Projekt Kantor

Generated by Doxygen 1.13.2

Chapter 1

This is the title of your main page.

This is your main page.

Here are images:

Chapter 2

Subsyste1 Title

Sub Subsystem 1

2.1 Section1

Some text

2.2 Section2

Some more text

2.2.1 Sub section1

Some moreee text

2.3 Sub Subsystem 1

Some text here! See this page for more: Section2

Chapter 3

Hierarchical Index

3.1 Class Hierarchy

This inheritance list is sorted roughly, but not completely, alphabetically:

Chapter 4

Class Index

4.1 Class List

Here are the classes, structs, unions and interfaces with brief descriptions:

Chapter 5

File Index

5.1 File List

Here is a list of all files with brief descriptions:

Chapter 6

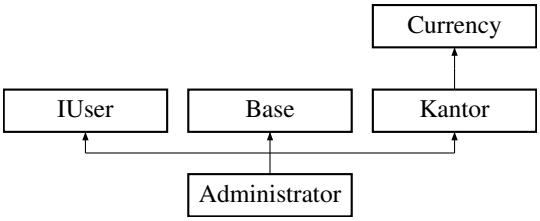
Class Documentation

6.1 Administrator Class Reference

Klasa reprezentująca administratora systemu.

`#include <Admin.h>`

Inheritance diagram for Administrator:



Public Member Functions

- Administrator (std::string password)

  Konstruktor klasy Administrator.
- void typePassword ()

  Metoda do wprowadzania hasła.
- void displayMenu () override

  Wyświetla menu administratora.
- void changeRates ()

  Metoda do zmiany kursów walut.
- void saveRatesToFile (std::string filename, std::map< std::string, double > rates)

  Zapisuje kursy walut do pliku.
- void showResources ()

  Wyświetla zasoby systemu.
- void setChange ()

  Ustawia zmiany w systemie.
- void changePassword ()

  Zmienia hasło administratora.
- void showTransactions ()

Wyświetla historię transakcji.

- void clearTransactions ()

  Czyści historię transakcji.
- bool error (int choice, const std::string error_msg)

  Sprawdza błędy na podstawie wyboru użytkownika.
- bool error (const std::string &inputPassword, std::string error_msg)

  Sprawdza błędy na podstawie wprowadzonego hasła.

Public Member Functions inherited from IUser

- virtual ~IUser ()=default

  Wirtualny destruktor.

Public Member Functions inherited from Base

- void drawBorder (const std::vector< std::string > &menuItems) const

  Rysuje obramowanie wokół menu.

Public Member Functions inherited from Kantor

- void loadResources (const std::string file_name, std::map< std::string, double > &currencies)

  Wczytuje zasoby walut z pliku.
- void showResources ()

  Wyświetla dostępne zasoby walut.
- void changeResources (const std::string currency, double amount, const std::string command, std::string transaction)

  Zmienia zasoby walut.
- void makeTransaction (const std::string &imie, const std::string &nazwisko, const std::string &currency, double amount, double price, std::string transaction)

  Przeprowadza transakcję wymiany walut.
- bool error_cur (int choice, std::string error_msg)

  Sprawdza błędy związane z wyborem waluty.
- bool error_am (std::string currency, double amount, const std::string error_msg)

  Sprawdza błędy związane z ilością waluty.

Static Public Attributes

- static bool logged { false }

  Flaga informująca o stanie zalogowania administratora.

Additional Inherited Members

Protected Member Functions inherited from Currency

- Currency ()=default

  Konstruktor domyślny klasy Currency.
- void loadRates (std::string BuyRatesFile, std::string SellRatesFile)

  Wczytuje kursy kupna i sprzedaży z plików.
- void showRates ()

  Wyświetla kursy kupna i sprzedaży.

Protected Attributes inherited from Kantor

- std::map< std::string, double > currencies

  Mapa przechowująca dostępne waluty i ich ilości.

Protected Attributes inherited from Currency

- std::map< std::string, double > buyRates

  Mapa przechowująca kursy kupna walut.
- std::map< std::string, double > sellRates

  Mapa przechowująca kursy sprzedaży walut.

6.1.1 Detailed Description

Klasa reprezentująca administratora systemu.

Klasa dziedziczy po interfejsie IUser oraz klasach Base i Kantor. Odpowiada za zarządzanie systemem, w tym za zmiany kursów, zarządzanie zasobami oraz obsługę transakcji.

6.1.2 Constructor & Destructor Documentation

6.1.2.1 Administrator()

```
Administrator::Administrator (
            std::string password)
```

Konstruktor klasy Administrator.

Parameters

in	*password*	Hasło administratora.

6.1.3 Member Function Documentation

6.1.3.1 changePassword()

```
void Administrator::changePassword ()
```

Zmienia hasło administratora.

6.1.3.2 changeRates()

```
void Administrator::changeRates ()
```

Metoda do zmiany kursów walut.

6.1.3.3 clearTransactions()

```
void Administrator::clearTransactions ()
```

Czyści historię transakcji.

6.1.3.4 displayMenu()

```
void Administrator::displayMenu ()  [override], [virtual]
```

Wyświetla menu administratora.

Metoda nadpisuje funkcję displayMenu z interfejsu IUser.

Implements IUser.

6.1.3.5 error() [1/2]

```
bool Administrator::error (
            const std::string & inputPassword,
            std::string error_msg)
```

Sprawdza błędy na podstawie wprowadzonego hasła.

Parameters

in	*inputPassword*	Wprowadzone hasło.
in	*error_msg*	Wiadomość błędu do wyświetlenia w przypadku nieprawidłowego hasła.

Returns

Zwraca true, jeśli wystąpił błąd; false w przeciwnym razie.

6.1.3.6 error() [2/2]

```
bool Administrator::error (
            int choice,
            const std::string error_msg)
```

Sprawdza błędy na podstawie wyboru użytkownika.

Parameters

in	*choice*	Wybór użytkownika.
in	*error_msg*	Wiadomość błędu do wyświetlenia w przypadku nieprawidłowego wyboru.

Returns

Zwraca true, jeśli wystąpił błąd; false w przeciwnym razie.

6.1.3.7 saveRatesToFile()

```
void Administrator::saveRatesToFile (
            std::string filename,
            std::map< std::string, double > rates)
```

Zapisuje kursy walut do pliku.

Parameters

in	*filename*	Nazwa pliku, do którego mają zostać zapisane kursy.
in	*rates*	Mapa zawierająca kursy walut.

6.1.3.8 setChange()

```
void Administrator::setChange ()
```

Ustawia zmiany w systemie.

6.1.3.9 showResources()

```
void Administrator::showResources ()
```

Wyświetla zasoby systemu.

6.1.3.10 showTransactions()

```
void Administrator::showTransactions ()
```

Wyświetla historię transakcji.

6.1.3.11 typePassword()

```
void Administrator::typePassword ()
```

Metoda do wprowadzania hasła.

6.1.4 Member Data Documentation

6.1.4.1 logged

```
bool Administrator::logged { false }  [static]
```

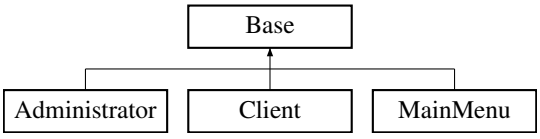
Flaga informująca o stanie zalogowania administratora.

6.2 Base Class Reference

Klasa bazowa zawierająca metody wspólne dla innych klas.

`#include <Border.h>`

Inheritance diagram for Base:



Public Member Functions

- void drawBorder (const std::vector< std::string > &menuItems) const

  Rysuje obramowanie wokół menu.

6.2.1 Detailed Description

Klasa bazowa zawierająca metody wspólne dla innych klas.

Klasa zawiera metodę do rysowania obramowania wokół elementów menu.

6.2.2 Member Function Documentation

6.2.2.1 drawBorder()

```
void Base::drawBorder (
            const std::vector< std::string > & menuItems) const  [inline]
```

Rysuje obramowanie wokół menu.

Metoda rysuje obramowanie wokół przekazanych elementów menu. Każdy element menu jest wyświetlany w osobnej linii z obramowaniem z góry, dołu oraz po bokach.

Parameters

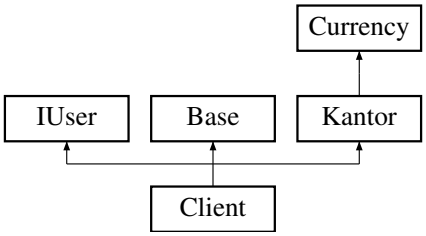
in	*menuItems*	Wektor zawierający elementy menu do wyświetlenia.

6.3 Client Class Reference

Klasa reprezentująca klienta systemu.

```
#include <Client.h>
```

Inheritance diagram for Client:



Public Member Functions

- void setName ()

  Ustawia imię klienta.
- void displayMenu () override

  Wyświetla menu klienta.
- void setTransaction (const std::string transaction)

  Ustawia transakcję klienta.
- bool error (int choice, std::string error_msg)

  Sprawdza błędy na podstawie wyboru użytkownika.
- bool error_transaction (int choice, int size, std::string error_msg)

  Sprawdza błędy na podstawie wyboru transakcji.

Public Member Functions inherited from IUser

- virtual ~IUser ()=default

  Wirtualny destruktor.

Public Member Functions inherited from Base

- void drawBorder (const std::vector< std::string > &menuItems) const

  Rysuje obramowanie wokół menu.

Public Member Functions inherited from Kantor

- void loadResources (const std::string file_name, std::map< std::string, double > &currencies)

  Wczytuje zasoby walut z pliku.
- void showResources ()

  Wyświetla dostępne zasoby walut.
- void changeResources (const std::string currency, double amount, const std::string command, std::string transaction)

  Zmienia zasoby walut.
- void makeTransaction (const std::string &imie, const std::string &nazwisko, const std::string &currency, double amount, double price, std::string transaction)

  Przeprowadza transakcję wymiany walut.
- bool error_cur (int choice, std::string error_msg)

  Sprawdza błędy związane z wyborem waluty.
- bool error_am (std::string currency, double amount, const std::string error_msg)

  Sprawdza błędy związane z ilością waluty.

Additional Inherited Members

Protected Member Functions inherited from Currency

- Currency ()=default

  Konstruktor domyślny klasy Currency.
- void loadRates (std::string BuyRatesFile, std::string SellRatesFile)

  Wczytuje kursy kupna i sprzedaży z plików.
- void showRates ()

  Wyświetla kursy kupna i sprzedaży.

Protected Attributes inherited from Kantor

- std::map< std::string, double > currencies
  Mapa przechowująca dostępne waluty i ich ilości.

Protected Attributes inherited from Currency

- std::map< std::string, double > buyRates
  Mapa przechowująca kursy kupna walut.
- std::map< std::string, double > sellRates
  Mapa przechowująca kursy sprzedaży walut.

6.3.1 Detailed Description

Klasa reprezentująca klienta systemu.

Klasa dziedziczy po interfejsie IUser oraz klasach Base i Kantor. Odpowiada za zarządzanie informacjami o kliencie oraz operacjami związanymi z transakcjami.

6.3.2 Member Function Documentation

6.3.2.1 displayMenu()

```
void Client::displayMenu ()  [override], [virtual]
```

Wyświetla menu klienta.

Metoda nadpisuje funkcję displayMenu z interfejsu IUser.

Implements IUser.

6.3.2.2 error()

```
bool Client::error (
            int choice,
            std::string error_msg)
```

Sprawdza błędy na podstawie wyboru użytkownika.

Parameters

in	*choice*	Wybór użytkownika.
in	*error_msg*	Wiadomość błędu do wyświetlenia w przypadku nieprawidłowego wyboru.

Returns

Zwraca true, jeśli wystąpił błąd; false w przeciwnym razie.

6.3.2.3 error_transaction()

```
bool Client::error_transaction (
            int choice,
            int size,
            std::string error_msg)
```

Sprawdza błędy na podstawie wyboru transakcji.

Parameters

in	*choice*	Wybór użytkownika.
in	*size*	Oczekiwana wielkość transakcji.
in	*error_msg*	Wiadomość błędu do wyświetlenia w przypadku nieprawidłowej transakcji.

Returns

Zwraca true, jeśli wystąpił błąd; false w przeciwnym razie.

6.3.2.4 setName()

```
void Client::setName ()
```

Ustawia imię klienta.

6.3.2.5 setTransaction()

```
void Client::setTransaction (
            const std::string transaction)
```

Ustawia transakcję klienta.

Parameters

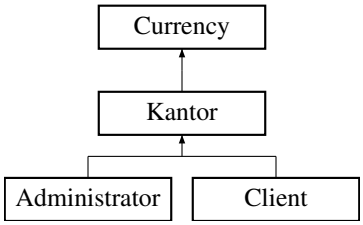
in	*transaction*	Opis transakcji.

6.4 Currency Class Reference

Klasa do zarządzania kursami walut.

```
#include <Currency.h>
```

Inheritance diagram for Currency:



Protected Member Functions

- Currency ()=default

  Konstruktor domyślny klasy Currency.
- void loadRates (std::string BuyRatesFile, std::string SellRatesFile)

  Wczytuje kursy kupna i sprzedaży z plików.
- void showRates ()

  Wyświetla kursy kupna i sprzedaży.

Protected Attributes

- std::map< std::string, double > buyRates

  Mapa przechowująca kursy kupna walut.
- std::map< std::string, double > sellRates

  Mapa przechowująca kursy sprzedaży walut.

6.4.1 Detailed Description

Klasa do zarządzania kursami walut.

Klasa Currency zawiera metody do wczytywania i wyświetlania kursów kupna i sprzedaży walut.

6.4.2 Constructor & Destructor Documentation

6.4.2.1 Currency()

```
Currency::Currency ()  [protected], [default]
```

Konstruktor domyślny klasy Currency.

6.4.3 Member Function Documentation

6.4.3.1 loadRates()

```
void Currency::loadRates (
            std::string BuyRatesFile,
            std::string SellRatesFile)  [inline], [protected]
```

Wczytuje kursy kupna i sprzedaży z plików.

Parameters

in	*BuyRatesFile*	Nazwa pliku zawierającego kursy kupna.
in	*SellRatesFile*	Nazwa pliku zawierającego kursy sprzedaży.

6.4.3.2 showRates()

```
void Currency::showRates ()  [inline], [protected]
```

Wyświetla kursy kupna i sprzedaży.

6.4.4 Member Data Documentation

6.4.4.1 buyRates

```
std::map<std::string, double> Currency::buyRates  [protected]
```

Mapa przechowująca kursy kupna walut.

6.4.4.2 sellRates

```
std::map<std::string, double> Currency::sellRates  [protected]
```

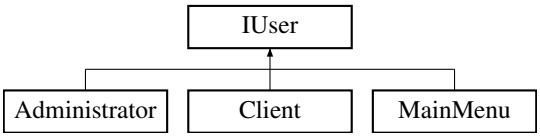
Mapa przechowująca kursy sprzedaży walut.

6.5 IUser Class Reference

Interfejs użytkownika.

```
#include <IUser.h>
```

Inheritance diagram for IUser:



Public Member Functions

- virtual void displayMenu ()=0
  Wyświetla menu użytkownika.
- virtual ~IUser ()=default
  Wirtualny destruktor.

6.5.1 Detailed Description

Interfejs użytkownika.

Klasa IUser definiuje interfejs dla użytkowników systemu, wymagający implementacji metody displayMenu.

6.5.2 Constructor & Destructor Documentation

6.5.2.1 ~IUser()

```
virtual IUser::~IUser ()  [virtual], [default]
```

Wirtualny destruktor.

Umożliwia prawidłowe zarządzanie zasobami w klasach dziedziczących.

6.5.3 Member Function Documentation

6.5.3.1 displayMenu()

```
virtual void IUser::displayMenu ()  [pure virtual]
```

Wyświetla menu użytkownika.

Metoda czysto wirtualna, która jest zaimplementowana przez klasy dziedziczące.

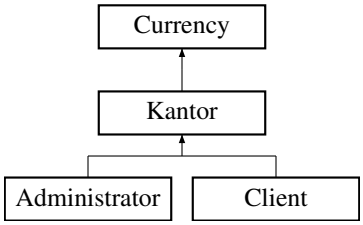
Implemented in Administrator, Client, and MainMenu.

6.6 Kantor Class Reference

Klasa reprezentująca kantor wymiany walut.

`#include <Kantor.h>`

Inheritance diagram for Kantor:



Public Member Functions

- void loadResources (const std::string file_name, std::map< std::string, double > &currencies)

  Wczytuje zasoby walut z pliku.
- void showResources ()

  Wyświetla dostępne zasoby walut.
- void changeResources (const std::string currency, double amount, const std::string command, std::string transaction)

  Zmienia zasoby walut.
- void makeTransaction (const std::string &imie, const std::string &nazwisko, const std::string &currency, double amount, double price, std::string transaction)

  Przeprowadza transakcję wymiany walut.
- bool error_cur (int choice, std::string error_msg)

  Sprawdza błędy związane z wyborem waluty.
- bool error_am (std::string currency, double amount, const std::string error_msg)

  Sprawdza błędy związane z ilością waluty.

Protected Attributes

- std::map< std::string, double > currencies

  Mapa przechowująca dostępne waluty i ich ilości.

Protected Attributes inherited from Currency

- std::map< std::string, double > buyRates

  Mapa przechowująca kursy kupna walut.
- std::map< std::string, double > sellRates

  Mapa przechowująca kursy sprzedaży walut.

Additional Inherited Members

Protected Member Functions inherited from Currency

- Currency ()=default

  Konstruktor domyślny klasy Currency.
- void loadRates (std::string BuyRatesFile, std::string SellRatesFile)

  Wczytuje kursy kupna i sprzedaży z plików.
- void showRates ()

  Wyświetla kursy kupna i sprzedaży.

6.6.1 Detailed Description

Klasa reprezentująca kantor wymiany walut.

Klasa Kantor dziedziczy po klasie Currency i zawiera metody do zarządzania zasobami walut oraz realizacji transakcji.

6.6.2 Member Function Documentation

6.6.2.1 changeResources()

```
void Kantor::changeResources (
            const std::string currency,
            double amount,
            const std::string command,
            std::string transaction)
```

Zmienia zasoby walut.

Parameters

in	*currency*	Nazwa waluty.
in	*amount*	Ilość waluty do zmiany.
in	*command*	Komenda określająca typ zmiany (dodanie/usunięcie).
in	*transaction*	Opis transakcji.

6.6.2.2 error_am()

```
bool Kantor::error_am (
            std::string currency,
            double amount,
            const std::string error_msg)
```

Sprawdza błędy związane z ilością waluty.

Parameters

in	*currency*	Nazwa waluty.
in	*amount*	Ilość waluty.
in	*error_msg*	Wiadomość błędu do wyświetlenia w przypadku nieprawidłowej ilości.

Returns

Zwraca true, jeśli wystąpił błąd; false w przeciwnym razie.

6.6.2.3 error_cur()

```
bool Kantor::error_cur (
            int choice,
            std::string error_msg)
```

Sprawdza błędy związane z wyborem waluty.

Parameters

in	*choice*	Wybór waluty.
in	*error_msg*	Wiadomość błędu do wyświetlenia w przypadku nieprawidłowego wyboru.

Returns

Zwraca true, jeśli wystąpił błąd; false w przeciwnym razie.

6.6.2.4 loadResources()

```
void Kantor::loadResources (
            const std::string file_name,
            std::map< std::string, double > & currencies)
```

Wczytuje zasoby walut z pliku.

Parameters

in	*file_name*	Nazwa pliku zawierającego zasoby walut.
out	*currencies*	Mapa, do której zostaną wczytane zasoby walut.

6.6.2.5 makeTransaction()

```
void Kantor::makeTransaction (
            const std::string & imie,
            const std::string & nazwisko,
            const std::string & currency,
            double amount,
            double price,
            std::string transaction)
```

Przeprowadza transakcję wymiany walut.

Parameters

in	*imie*	Imię klienta.
in	*nazwisko*	Nazwisko klienta.
in	*currency*	Nazwa waluty.
in	*amount*	Ilość waluty do wymiany.
in	*price*	Cena wymiany.
in	*transaction*	Opis transakcji.

6.6.2.6 showResources()

```
void Kantor::showResources ()
```

Wyświetla dostępne zasoby walut.

6.6.3 Member Data Documentation

6.6.3.1 currencies

```
std::map<std::string, double> Kantor::currencies  [protected]
```

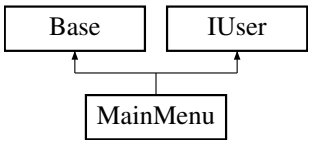
Mapa przechowująca dostępne waluty i ich ilości.

6.7 MainMenu Class Reference

Klasa reprezentująca główne menu aplikacji.

```
#include <MainMenu.h>
```

Inheritance diagram for MainMenu:



Public Member Functions

- MainMenu ()

  Konstruktor domyślny klasy MainMenu.
- void displayMenu () override

  Wyświetla główne menu aplikacji.
- bool error (int choice, std::string error_msg)

  Sprawdza błędy na podstawie wyboru użytkownika.

Public Member Functions inherited from Base

- void drawBorder (const std::vector< std::string > &menuItems) const
  Rysuje obramowanie wokół menu.

Public Member Functions inherited from IUser

- virtual ~IUser ()=default
  Wirtualny destruktor.

6.7.1 Detailed Description

Klasa reprezentująca główne menu aplikacji.

Klasa MainMenu dziedziczy po klasach Base i IUser, zapewniając interfejs głównego menu aplikacji.

6.7.2 Constructor & Destructor Documentation

6.7.2.1 MainMenu()

```
MainMenu::MainMenu ()
```

Konstruktor domyślny klasy MainMenu.

6.7.3 Member Function Documentation

6.7.3.1 displayMenu()

```
void MainMenu::displayMenu ()  [override], [virtual]
```

Wyświetla główne menu aplikacji.

Metoda nadpisuje funkcję displayMenu z interfejsu IUser.

Implements IUser.

6.7.3.2 error()

```
bool MainMenu::error (
            int choice,
            std::string error_msg)
```

Sprawdza błędy na podstawie wyboru użytkownika.

Parameters

in	*choice*	Wybór użytkownika.
in	*error_msg*	Wiadomość błędu do wyświetlenia w przypadku nieprawidłowego wyboru.

Returns

Zwraca true, jeśli wystąpił błąd; false w przeciwnym razie.

6.8 Transaction Class Reference

Klasa reprezentująca transakcję wymiany walut.

`#include <Transactions.h>`

Public Member Functions

- Transaction ()

  Konstruktor domyślny klasy Transaction.
- Transaction (const std::string &name, const std::string &surname, const std::string &curr, double amt, double pr, const std::string &typ)

  Konstruktor klasy Transaction z parametrami.
- void saveToFile (const std::string &file_name, std::string transaction) const

  Zapisuje transakcję do pliku.

6.8.1 Detailed Description

Klasa reprezentująca transakcję wymiany walut.

Klasa Transaction zawiera informacje o transakcji, takie jak imię i nazwisko klienta, waluta, ilość, cena, typ transakcji oraz data i czas.

6.8.2 Constructor & Destructor Documentation

6.8.2.1 Transaction() [1/2]

```
Transaction::Transaction ()  [inline]
```

Konstruktor domyślny klasy Transaction.

6.8.2.2 Transaction() [2/2]

```
Transaction::Transaction (
            const std::string & name,
            const std::string & surname,
            const std::string & curr,
            double amt,
            double pr,
            const std::string & typ)
```

Konstruktor klasy Transaction z parametrami.

Parameters

in	*name*	Imię klienta.
in	*surname*	Nazwisko klienta.
in	*curr*	Waluta transakcji.
in	*amt*	Ilość waluty.
in	*pr*	Cena transakcji.
in	*typ*	Typ transakcji: "BUY" lub "SELL".

6.8.3 Member Function Documentation

6.8.3.1 saveToFile()

```
void Transaction::saveToFile (
            const std::string & file_name,
            std::string transaction) const
```

Zapisuje transakcję do pliku.

Parameters

in	*file_name*	Nazwa pliku, do którego zostanie zapisana transakcja.
in	*transaction*	Opis transakcji do zapisania.

Chapter 7

File Documentation

7.1 doc_pages/mainpage.md File Reference

7.2 doc_pages/subsubsystem.md File Reference

7.3 doc_pages/subsystem1.md File Reference

7.4 ProjektKantor/Admin.cpp File Reference

```
#include <iostream>
#include <conio.h>
#include <vector>
#include "Admin.h"
#include "MainMenu.h"
#include "Transactions.h"
```

7.5 ProjektKantor/Admin.h File Reference

```
#include "IUser.h"
#include "Border.h"
#include "Kantor.h"
```

Classes

- class Administrator

  Klasa reprezentująca administratora systemu.

7.6 Admin.h

Go to the documentation of this file.

```
00001 #pragma once
00002
00003 #include "IUser.h"
00004 #include "Border.h"
00005 #include "Kantor.h"
00006
00014 class Administrator : public IUser, public Base, public Kantor
00015 {
00016     std::string password{};
00017     std::string passwordFile = "password.txt";
00018 public:
00019     static bool logged;
00020
00026     Administrator(std::string password);
00027
00031     void typePassword();
00032
00038     void displayMenu() override;
00039
00043     void changeRates();
00044
00051     void saveRatesToFile(std::string filename, std::map<std::string, double> rates);
00052
00056     void showResources();
00057
00061     void setChange();
00062
00066     void changePassword();
00067
00071     void showTransactions();
00072
00076     void clearTransactions();
00077
00085     bool error(int choice, const std::string error_msg);
00086
00094     bool error(const std::string& inputPassword, std::string error_msg);
00095
00096 private:
00100     void savePasswordToFile();
00101
00105     void loadPasswordFromFile();
00106 };
```

7.7 ProjektKantor/Border.h File Reference

```
#include <iostream>
#include <string>
#include <vector>
#include <algorithm>
```

Classes

- class Base

  Klasa bazowa zawierająca metody wspólne dla innych klas.

7.8 Border.h

Go to the documentation of this file.

```
00001 #pragma once
00002
00003 #include <iostream>
```

```
00004 #include <string>
00005 #include <vector>
00006 #include <algorithm>
00007
00014 class Base
00015 {
00016 public:
00025     void drawBorder(const std::vector<std::string>& menuItems) const {
00026
00027         // Znajdź najdłuższą linię
00028         size_t maxLength{ 0 };
00029         for (const auto& item : menuItems) {
00030             maxLength = std::max(maxLength, item.length());
00031         }
00032
00033         // Rysuj górne obramowanie
00034         std::cout « "+";
00035         for (size_t i{ 0 }; i < maxLength + 2; ++i) {
00036             std::cout « "-";
00037         }
00038         std::cout « "+\n";
00039
00040         // Rysuj menu z bocznymi obramowaniami
00041         for (const auto& item : menuItems) {
00042             std::cout « "| " « item;
00043             for (size_t i{ item.length() }; i < maxLength; ++i) {
00044                 std::cout « " ";
00045             }
00046             std::cout « " |\n";
00047         }
00048
00049         // Rysuj dolne obramowanie
00050         std::cout « "+";
00051         for (size_t i{ 0 }; i < maxLength + 2; ++i) {
00052             std::cout « "-";
00053         }
00054         std::cout « "+\n";
00055     }
00056 };
```

7.9 ProjektKantor/Client.cpp File Reference

```
#include <iostream>
#include <vector>
#include <conio.h>
#include "Client.h"
#include "MainMenu.h"
```

7.10 ProjektKantor/Client.h File Reference

```
#include "IUser.h"
#include "Border.h"
#include "Kantor.h"
```

Classes

- class Client

  Klasa reprezentująca klienta systemu.

7.11 Client.h

Go to the documentation of this file.

```
00001 #pragma once
00002
00003 #include "IUser.h"
00004 #include "Border.h"
00005 #include "Kantor.h"
00006
00014 class Client : public IUser, public Base, public Kantor
00015 {
00016     std::string imie{};
00017     std::string nazwisko{};
00018
00019 public:
00023     void setName();
00024
00030     void displayMenu() override;
00031
00037     void setTransaction(const std::string transaction);
00038
00046     bool error(int choice, std::string error_msg);
00047
00056     bool error_transaction(int choice, int size, std::string error_msg);
00057 };
```

7.12 ProjektKantor/Currency.h File Reference

```
#include <fstream>
#include <iostream>
#include <sstream>
#include <map>
#include <string>
```

Classes

- class Currency

  Klasa do zarządzania kursami walut.

7.13 Currency.h

Go to the documentation of this file.

```
00001 #pragma once
00002 #include <fstream>
00003 #include <iostream>
00004 #include <sstream>
00005 #include <map>
00006 #include <string>
00007
00014 class Currency
00015 {
00016 protected:
00017     std::map<std::string, double> buyRates;
00018     std::map<std::string, double> sellRates;
00019
00023     Currency() = default;
00024
00031     void loadRates(std::string BuyRatesFile, std::string SellRatesFile)
00032     {
00033         loadBuyRates(BuyRatesFile);
00034         loadSellRates(SellRatesFile);
00035     }
00036
00040     void showRates()
```

```
00041     {
00042         system("cls");
00043         std::cout « "Kursy kupna:\n";
00044         for (const auto& pair : buyRates)
00045         {
00046             std::cout « pair.first « ": " « pair.second « "\n";
00047         }
00048         std::cout « "\nKursy sprzedaży:\n";
00049         for (const auto& pair : sellRates)
00050         {
00051             std::cout « pair.first « ": " « pair.second « "\n";
00052         }
00053     }
00054
00055 private:
00061     void loadBuyRates(std::string BuyRatesFile)
00062     {
00063         loadRatesFromFile(BuyRatesFile, buyRates);
00064     }
00065
00071     void loadSellRates(std::string SellRatesFile)
00072     {
00073         loadRatesFromFile(SellRatesFile, sellRates);
00074     }
00075
00082     void loadRatesFromFile(const std::string file_name, std::map<std::string, double>& RatesMap)
00083     {
00084         std::ifstream in(file_name);
00085
00086         if (!in.is_open())
00087         {
00088             std::cerr « "Nie udało się otworzyć pliku '" « file_name « "' !" « std::endl;
00089             return;
00090         }
00091
00092         std::string line;
00093
00094         while (std::getline(in, line))
00095         {
00096             std::istringstream iss(line);
00097             std::string currency;
00098             double rate;
00099             if (!(iss » currency » rate))
00100             {
00101                 std::cerr « "Błąd wczytywania kursu z linii!" « std::endl;
00102                 continue;
00103             }
00104
00105             RatesMap[currency] = rate;
00106         }
00107
00108         in.close();
00109     }
00110 };
```

7.14 ProjektKantor/IUser.h File Reference

Classes

- class IUser

  Interfejs użytkownika.

7.15 IUser.h

Go to the documentation of this file.

```
00001 #pragma once
00002
00009 class IUser
00010 {
00011 public:
00017     virtual void displayMenu() = 0;
00018
00024     virtual ~IUser() = default;
00025 };
```

7.16 ProjektKantor/Kantor.cpp File Reference

```
#include <conio.h>
#include <fstream>
#include <sstream>
#include "Kantor.h"
#include "Transactions.h"
```

7.17 ProjektKantor/Kantor.h File Reference

```
#include "Currency.h"
#include <string>
#include <chrono>
#include <map>
```

Classes

- class Kantor

  Klasa reprezentująca kantor wymiany walut.

7.18 Kantor.h

Go to the documentation of this file.

```
00001 #pragma once
00002 #include "Currency.h"
00003 #include <string>
00004 #include <chrono>
00005 #include <map>
00006
00013 class Kantor : public Currency
00014 {
00015 protected:
00016     std::map<std::string, double> currencies;
00017
00018 public:
00025     void loadResources(const std::string file_name, std::map<std::string, double>& currencies);
00026
00030     void showResources();
00031
00040     void changeResources(const std::string currency, double amount, const std::string command,
      std::string transaction);
00041
00052     void makeTransaction(const std::string& imie, const std::string& nazwisko, const std::string&
      currency, double amount, double price, std::string transaction);
00053
00061     bool error_cur(int choice, std::string error_msg);
00062
00071     bool error_am(std::string currency, double amount, const std::string error_msg);
00072 };
```

7.19 ProjektKantor/main.cpp File Reference

```
#include "MainMenu.h"
```

Functions

- int main ()

7.19.1 Function Documentation

7.19.1.1 main()

```
int main ()
```

7.20 ProjektKantor/MainMenu.cpp File Reference

```
#include <iostream>
#include <vector>
#include <conio.h>
#include "MainMenu.h"
```

7.21 ProjektKantor/MainMenu.h File Reference

```
#include "Client.h"
#include "Admin.h"
#include "Border.h"
```

Classes

- class MainMenu

  Klasa reprezentująca główne menu aplikacji.

7.22 MainMenu.h

Go to the documentation of this file.

```
00001 #pragma once
00002
00003 #include "Client.h"
00004 #include "Admin.h"
00005 #include "Border.h"
00006
00013 class MainMenu : public Base, public IUser
00014 {
00015     Client client;
00016     Administrator admin;
00017
00018 public:
00022     MainMenu();
00023
00029     void displayMenu() override;
00030
00038     bool error(int choice, std::string error_msg);
00039 };
```

7.23 ProjektKantor/Transactions.cpp File Reference

```
#include "Transactions.h"
#include <fstream>
#include <sstream>
#include <chrono>
#include <iomanip>
```

7.24 ProjektKantor/Transactions.h File Reference

```
#include <string>
```

Classes

- class Transaction

  Klasa reprezentująca transakcję wymiany walut.

7.25 Transactions.h

Go to the documentation of this file.

```
00001 #pragma once
00002 #include <string>
00003
00010 class Transaction
00011 {
00012 private:
00013     std::string clientName{};
00014     std::string clientSurname{};
00015     std::string currency{};
00016     double amount{};
00017     double price{};
00018     std::string type{};
00019     std::string datetime{};
00020
00021 public:
00025     Transaction() {}
00026
00037     Transaction(const std::string& name, const std::string& surname, const std::string& curr, double
      amt, double pr, const std::string& typ);
00038
00045     void saveToFile(const std::string& file_name, std::string transaction) const;
00046 };
```