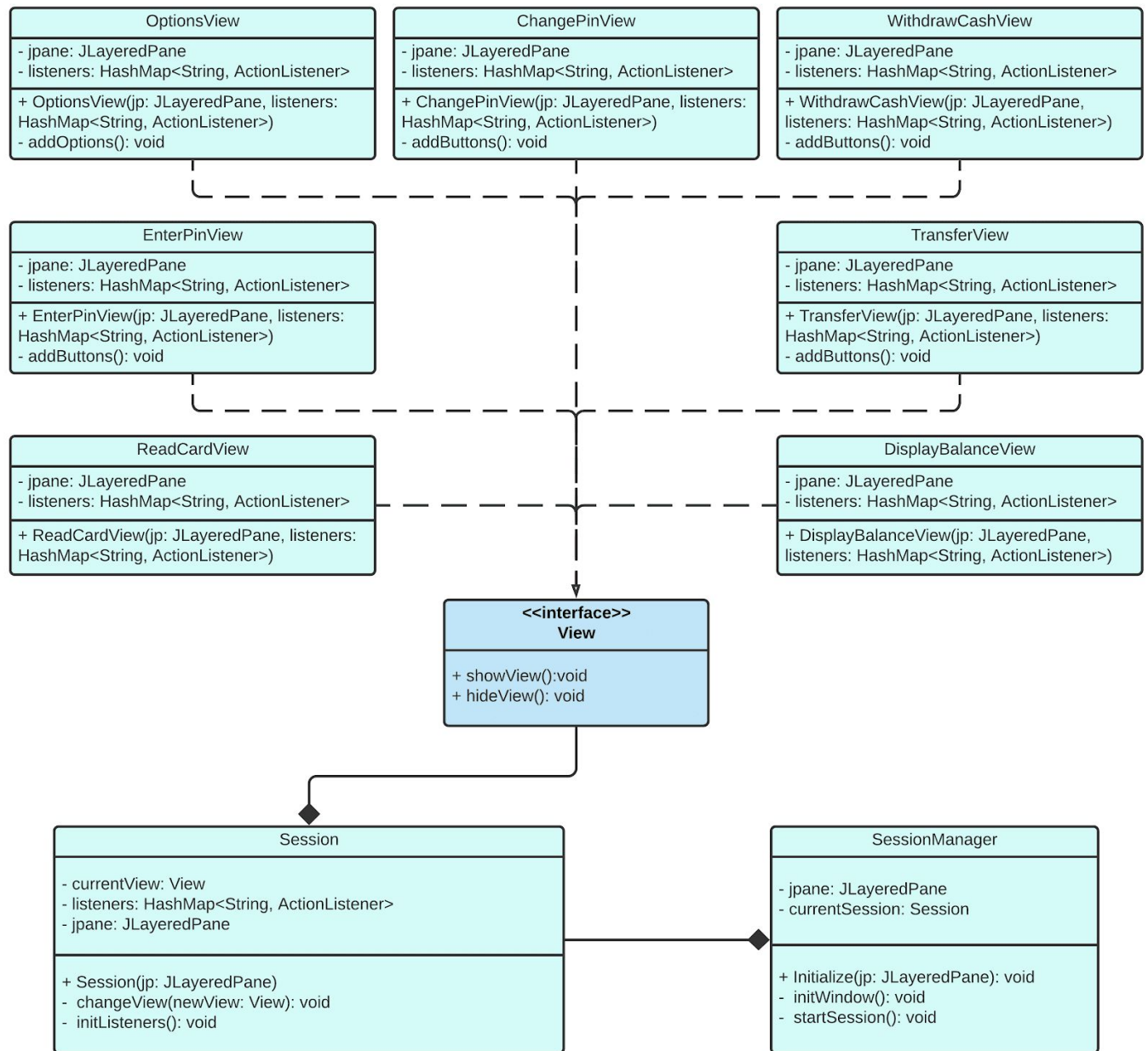
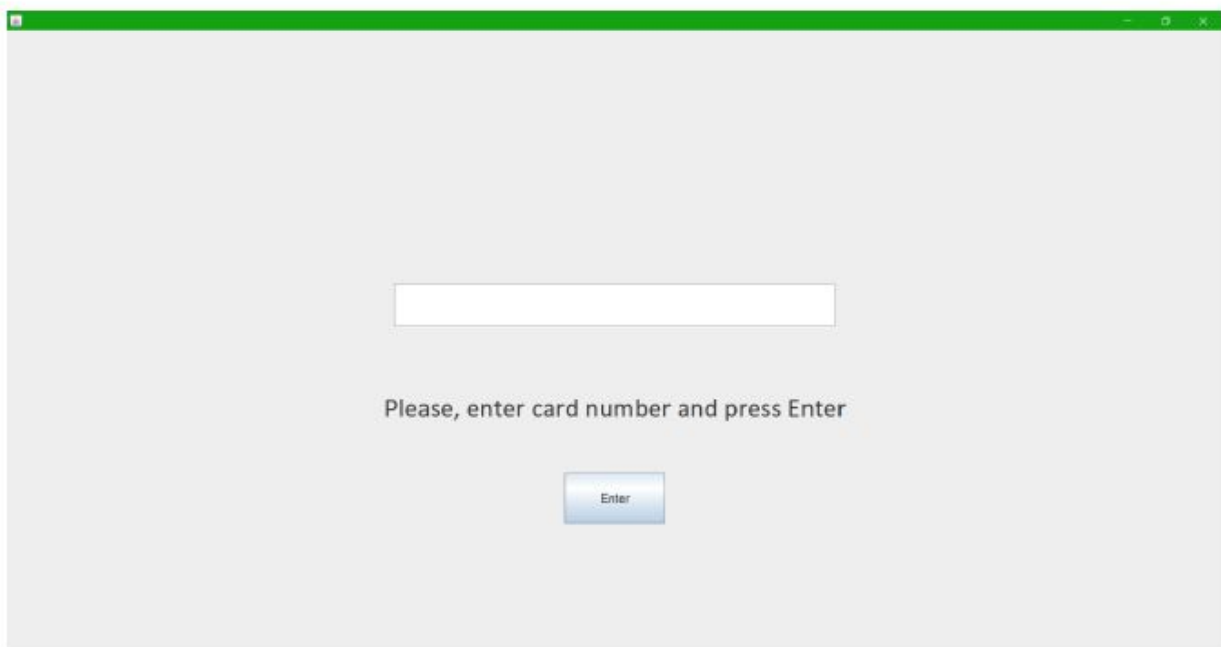
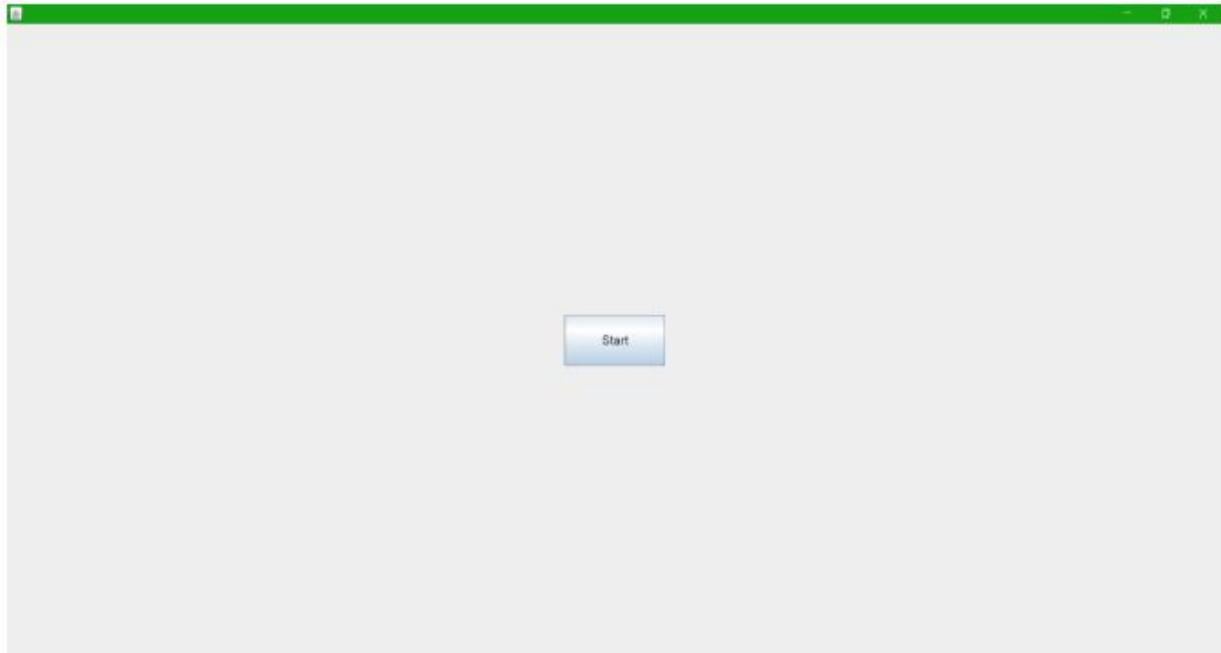
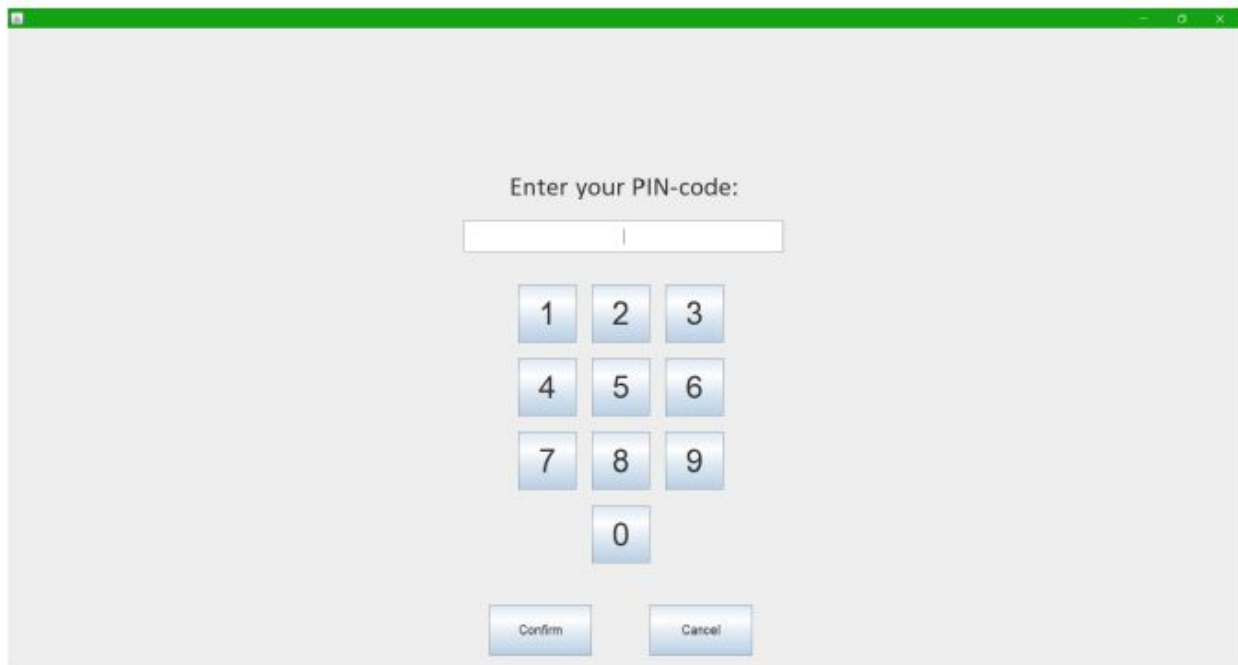


Diagram 4 - UML-diagram of classes of ATM



## Prototype screenshots



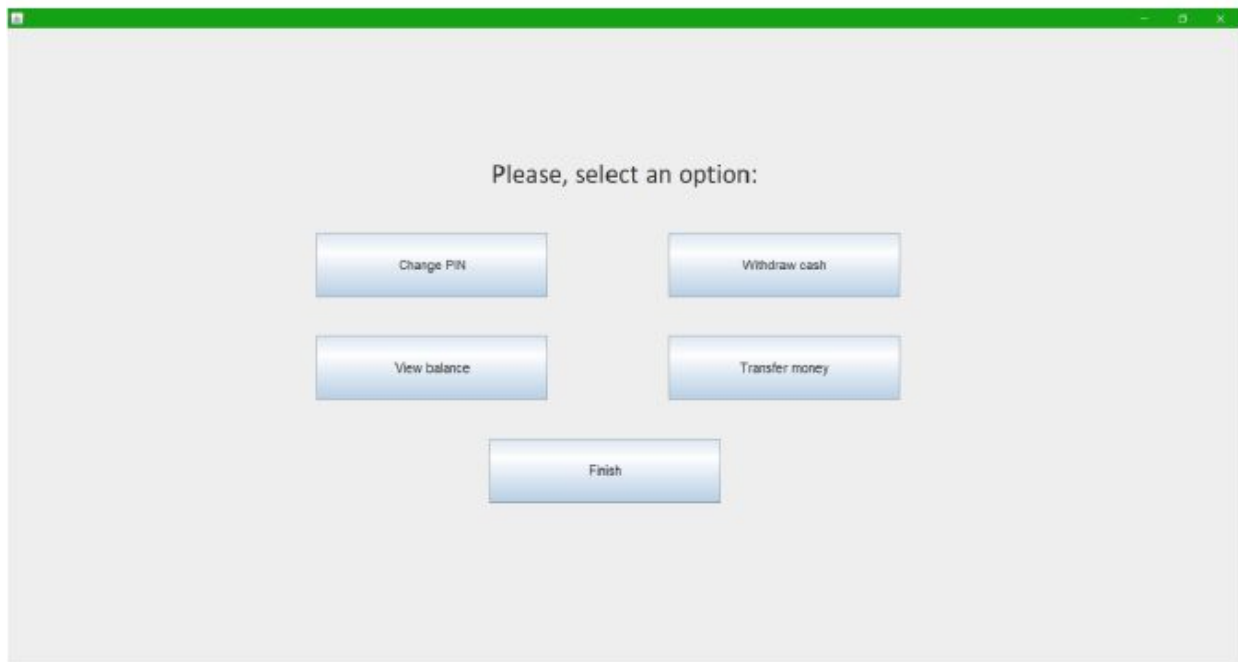


Enter your PIN-code:

1	2	3
4	5	6
7	8	9
0		

ConfirmCancel

This is a screenshot of a PIN entry screen. It features a title bar with a green background and standard window controls. The main area has a light gray background. At the top, the text "Enter your PIN-code:" is displayed. Below it is a single-line text input field. Underneath the input field is a numeric keypad with buttons for digits 1 through 9 and 0. The buttons are arranged in a 3x3 grid for digits 1-9, with the 0 button centered below the 7-9 row. At the bottom of the screen are two buttons labeled "Confirm" and "Cancel".



Please, select an option:

Change PINWithdraw cashView balanceTransfer moneyFinish

This is a screenshot of a main menu screen. It features a title bar with a green background and standard window controls. The main area has a light gray background. At the top, the text "Please, select an option:" is displayed. Below this text are five buttons arranged in a grid: "Change PIN" and "Withdraw cash" in the top row, "View balance" and "Transfer money" in the middle row, and "Finish" centered at the bottom. All buttons have a light blue gradient and a slight shadow effect.

## Testing API endpoints

### Checking balance

GET ▼ http://localhost:49905/api/balance/2

Params Authorization Headers (9) Body ● Pre-request Script Tests Se

Query Params

KEY	VALUE
Key	Value

Body Cookies Headers (5) Test Results

Pretty Raw Preview Visualize BETA JSON ▼

```
1 {
2   "ok": true,
3   "cardNum": 2,
4   "balance": 2.00
5 }
```

### Checking card for existence

GET ▼ http://localhost:49905/api/cardExists/1

Params Authorization Headers (9) Body ● Pre-request Script Te

Query Params

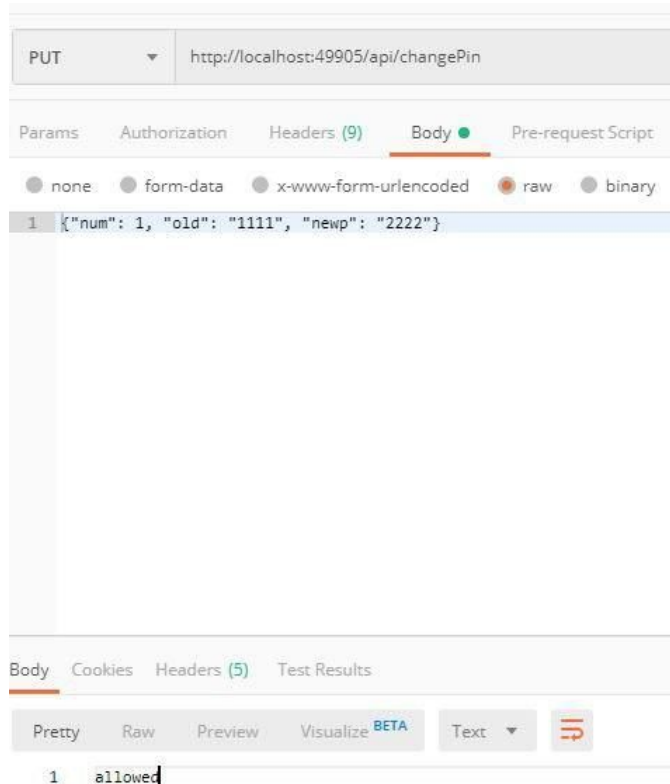
KEY	VALUE
Key	Value

Body Cookies Headers (5) Test Results

Pretty Raw Preview Visualize BETA JSON ▼

```
1 {
2   "ok": true,
3   "cardNum": 1,
4   "isValid": true
5 }
```

## Changing PIN-code



## Successful withdrawal

