

# Bank Simulator

- **Analysis of the topic**

The programme involves simulating the operation of a bank from the point of view of both the customer and the bank employee.

After starting the application, each user must log in. In the case of an employee, it is enough to enter the password, while a customer additionally has to enter a one-time password. If the customer enters the password incorrectly 5 times, the account is blocked and this fact must be reported to the bank in order to remove the blockade. After logging in, a session of 5 minutes begins, after which the client is automatically logged out if it is not extended.

After logging in, the client has access to his bank accounts and their information such as balance (in PLN, EUR and USD), transaction history, deposits, loans and cyclical transfers. He can buy and sell currencies at the exchange rate taken from the NBP website, change his password, generate a new list of one-time passwords, make transfers, order cyclical transfers and cancel them.

The bank's employees are categorised by authority on a scale of 1 - 4:

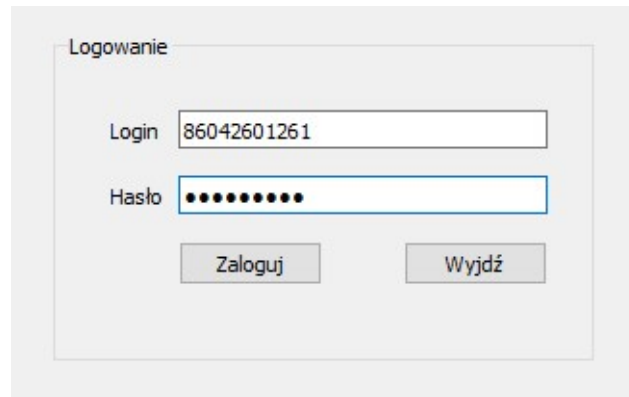
- 4 has access to: view the list of customers, change your password, grant and cancel loans,
- 3 has access to: view the list of customers, change your password, grant and cancel deposits,
- 2 has access to: viewing the list of customers and employees, changing your password, the ability to create new customers in the database and edit them, the ability to create accounts for customers and edit them, editing the customer's balance, unlocking blocked customers,
- 1 has all rights from group 1 - 3 and additionally: creation and editing of bank employees with rights.

After completing the operations, log out and close the application to save the changes.

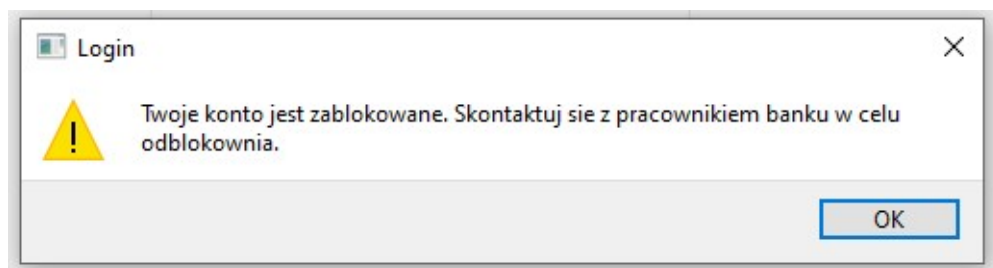
The application uses Qt libraries to implement the graphical interface. This library is also used to handle threads and perform web queries to obtain the current exchange rate. The rationale for the choice of classes and algorithms is included in section 3 *Internal specification*.

- **External specifications**

After starting the program, the first thing the user will see is the login window. The *Log in* button is used to log in and the *Exit* button is used to close the application properly, during which all changes made will be saved.



After pressing *Zaloguj*, if the user enters an incorrect password, he/she will receive an appropriate message. If the wrong password is entered five times, a blockade follows, which can be cancelled only by a bank employee.



If the login was successful, depending on whether a customer has logged in, another window opens.

- **Customer perspective**

In order to log in, the customer has to pass the second stage of verification by entering a one-time password. If the password is entered incorrectly five times, the customer will also be blocked as in the case of the first login stage

Logowanie

Wprowadź hasło jednorazowe nr 1

Zaloguj Anuluj

After successful login, the main window is displayed with the session time visible in the upper right corner. When this time reaches zero, the client is automatically logged out. The *Przedłuż sesję* button resets the time back to five minutes. The *Wyloguj* button is used to log out correctly.

Twoje konto

5 : 00 Przedłuż sesję Wyloguj

Konta

Wykonaj przelew

Wymień waluty

Historia

Lokaty

Kredyty

Przelewy cykliczne

Generuj listę haseł

Zmień hasło

Pressing one of the buttons: *Konta*, *Wykonaj Przelew*, *Wymień waluty*, *Historia*, *Lokaty*, *Kredyty*, *Przelewy cykliczne* will take you to the list for selecting the account for which the selected operation is to be performed. Double click on the account number.

Wybierz konto

45239220  
45239063

The *Konto* button is used to display general information about the selected account.

Informacje o koncie	
Saldo zł	269611
Saldo eur	380449
Saldo usd	0
Liczba lokat	1
Liczba przelewów cyklicznych	1
Liczba kredytów	1

The button *Wykonaj przelew* displays a window used to execute a transfer as well as cyclical transfers. If the customer does not have sufficient funds on the account, the transfer will not be executed. It is also not possible to make a transfer for a negative amount.

Nr konta

Tytuł

Treść 

Wiadomość ta ma na celu pokazać działanie historii operacji

Kwota

Jeśli przelew ma być cyklicznym, wpisz tu dzień miesiąca w którym ma być wykonywany

The *Wymień waluty* button allows you to exchange currencies according to the exchange rate downloaded from the NBP website. Double click on the line of interest.

	Waluta	Kupno	Sprzedaż
1	Dolar	3,7366	3,8120
2	Euro	4,5238	4,6152

You will be redirected to a box where you can enter the amount.

Euro

Kupno 4,5238      Sprzedaz 4,6152

Ile chcesz kupić

Ile chcesz sprzedać?

Summary of operations how much it will cost.

Zapłacisz	553,824000	zł za	120	eur
Otrzymasz	0,000000	zł za	0	eur

Zatwierdź

The *Historia* button displays the history of operations related to the selected account. Each change in the account balance results in a corresponding entry, e.g. a transfer sent or received.

0. 1.5.2021 Zmiana stanu konta: 120.000000eur  
1. 1.5.2021 Zmiana stanu konta: -553.824000zł  
2. 1.5.2021 Przelew wysłany: 120zł Adresat: 2939;  
3. 1.5.2021 Kredyt udzielono: 15000zł  
4. 1.5.2021 Lokata założono: -1000.000000zł  
5. 1.5.2021 Zmiana stanu konta: 380449.000000e

1.5.2021  
Przelew wysłany: 120zł Adresat: 2939201367 Tytuł: Demonstracyjny przelew Treść:  
Wiadomość ta ma na celu pokazać  
działanie historii operacji

The *Lokaty* button displays the current deposits in the selected account.

	Kwota na lokacie	Oprocentowanie	Data otwarcia lokaty	Data zamknięcia lokaty
1	1000	15	1.5.2021	15.12.2021

The *Kredyty* button displays information about the current credits for the selected account.

	Kwota kredytu	Oprocentowanie	Miesięczna rata	Data podpisania umowy	Data ostatniej wpłaconej raty	Data spłaty kredytu
1	15000	1.2	506.000000	1.5.2021	1.5.2021	21.11.2023

The *Przelewy cykliczne* button displays the current cyclical transfer orders. To cancel an order, double-click on the selected line.

	Kwota przelewu	Dzien miesiaca wykonywania	Nr odbiorcy	Tytul	Tresc
1	3.16	26	362732854	Przelew do instrukcji	Jak w tytule...

The *Generuj listę haseł* button will generate new one-time passwords and save them to a file whose name is the customer login.

The *Zmień hasło* button allows you to change your password.

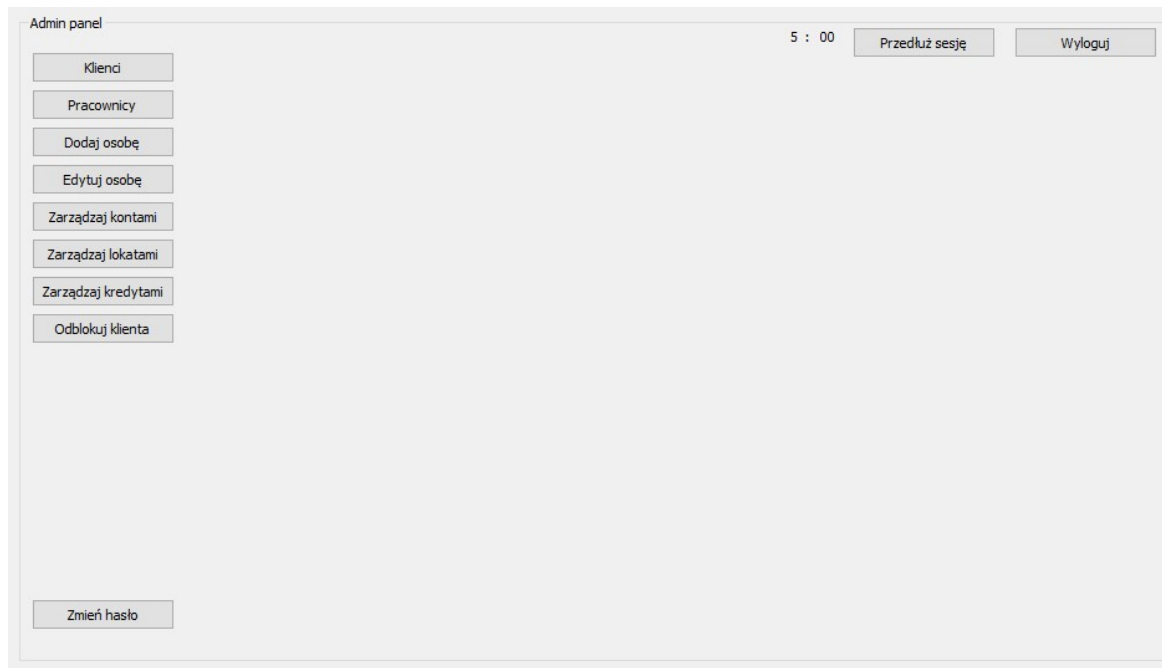
Podaj stare hasło	<input type="text"/>
Podaj nowe hasło	<input type="text"/>
Powtórz nowe hasło	<input type="text"/>
<input type="button" value="Zatwierdź"/>	

When you are finished, log out and close the application properly with the *Wyjdź* button to save your changes.

- The bank employee's perspective



After successful login, the main window is displayed with the session time visible in the upper right corner. When this time reaches zero, the client is automatically logged out. The *Przedłuż sesję* button resets the time back to five minutes. The *Wyloguj* button is used to correctly log out.



The *Klienci* button displays all customers in the database.

	Pesel	Imię	Nazwisko	Liczba kont
1	63753721793	ad	ad	2
2	89331272436	ko	lo	1
3	75547892583	gdg	gdg	1
4	66666666666	bs	zzzsbzsbzbbzs	2
5	86042601261	Anatolij	Diatłow	2

The *Pracownicy* button displays all employees in the database.

	Login	Imię	Nazwisko	Uprawnienia
1	738693610931	marek	mich	1
2	123456789111	asd	asd	1
3	666666666662	asg	asg	2
4	888888888882	yhwhys	eheshshe	2

The *Dodaj osobę* button allows you to add an administrator or customer to the database.

Pesel

Hasło

Potwierdź hasło

Imię

Nazwisko

Poziom uprawnień

Utwórz

Button *Edytuj osobę*, allows editing the data of a bank customer or employee. You should first select whether you want to edit a client or an employee, and then double-click on the selected person in the table

Kogo chcesz edytować?

Klient

Admin

	Pesel	Imię	Nazwisko	Liczba kont
1	63753721793	ad	ad	2
2	89331272436	ko	lo	1
3	75547892583	gdg	gdg	1
4	66666666666	bs	zzzsbzsbzsbz	2
5	86042601261	Anatolij	Diatłow	2

Pesel	<input type="text" value="86042601261"/>
Hasło	<input type="text"/>
Powtórz hasło	<input type="text"/>
Imię	<input type="text" value="Anatolij"/>
Nazwisko	<input type="text" value="Diatłow"/>
<input type="button" value="Usuń konto"/> <input type="button" value="Zatwierdź"/>	

The *Zarządzaj kontami* button allows you to create a new account, delete an existing account, or change the balance on an account. After selecting one of these three options, select the customer from the table.

<input type="button" value="Utwórz konto"/>
<input type="button" value="Usuń konto"/>
<input type="button" value="Zmień saldo"/>

	Pesel	Imię	Nazwisko	Liczba kont
1	63753721793	ad	ad	2
2	89331272436	ko	lo	1
3	75547892583	gdg	gdg	1
4	66666666666	bs	zzzsbzsbzbbzs	2
5	86042601261	Anatolij	Diatłow	2

When creating a new account, you can enter the starting balance in PLN.

Numer konta 45234153  
 Imię Anatolij  
 Nazwisko Diatłow  
 Pesel 86042601261  
 Saldo

Zatwierdź

When editing a balance or deleting an account after selecting a customer, you need to choose which of their accounts is to be deleted.

45239220  
 45239063

The change balance window shows the current balance of your account.

Aktualne saldo zł	268937.02
Aktualne saldo eur	380569.00
Aktualne saldo usd	0.00

Wprowadź kwoty o ile chcesz zmienić saldo

<input type="text"/>	zł
<input type="text"/>	eur
<input type="text"/>	usd

Potwierdź

The *Zarządzaj lokatami* button gives you the option to create and delete deposits. Once you have selected one of these options, select the customer and then the account.

When creating a deposit you should enter the amount, interest rate and date of closing the deposit. The program checks whether the customer has sufficient funds to set up the deposit. The selection of a past date is not possible. Entering a negative amount or interest rate is also blocked.

Podaj kwotę lokaty

Podaj oprocentowanie

Wybierz datę zamknięcia lokaty

lipiec 2021							
	pon.	wt.	śr.	czw.	pt.	sob.	niedz.
26	28	29	30	1	2	3	4
27	5	6	7	8	9	10	11
28	12	13	14	15	16	17	18
29	19	20	21	22	23	24	25
30	26	27	28	29	30	31	1

Wybrana data: 8 . 7 . 2021

Zatwierdź

When deleting a deposit, simply double-click on it. Funds without accrued interest will be automatically returned.

0	1000.000000	15.000000%	15.12.2021
1	150.000000	1.500000%	8.7.2021

The *Zarządzaj kredytami* button gives you the option to grant and remove credits. After selecting one of these options, select the customer and then the account.

When granting credit, its amount and interest rate must first be stated. Both the amount and the interest rate may not be negative.

Kwota kredytu

Oprocentowanie

The next step is to select the month and year of the loan repayment. It is not possible to select a past date. The monthly instalment will be automatically calculated based on the date selected.

Wybierz miesiąc i rok spłaty kredytu

← styczeń 2023 →							
	pon.	wt.	śr.	czw.	pt.	sob.	niedz.
52	26	27	28	29	30	31	1
1	2	3	4	5	6	7	8
2	9	10	11	12	13	14	15
3	16	17	18	19	20	21	22
4	23	24	25	26	27	28	29

Termin spłaty 21 . 1 . 2023

Miesięczna rata 611,400000

The loan is cancelled in the same way as for deposits. The application will calculate how many instalments the customer still has to repay and will deduct the appropriate amount from the account.

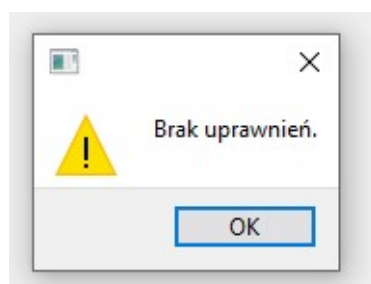
The *Odblokuj klienta* button displays a list of customers who are blocked. To unblock, double-click on the selected person.

	Pesel	Imię	Nazwisko
1	86042601261	Anatolij	Diatłow

The *Zmień hasło* button allows you to change your password.

Podaj staje hasło	<input type="text"/>
Podaj nowe hasło	<input type="text"/>
Powtórz nowe hasło	<input type="text"/>
<input type="button" value="Zatwierdź"/>	

If an attempt is made to use an option that the employee does not have access to, a message will be displayed to inform you of this.



When you are finished, log out and close the application properly with the *Wyjdź* button to save your changes.

All fields in the application are validated to avoid situations where, for example, someone tries to enter letters in the place where an amount should be entered.

- **Internal specifications**
- Class diagram





The choice of classes is supposed to reflect the network of connections between data in the bank and to enable the extension of the application with new functionalities.

Bank employees are represented by the class *Admin* inheriting from *Osoba*. Each object is a unique employee. Employees are managed by an object of class *ListAdminow*.

Customers are represented by the class *Klient* inheriting from the class *Person*. Each *Klient class object* represents a unique customer, and an object of the *ListaKlientow* class is responsible for managing customers. Each customer may have any number of accounts, which are objects of the *Konto class*. The accounts are managed by an object of the *ListaKontclass*.

The *Data* class is inherited by the *Lokata, Kredyt, Historia, Przelew classes*. Objects of these classes are placed in appropriate vectors, which are contained in a given object of the *Konto* type. Each new credit, opened deposit, order of cyclic transfer causes the creation of an object representing these operations. Any change in the balance causes the creation of a *Historia* type object containing information about this event.

The object created from the *Sesja* class is responsible for creating a new thread using *QThread*.

The *MainWindow* class is a class from the Qt library and is responsible for the graphical interface through which the user can manage all existing objects.

All objects except *ListAdminow, ListaKlientow, ListaKont* are stored in vectors. This allows for easy and efficient management from the graphical interface level, as well as their later saving and reading from files. When adding new objects, the application maintains chronology to avoid a situation where e.g. a client wanting to check his account history receives results that are not sorted chronologically.

Besides the GUI and threads, the Qt library is also responsible for downloading the exchange rate from the NBP website using *QtNetwork*.

Used object-oriented techniques from classes:

- Exception mechanism - in case of failure to read data from text files an exception is raised
- STL containers - data is stored in vectors

- STL algorithms and iterators - all operations on vectors that require finding a particular element use iterators in loops. The functions used from the algorithm library are e.g. transform, which in the case of this application together with the use of lambda functions takes care of correct writing and reading of whitespace characters from files.
  - Threads - session duration is monitored by a separate thread
  - Regex - all fields are validated, e.g. in places where you should enter the amount it is not possible to enter letters and other characters, the client is not allowed to enter numbers which have more than 2 decimal places, white characters are not allowed in some cases, e.g. password creation, etc.
- 
- Detailed description of the fields and methods
  - MainWindow
    - *\*ui* - pointer to GUI
    - *id* - stores the login of the currently logged person
    - *l\_admini*, *l\_klienci*, *l\_konta* - objects storing information about all employees, clients, accounts respectively
    - *\*watek* - pointer to thread responsible for watching session time user
  - Sesja
    - *stop* - information whether the thread is running
    - *run* - starts the thread
  - ListaAdminow
    - *admini* - vector storing all employees
    - *wczytaj()* - creates objects representing employees by loading data from a file, and then places them in the *admin* vector. If it fails to open the file, it raises an exception
    - *zapisz()* - saves all objects from *admin* vector to file
    - *znajdzIndeks(...)* - finds the index number in the *admin* vector under which the object containing the login specified as the function argument is placed

- *dodaj(...)* - based on the arguments taken from the GUI it tries to create a new object of the *Admin* type and add it to the *admin* vector. Before doing so, it checks the validity of the entered data and the uniqueness of the user. If this process fails, the user will not be added
- *logowanie(...)* - based on the arguments taken from the GUI it tries to log the employee to his profile
- *sprawdzCzyIstnieje(...)* - checks the uniqueness of the login when creating or editing an employee
- *edytujAdmina(...)* - based on the arguments taken from the GUI tries to change the employee data. It checks the correctness of the entered data and the uniqueness of the login. If this process fails, the user will not be changed
- *Osoba*
  - Abstract class
  - *pesel* - contains user's login
  - *haslo* - contains user's password
  - *imie* - contains the user's first name
  - *nazwisko* - contains the surname of the user
- *Admin*
  - *uprawnienia* - stores the number informing about the entitlements
  - *getPesel()* - returns user's pesel
  - *setPesel(...)* - sets user's pesel
  - *getHaslo()* - returns user's password
  - *setHaslo(...)* - sets user's password
  - *zmienHaslo(...)* - on the basis of arguments taken from the GUI tries to change the password to the currently logged user. Checks for compatibility with the old password and if the same new password is entered twice
- *ListaKlientow*
  - *klienci* - vector storing all customers
  - *dodaj(...)* - on the basis of the arguments taken from the GUI tries to create a new object of the *client* type and add it to the *clients* vector. Before doing so, it

checks the validity of the entered data and the uniqueness of the user. If this process fails, the user will not be added.

- *sprawdzCzyIstnieje(...)* - checks the uniqueness of the login when an employee creates or edits a customer
- *wczytaj()* - creates objects representing clients by loading data from a file, and then places them in the *clients* vector. If it fails to open the file, it raises an exception
- *zapisz()* - saves all objects from *client* vector to file
- *logowanie(...)* - based on the arguments taken from the GUI, tries to log the client into its profile. If the wrong password is entered five times, it is blocked and the login is not possible
- *znajdzIndeks(...)* - finds the index number in the vector of *clients* under which the object containing the login specified as the function argument is placed
- *znajdzBlokady()* - finds customers who are blocked because they have entered the wrong password five times
- *edytujKlienta(...)* - based on arguments taken from GUI tries to change client data. It checks the correctness of the entered data and the uniqueness of the login. If this process fails, the user will not be changed. It is also possible to delete a client and with it all its accounts
- Klient
  - *nr\_hasla* - stores the number of the one-time password to be entered at login
  - *blokada* - stores information whether the user is blocked
  - *ile\_kont* - stores information about the number of bank accounts of the customer
  - *jednorazowe* - vector storing one-time passwords
  - *konta* - vector storing all objects representing user accounts
  - *getPesel()* - returns user's pesel
  - *setPesel(...)* - sets user's pesel
  - *getHaslo()* - returns user's password
  - *setHaslo(...)* - sets user's password

- *logTmp(...)* - based on a one-time password taken from the GUI, attempts to log the client into his profile. If the wrong password is entered five times, it is blocked and the login is not possible
- *generujTmp()* - generates a new list of one-time passwords
- *zmienHaslo(...)* - on the basis of arguments taken from the GUI tries to change the password to the currently logged user. Checks for compatibility with the old password and if the same new password is entered twice
- *dodajKonto (...)* - creates a new bank account and adds it for the customer with the login taken from the GUI
- ListaKont
  - *konta* - vector storing all accounts
  - *dodaj (...)* - based on the arguments taken from the GUI tries to create a new object of the *Account* type and add it to the *account* vector. Before that a unique account number is generated.
  - *generujNr (...)* - generates a unique account number
  - *wczytaj ()* - creates objects representing accounts by loading data from a file, and then places them in the *account* vector. If it fails to open the file, it reports an exception
  - *zapisz ()* - saves all objects from *account* vector to file
  - *znajdzIndeks(...)* - finds the index number in the *account* vector under which the object containing the account number given as the function argument is placed
  - *naliczLokaty(...)* - checks if the deposit accrual date has passed, if yes it increases the balance on a given account, removes the *Deposit* type object and generates an entry to the history.
  - *wyslijPrzelew(...)* - sends a transfer to the account number taken from the GUI
  - *naliczRaty()* - calculates credit instalments (including overdue ones) for accounts with credits, taking the appropriate amount and then generates a history entry; if the credit has been repaid, it removes the *Credit* type object and generates a history entry
  - *przelewCykl()* - checks if a transfer should be sent and if there are any outstanding orders, if there are sufficient funds on the account the transfer is executed and with it an appropriate entry in the history.

- *przelewHistoria(...)* - generates a history entry for the sender and recipient of the transfer
- *zmienPesel(...)* - changes client's pesel
- *usunKonto(...)* - deletes the account with the selected number
- Account
  - *saldozl* - how many zlotys on your account
  - *saldoeu* - how many euros in your account
  - *saldod* - how many dollars on your account
  - *numer* - account number
  - *wlaściciel* - pesel of the account owner
  - *lokaty* - vector storing all the deposits for a given account
  - *przelewy* - vector storing all cyclical transfers for the given account
  - *historia* - a vector storing the history of account balance changes in each currency
  - *getSaldo(...)* - returns the account balance in the selected currency
  - *setSaldo(...)* - sets the account balance in the selected currency
  - *editSaldo(...)* - increases or decreases the account balance in the selected currency
  - *dodajLokata(...)* - creates a new deposit object for the selected account and adds it to the *deposit* vector
  - *przelewMozliwy(...)* - checks if transfer execution from a given account is possible, if yes it removes the appropriate amount from the sender and returns information that the transfer can be sent
  - *rataKredytu(...)* - calculates a loan instalment based on the amount, interest rate and repayment date
  - *dodajKredyt(...)* - creates a new object of the *Credit* type and adds it to the vector of *credits* for the selected account
  - *dodajPrzelew(...)* - creates a new *Transfer* type object and adds it to the *transfers* vector for the selected account

- Data
  - *data* - stores the start date, e.g. of a loan, deposit, etc.
  - *data2* - holds the end date
- Lokata
  - *kwota* - stores the amount paid into the deposit
  - *procent* - holds the interest on the deposit
  - *getKwota()* - returns the deposit amount
  - *setKwota(...)* - sets the deposit amount
- Kredyt
  - *rata* - stores the amount of the monthly instalment
  - *kwota* - I store the total amount of credit
  - *procent* - credit interest rate
  - *r\_zaplacone* - the year of the last instalment paid
  - *m\_zaplacone* - month of last instalment paid
  - *setRata(...)* - sets a credit instalment
  - *setAmount(...)* - sets the loan amount
  - *setProcent(...)* - sets the loan percentage
  - *getKwota()* - returns the deposit amount
  - *getKwota()* - returns the deposit amount
  - *getKwota()* - returns the deposit amount
  - *ileDoSplaty()* - calculates in case of cancellation of a credit how much money is left to repay and returns this value
- Historia
  - *operacja* - stores information on the operation affecting the account balance
- Przelew
  - *kwota* - stores the amount of the transfer



- *nr* - stores the recipient's number
- *tresc* - stores the content of the transfer
- *tytul* - stores the title of the transfer
- *cykliczny* - stores information on which day of the month the transfer is to be made
- *zapłacone* - stores information whether a cyclical transfer in the current month has already been made

- General scheme of operation

When started, the program creates 3 objects: *I\_konta*, *I\_admini*, *I\_klienci*, and then tries to load all the data from the three files into them. If this fails, an exception is raised. Then cyclic transfers are executed and loan instalments are paid if the conditions for these operations are met (overdue ones are also executed, e.g. if the program has not been started for 3 months). The first thing the user has to do is to log in. In the case of a client, the program checks whether it is blocked. It is blocked if the wrong password is entered five times. After logging in the thread counting down the session time of 5 minutes starts. After 5 minutes an automatic logout takes place if the user does not reset the time with the button. Subsequent operations are performed using the buttons. They are described in detail in the instruction. During the program's operation all data are validated to avoid errors, such as sending a negative transfer. Additionally, it is not possible for the customer to send a transfer with an amount that has more than 2 decimal places. In the case of currency exchange, if the downloading of the exchange rate fails, this option will not be available. In order to save changes to files you should log out and close the program with button *Wyjdź*.

## • Testing

- All fields in the form have been tested to see if validation is successful.
- It is not possible to create two identical clients. A bank employee can have more employee accounts if they have different permissions on each of them.

- It is not possible to set up a deposit and send a transfer if there are not enough funds in your account.
- All saving and loading combinations have been tested, e.g. the account has only a deposit, the account has a loan and a deposit, the account has a couple of cyclical transfers and a deposit, etc.
- The accrual of overdue instalments and the execution of overdue cyclical transfers have been tested over a range from one month to a few years back.
- The one-time password system will automatically generate a new list for the customer if they have used all of them and have not manually renewed them.
- Closed deposits and repaid loans are correctly deleted.
- All history information is correctly displayed, especially received and sent transfers, as they additionally contain the title and content.

## • **Summary**

The creation of an application simulating the operation of a bank was time-consuming. It was necessary to take into account all the data that the user can enter, so that there is no situation that the program does not know what to do. You can not allow the situation that the money somewhere "disappears". The biggest disadvantage of the program is that saving data takes place only after closing the application. In case of power failure or system suspension, the changes will not be saved.