

Gliwice, 28.05.2020r.

Laboratorium Programowania Komputerów 4

Temat:

TinyBusiness

Autor: Krzysztof Doniec

Informatyka, sem. 4, gr. 3, sekcja 1

Prowadzący: dr inż. Anna Gorawska

1. Temat

Celem projektu jest program dla magazynu z bazą towaru jaki jest na magazynie i możliwość modyfikowania go poprzez wystawienie paragonu/faktury lub wprowadzenie danych z faktury (podanie ścieżki do dokumentu XML).

Faktury i paragony byłyby wystawiane w formacie XML, do którego chciałbym skorzystać z biblioteki TinyXML. Użytkownik będzie mógł zmieniać i przeglądać informacje o sobie (jako właścicielu), które będą na generowanych dokumentach.

Co do szczegółów to na towar składałaby się: jego nazwa, ilość, cena zakupu, cena sprzedaży oraz stawka VAT. Towar na stanie przed zapisaniem do pliku będzie sortowany po nazwie. Klasa faktura dziedziczyłaby po paragon i posiadała dodatkowo dane klienta (obiekt Party). W czasie wystawiania faktury i paragonu jest możliwość anulowania, które nie zmieni bazy danych w stosunku do sprzed wystawienia. Baza danych będzie zapisywana do pliku binarnego. W niej także będą zapisane informacje o właścicielu.

Program korzysta z następujących parametrów:

- h pomoc programu,
- help pomoc programu,
- d plik z bazą danych (binarny),

2. Analiza tematu

2.1. Klasy

Paragon reprezentuje klasa Receipt, fakturę - Invoice. W ramach mechanizmów obiektowych wykorzystano polimorfizm oraz dziedziczenie w klasach Receipt i dziedziczącej po niej Invoice.

Danymi programu zarządza klasa DataManager przechowująca wektor z obiektami Item oraz dane właściciela w formie obiektu Party.

Klasy DataManager, DailyRaport i Receipt/Invoice są skonstruowane tak, by mogły działać od siebie niezależnie i od użytkownika będzie zależeć w jaki sposób mają współpracować.

Klasa Interface korzysta z pozostałych klas i jest odpowiedzialna za to w jaki sposób końcowy użytkownik może korzystać z programu.

2.2. Biblioteki

W projekcie wykorzystano bibliotekę TinyXML-2. W tym formacie są generowane pliki paragonów i faktur oraz wczytywane do programu towary.

Ponieważ biblioteka ta oferuje wiele więcej możliwości niepotrzebnych w projekcie oraz nie korzysta z mechanizmu wyjątków i klasy string postanowiłem napisać klasę XMLDoc, która mocno upraszcza i umiła korzystanie z biblioteki.

3. Specyfikacja zewnętrzna

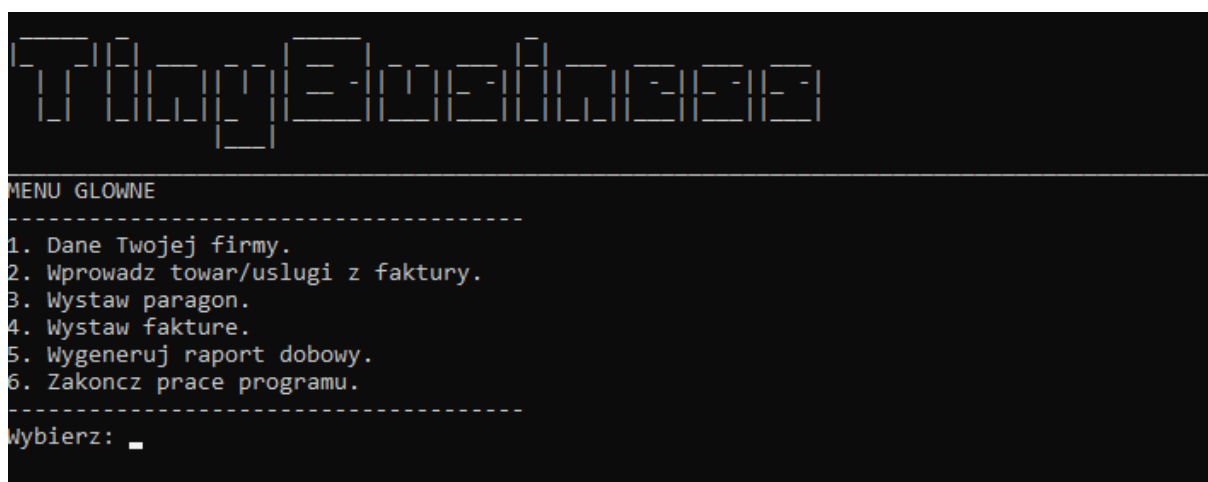
3.1. Instrukcja dla użytkownika

Interfejs programu wykorzystuje konsolę systemową oraz argumenty wiersza poleceń.

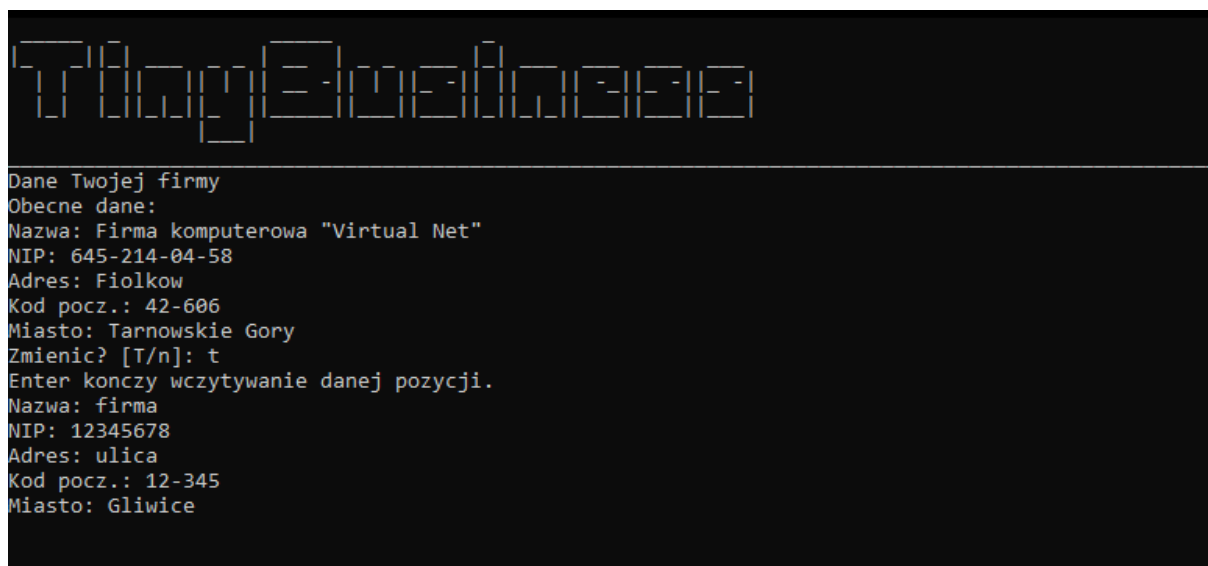
Jeśli użytkownik wpisze błędny format liczby (np. zamiast liczby wprowadzi litery lub wprowadzi liczbę ujemną w miejscu wymagającym nieujemnej) program wyświetli „Wprowadzono nieprawidłowy format. Wprowadz jeszcze raz:”.

Jeśli plik podany w parametrze -d nie istnieje lub jest pusty to program zapyta się o dane właściciela i utworzy jego profil.

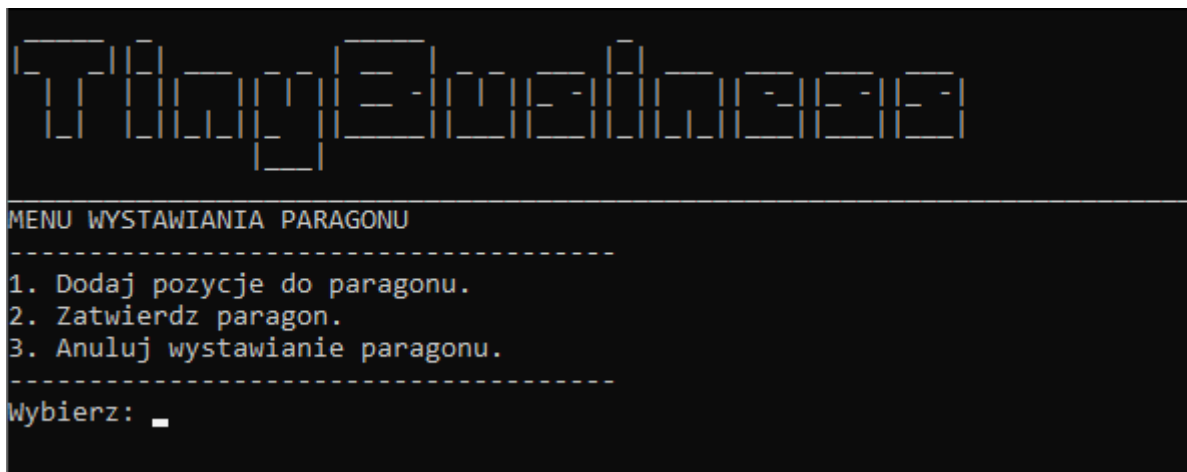
3.2. Przykład działania. Zrzuty ekranu



Rysunek 1 Menu główne programu



Rysunek 2 Okno przeglądania/modyfikowania danych właściciela



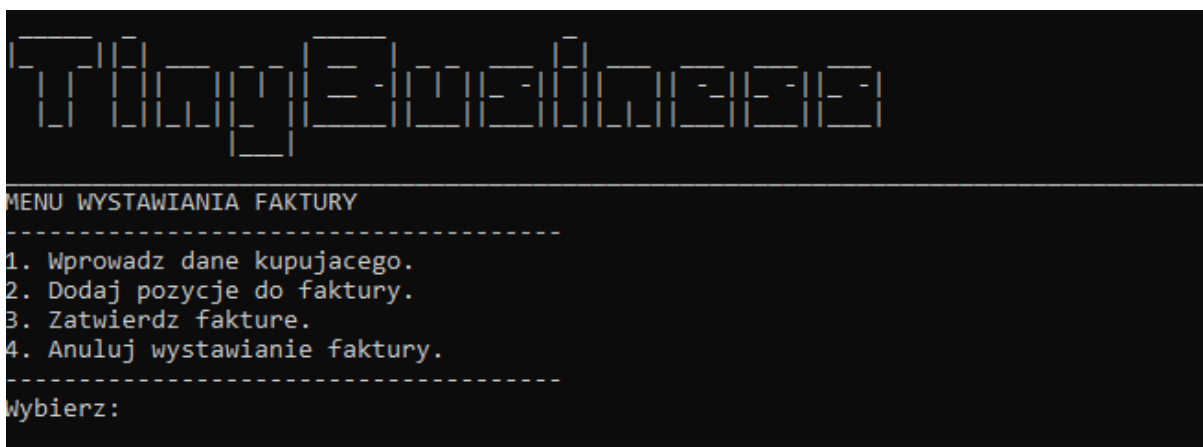
Rysunek 3 Menu wystawiania paragonu

Stan:

lp.	Nazwa	Ilosc	Cena sprz.	VAT
1.	Frytki	50	107374176.00	8%
2.	Kebab cienki	2137	107374176.00	8%
3.	Kebab cienki XXL	3	107374176.00	8%
4.	Kebab cienki bez sur wki	500	107374176.00	8%
5.	Kebab gruby	49994	13.00	8%
6.	Kebab gruby XXL	3	107374176.00	8%
7.	Kebab gruby bez sur wki	13	107374176.00	8%
8.	King Star Kebab Specja é XXXL	14999	23.00	8%
9.	King Star Specja é	1337	107374176.00	8%
10.	Lunch Box	3	107374176.00	8%
11.	Lunch Box + nap j	7	107374176.00	8%
12.	Pojemnik na insulin-ö 1,8 ml	33	6.99	8%
13.	Talerz King Star Specja é	1	107374176.00	8%
14.	Talerz z baranin-ü	666	107374176.00	8%
15.	Talerz z baranin-ü XXL	69	107374176.00	8%
16.	Talerz z kurczakiem	78	107374176.00	8%
17.	Talerz z kurczakiem XXL	915	107374176.00	8%
18.	Talerz z mi-ösem mieszanym	850	107374176.00	8%
19.	Talerz z mi-ösem mieszanym XXL	2	107374176.00	8%

Indeks pozycji do dodania [0 aby anulowac dodawanie]: 12
Ilosc: 33
Cena sprzedazy [PLN]: 6.99_

Rysunek 4 Okno z przykładowym towarem na stanie



Rysunek 5 Menu wystawiania faktury

```

Tłumaczenie
-----
MENU WYSTAWIANIA FAKTURY
-----
1. Wprowadz dane kupującego.
2. Dodaj pozycje do faktury.
3. Zatwierdz fakture.
4. Anuluj wystawianie faktury.
-----
Wybierz: 1
Dane kupującego.
Nazwa: firma
NIP: 12345678
Adres: adres
Kod poczt.: 12-345
Miasto: miasto_

```

Rysunek 6 Okno wprowadzenia danych kupującego

4. Specyfikacja wewnętrzna

Szczegółowy opis typów i funkcji zawarty jest w załączniku.

4.1. Uwzględnione zagadnienia z zajęć tematycznych

4.1.1. Mechanizm wyjątków

Zagnieżdżona klasa XMLDoc::XMLException oraz łapiący ją blok try-catch:

Podany plik nie jest plikiem XML.

```

Tłumaczenie
-----
MENU GŁÓWNE
-----
1. Dane Twojej firmy.
2. Wprowadz towar/usługi z faktury.
3. Wystaw paragon.
4. Wystaw fakturę.
5. Wygeneruj raport dobowy.
6. Zakończ prace programu.
-----
Wybierz: 2
Podaj ścieżkę z fakturą: C:\Users\krzydon683\Desktop\nowy.xml
Błędna zawartość pliku. [ENTER]

```

Rysunek 7 Efekt złapania wyjątku XMLException

4.1.2. Wzorce

W projekcie znajduje się łącznie 6 metod/funkcji które są wzorcami. Przykładowo metoda `template<typename T> T Interface::getNumberFromCin(const char *statement)` zajmuje się odpowiednim odczytaniem danego typu od użytkownika i jego zwrócenie w poprawnej formie.

4.1.3. Kontenery STL

Klasa `DataManager` korzysta z wektora przechowującego pozycje na stanie, natomiast klasa `XMLDoc` korzysta ze stosu do wygodniejszego operowania na zagnieżdżeniu w dokumentach XML.

4.1.4. Algorytmy i iteratory STL

Metoda `sortAndWriteBase(std::string dbName)` klasy `DataManager` korzysta z algorytmu `sort` do posortowania wektora towaru po jego nazwie.

Metody `remove(InputIterator first, InputIterator last)` oraz operator `+=(Item &&c)` klasy `DataManager` korzystają z algorytmu `find` do znalezienia identycznych obiektów składowej `stock` i danym w parametrze przedziałem lub obiektem.

4.2. Indeks klas

4.1 Lista klas

Tutaj znajdują się klasy, struktury, unie i interfejsy wraz z ich krótkimi opisami:

DailyRaport	
Class to generate daily raport	15
DataManager	
Class managing all data saved in binary file	16
Interface	
Creates interface for user	18
Invoice	
Class with invoice data	20
Item	
Selling item data	21
Party	
Class with party data	22
Receipt	
Class with receipt data	23
XMLDoc	
Tinyxml library managment	25
XMLDoc::XMLException	
Exceptions thrown from read documents	28

5. Testowanie

Program został pomyślnie przetestowany pod względem wycieków pamięci za pomocą narzędzia Visual Leak Detector.

6. Uwagi i wnioski

Cały zakres wymagań względem projektu został spełniony.

Dzięki temu projektowi nauczyłem się pisać programy z wykorzystaniem zaawansowanych technik języka C++ na platformy Windows oraz Linux. Nauczyłem się korzystać z zewnętrznej biblioteki TinyXML.

TinyBusiness

Wygenerowano przez Doxygen 1.8.18

1 README	1
2 Indeks przestrzeni nazw	3
2.1 Lista przestrzeni nazw	3
3 Indeks hierarchiczny	5
3.1 Hierarchia klas	5
4 Indeks klas	7
4.1 Lista klas	7
5 Indeks plików	9
5.1 Lista plików	9
6 Dokumentacja przestrzeni nazw	11
6.1 Dokumentacja przestrzeni nazw helpfulness	11
6.1.1 Opis szczegółowy	11
6.1.2 Dokumentacja funkcji	11
6.1.2.1 date()	11
6.1.2.2 hour()	12
6.2 Dokumentacja przestrzeni nazw menusStrings	12
6.2.1 Opis szczegółowy	12
6.2.2 Dokumentacja zmiennych	12
6.2.2.1 invoiceMenu	13
6.2.2.2 logo	13
6.2.2.3 mainMenu	13
6.2.2.4 receiptMenu	13
7 Dokumentacja klas	15
7.1 Dokumentacja klasy DailyRaport	15
7.1.1 Opis szczegółowy	15
7.2 Dokumentacja klasy DataManager	16
7.2.1 Opis szczegółowy	17
7.2.2 Dokumentacja konstruktora i destruktorów	17
7.2.2.1 DataManager()	17
7.2.3 Dokumentacja funkcji składowych	17
7.2.3.1 checkItem()	17
7.2.3.2 loadFromXMLInvoice()	17
7.2.3.3 remove()	18
7.3 Dokumentacja klasy Interface	18
7.3.1 Opis szczegółowy	19
7.3.2 Dokumentacja konstruktora i destruktorów	19
7.3.2.1 Interface()	19
7.4 Dokumentacja klasy Invoice	20

7.4.1 Opis szczegółowy	20
7.4.2 Dokumentacja konstruktora i destruktora	20
7.4.2.1 Invoice()	20
7.5 Dokumentacja klasy Item	21
7.5.1 Opis szczegółowy	22
7.5.2 Dokumentacja funkcji składowych	22
7.5.2.1 show()	22
7.6 Dokumentacja klasy Party	22
7.6.1 Opis szczegółowy	23
7.7 Dokumentacja klasy Receipt	23
7.7.1 Opis szczegółowy	24
7.7.2 Dokumentacja konstruktora i destruktora	24
7.7.2.1 Receipt()	24
7.7.3 Dokumentacja funkcji składowych	24
7.7.3.1 for_eachItem()	25
7.8 Dokumentacja klasy XMLDoc	25
7.8.1 Opis szczegółowy	26
7.8.2 Dokumentacja funkcji składowych	26
7.8.2.1 addElement()	26
7.8.2.2 getFloat()	26
7.8.2.3 getInt()	26
7.8.2.4 getText()	27
7.8.2.5 loadFile()	27
7.8.2.6 saveXML()	27
7.9 Dokumentacja klasy XMLDoc::XMLException	28
7.9.1 Opis szczegółowy	28
8 Dokumentacja plików	29
8.1 Dokumentacja pliku D:/nauka/polsl/PK4/PK4/Projekt/project2/project2/main.cpp	29
8.1.1 Opis szczegółowy	29
8.2 Dokumentacja pliku D:/nauka/polsl/PK4/PK4/Projekt/project2/project2/windowsFunctions.h	29
Indeks	31

Rozdział 1

README

Tutaj umieszczać projekt

TinyBusiness

W ramach projektu chciałbym napisać program dla magazynu z bazą towaru jaki jest na magazynie i modyfikowanie go poprzez wystawienie paragonu/faktury lub wprowadzenie danych z faktury (podanie ścieżki do dokumentu XML).

Bazą danych jest klasa [DataManager](#) przechowująca wektor z obiektami [Item](#) oraz dane właściciela w formie obiektu [Party](#). Paragon reprezentuje klasa [Receipt](#), faktura, dziedzicząca po niej, [Invoice](#). Faktury i paragony byłyby wystawiane w formacie XML, do którego chciałbym skorzystać z biblioteki TinyXML. Użytkownik będzie mógł zmieniać i przeglądać informacje o sobie (jako właścicielu), które będą na generowanych dokumentach.

Co do szczegółów to na towar ([Item](#)) składałaby się: jego nazwa, ilość, cena zakupu, cena sprzedaży oraz stawka VAT. Towar na stanie przed zapisaniem do pliku będzie sortowany po nazwie. Klasa faktura ([Invoice](#)) dziedziczyłaby po paragon ([Receipt](#)) i posiadała dodatkowo dane klienta (obiekt [Party](#)). W czasie wystawiania faktury i paragonu jest możliwość anulowania, które nie zmieni bazy danych w stosunku do sprzed wystawienia. Baza danych będzie zapisywana do pliku binarnego. W niej także będą zapisane informacje o właścicielu. Klasy [BaseData](#) i [Receipt](#)/[Invoice](#) są skonstruowane tak, by mogły działać od siebie niezależnie i od użytkownika będzie zależeć w jaki sposób obie mają współpracować.

Rozdział 2

Indeks przestrzeni nazw

2.1 Lista przestrzeni nazw

Tutaj znajdują się wszystkie udokumentowane przestrzenie nazw wraz z ich krótkimi opisami:

helpfulness	
Useful functions	11
menusStrings	
String literals for program menus	12

Rozdział 3

Indeks hierarchiczny

3.1 Hierarchia klas

Ta lista dziedziczenia posortowana jest z grubsza, choć nie całkowicie, alfabetycznie:

DailyRaport	15
DataManager	16
exception	
XMLDoc::XMLException	28
Interface	18
Item	21
Party	22
Receipt	23
Invoice	20
XMLDoc	25

Rozdział 4

Indeks klas

4.1 Lista klas

Tutaj znajdują się klasy, struktury, unie i interfejsy wraz z ich krótkimi opisami:

DailyRaport		
	Class to generate daily raport	15
DataManager		
	Class managing all data saved in binary file	16
Interface		
	Creates interface for user	18
Invoice		
	Class with invoice data	20
Item		
	Selling item data	21
Party		
	Class with party data	22
Receipt		
	Class with receipt data	23
XMLDoc		
	Tinyxml library managment	25
XMLDoc::XMLException		
	Exceptions thrown from read documents	28

Rozdział 5

Indeks plików

5.1 Lista plików

Tutaj znajduje się lista wszystkich udokumentowanych plików z ich krótkimi opisami:

D:/nauka/polsl/PK4/PK4/Projekt/project2/project2/ dailyRaport.hpp	??
D:/nauka/polsl/PK4/PK4/Projekt/project2/project2/ dataManager.hpp	??
D:/nauka/polsl/PK4/PK4/Projekt/project2/project2/ helpfulness.hpp	??
D:/nauka/polsl/PK4/PK4/Projekt/project2/project2/ interface.hpp	??
D:/nauka/polsl/PK4/PK4/Projekt/project2/project2/ invoice.hpp	??
D:/nauka/polsl/PK4/PK4/Projekt/project2/project2/ item.hpp	??
D:/nauka/polsl/PK4/PK4/Projekt/project2/project2/ main.cpp	29
D:/nauka/polsl/PK4/PK4/Projekt/project2/project2/ menusStringLiterals.h	??
D:/nauka/polsl/PK4/PK4/Projekt/project2/project2/ party.hpp	??
D:/nauka/polsl/PK4/PK4/Projekt/project2/project2/ receipt.hpp	??
D:/nauka/polsl/PK4/PK4/Projekt/project2/project2/ windowsFunctions.h	29
D:/nauka/polsl/PK4/PK4/Projekt/project2/project2/ XMLDoc.hpp	??

Rozdział 6

Dokumentacja przestrzeni nazw

6.1 Dokumentacja przestrzeni nazw helpfulness

Useful functions.

Funkcje

- `template<typename T >`
`std::string toStringPrecision2 (T input)`
Convert numerical value to string with setprecision(2)
- `std::string date (const char delim='\0')`
- `std::string hour (const char delim='\0')`
- `std::istream & readStringBin (std::istream &is, std::string &str)`
- `std::ostream & writeStringBin (std::ostream &os, const std::string &str)`

6.1.1 Opis szczegółowy

Useful functions.

6.1.2 Dokumentacja funkcji

6.1.2.1 date()

```
std::string helpfulness::date (  
    const char delim = '\0' )
```

Zwraca

date as a string

Parametry

<i>delim</i>	delimitate numbers
--------------	--------------------

6.1.2.2 hour()

```
std::string helpfulness::hour (
    const char delim = '\0' )
```

Zwraca

hour as a string

Parametry

<i>delim</i>	delimitate numbers
--------------	--------------------

6.2 Dokumentacja przestrzeni nazw menusStrings

String literals for program menus.

Zmienne

- constexpr const char * [logo](#)
TinyBusiness logo.
- constexpr const char * [mainMenu](#)
Main menu options.
- constexpr const char * [receiptMenu](#)
[Receipt](#) menu options.
- constexpr const char * [invoiceMenu](#)
[Invoice](#) menu options.

6.2.1 Opis szczegółowy

String literals for program menus.

6.2.2 Dokumentacja zmiennych

Rozdział 7

Dokumentacja klas

7.1 Dokumentacja klasy DailyRaport

Class to generate daily raport.

```
#include <dailyRaport.hpp>
```

Metody publiczne

- void `addSum` (double sum, double PTUSum)
Add payments to sum payments.
- bool `generate` ()
Generate daily report file.

Metody prywatne

- void `writeDocumentInfoXML` (XMLDoc &doc) const
- void `writePaymentXML` (XMLDoc &doc) const

Atrybuty prywatne

- long double `totalPayment`
Total daily payment.
- long double `totalPTUAmount`
Total daily PTU payment.

7.1.1 Opis szczegółowy

Class to generate daily raport.

Dokumentacja dla tej klasy została wygenerowana z plików:

- D:/nauka/polsl/PK4/PK4/Projekt/project2/project2/dailyRaport.hpp
- D:/nauka/polsl/PK4/PK4/Projekt/project2/project2/dailyRaport.cpp

7.2 Dokumentacja klasy DataManager

Class managing all data saved in binary file.

```
#include <DataManager.hpp>
```

Metody publiczne

- [DataManager](#) (std::string dbName="db.bin")
- void **createOwner** ()
- const [Party](#) * [pOwner](#) () const
Return pointer to owner data.
- unsigned int [getInvoiceNo](#) ()
Getter number of invoice.
- bool [checkStock](#) () const
Check if there is anything in the stock.
- bool [checkItem](#) (const unsigned int index, const int quantity) const
- const [Item](#) & [operator\[\]](#) (size_t n) const
Access to elements in stock.
- template<typename InputIterator >
void [remove](#) (InputIterator first, InputIterator last)
- void [showStock](#) () const
Show stock to console.
- void **showStockHeader** (const std::streamsize noWidth, const std::streamsize descriptionWidth, const std::streamsize quantityWidth, const std::streamsize spriceWidth, const std::streamsize vatWidth) const
- void **showStockContent** (const std::streamsize noWidth, const std::streamsize descriptionWidth, const std::streamsize quantityWidth, const std::streamsize spriceWidth, const std::streamsize vatWidth) const
- void [sortAndWriteBase](#) (std::string dbName="db.bin")
Sort and write base to bin.
- bool [loadFromXMLInvoice](#) (const std::string &docName)

Metody prywatne

- [DataManager](#) & [operator+=](#) ([Item](#) &&c)
Adding items to base; called from loadFromInvoice.
- void [readBase](#) (std::ifstream &ifs)
Read base from bin; called from constructor.
- void **readMonthAndInvoiceNo** (std::ifstream &ifs)
- void **readStock** (std::ifstream &ifs)
- void [writeMonthAndInvoiceNo](#) (std::ofstream &ofs) const
Write base elements to bin; called from sortAndWriteBase()
- void **writeStock** (std::ofstream &ofs) const

Atrybuty prywatne

- [Party](#) owner
- std::vector< [Item](#) > **stock**
- unsigned int [invoiceNo](#)
Number of next invoice.

7.2.1 Opis szczegółowy

Class managing all data saved in binary file.

7.2.2 Dokumentacja konstruktora i destruktora

7.2.2.1 DataManager()

```
DataManager::DataManager (
    std::string dbName = "db.bin" )
```

Parametry

<i>dbName</i>	name of file with user data
---------------	-----------------------------

7.2.3 Dokumentacja funkcji składowych

7.2.3.1 checkItem()

```
bool DataManager::checkItem (
    const unsigned int index,
    const int quantity ) const [inline]
```

Check whenever item exists in sufficient quantity

Zwracane wartości

<i>false</i>	if not enough quantity
--------------	------------------------

Wyjątki

<i>std::out_of_range</i>	for out-of-range index
--------------------------	------------------------

7.2.3.2 loadFromXMLInvoice()

```
bool DataManager::loadFromXMLInvoice (
    const std::string & docName )
```

Load Items from file given by parametr

Parametry

<i>docName</i>	document name with Items to add
----------------	---------------------------------

7.2.3.3 remove()

```
template<typename InputIterator >
void DataManager::remove (
    InputIterator first,
    InputIterator last ) [inline]
```

Remove range from stock

Parametry

<i>first,last</i>	Forward iterators to the initial and final positions in a sequence with items to remove
-------------------	---

Dokumentacja dla tej klasy została wygenerowana z plików:

- D:/nauka/polsl/PK4/PK4/Projekt/project2/project2/dataManager.hpp
- D:/nauka/polsl/PK4/PK4/Projekt/project2/project2/dataManager.cpp

7.3 Dokumentacja klasy Interface

Creates interface for user.

```
#include <interface.hpp>
```

Metody publiczne

- [Interface](#) (std::string dFileName)
- void [mainMenu](#) ()
main loop

Metody prywatne

- char [getCharFromCin](#) () const
- void [mainMenuSwitch](#) (char choice, bool &end)
- void [changeOwnerData](#) ()
- void [addFromInvoice](#) ()
Add items from invoice to dataManager.
- void [addItemsFromInvoice](#) ()
- void [receiptIssueMenu](#) ()
Receipt menu.

- void **invoiceIssueMenu** ()
Invoice menu.
- void **receiptIssueMenuSwitch** (char choice, bool &end, **Receipt** &re)
- void **invoiceIssueMenuSwitch** (char choice, bool &end, bool &buyerCreated, **Invoice** &inv)
- void **buyerNotCreated** () const
- bool **checkStock** () const
Check if there is anything in the stock.
- void **addItem** (**Receipt** &re)
Add chosen by user item from dataManager to receipt/invoice.
- void **addItemFromUser** (**Receipt** &re)
- void **showStock** () const
- void **checkAndRepairItemQuantity** (unsigned int index, unsigned int quantity) const
- void **confirmDocument** (**Receipt** &re)
- template<typename T >
T **getNumberFromCin** (const char *statement) const
- template<typename T >
void **checkAndRepairCin** (T &t) const
- void **dailyReport** ()

Atrybuty prywatne

- std::string **dFileName**
- **DataManager** dmngr
- **DailyRaport** dRaport

7.3.1 Opis szczegółowy

Creates interface for user.

7.3.2 Dokumentacja konstruktora i destruktor

7.3.2.1 Interface()

```
Interface::Interface (
    std::string dFileName ) [inline]
```

Parametry

<i>dFileName</i>	name of file with user data
------------------	-----------------------------

Dokumentacja dla tej klasy została wygenerowana z plików:

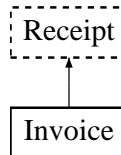
- D:/nauka/polsl/PK4/PK4/Projekt/project2/project2/interface.hpp
- D:/nauka/polsl/PK4/PK4/Projekt/project2/project2/interface.cpp

7.4 Dokumentacja klasy Invoice

Class with invoice data.

```
#include <invoice.hpp>
```

Diagram dziedziczenia dla Invoice



Metody publiczne

- `Invoice` (const `Party` *`owner`, unsigned int `invoiceNo`)
- void `createBuyer` ()
Set contractor from user.
- virtual bool `createDocument` () const override
Create XML document.

Metody chronione

- void `addDocInfoToDocument` (`XMLDoc` &doc) const
- void `writeDocInfoXML` (`XMLDoc` &doc) const
- void `addBuyerDataToDocument` (`XMLDoc` &doc) const
- void `addItemsToDocument` (`XMLDoc` &doc) const
- void `addSumToDocument` (`XMLDoc` &doc) const

Atrybuty prywatne

- `Party` `contractor`
- unsigned int `invoiceNo`
Number of invoice.

7.4.1 Opis szczegółowy

Class with invoice data.

7.4.2 Dokumentacja konstruktora i destruktoru

7.4.2.1 Invoice()

```

Invoice::Invoice (
    const Party * owner,
    unsigned int invoiceNo ) [inline]
  
```

Parametry

<i>owner</i>	pointer to owner data
<i>invoiceNo</i>	number of invoice

Dokumentacja dla tej klasy została wygenerowana z plików:

- D:/nauka/polsl/PK4/PK4/Projekt/project2/project2/invoice.hpp
- D:/nauka/polsl/PK4/PK4/Projekt/project2/project2/invoice.cpp

7.5 Dokumentacja klasy Item

Selling item data.

```
#include <item.hpp>
```

Metody publiczne

- int **getQuantity** () const
- float **getPurchasePrice** () const
- float **getSalesPrice** () const
- double **getNettoPrice** () const
- int **getVAT** () const
- void **setSaleData** (unsigned int qty, float sPrice)
- void **setPurchasePrice** (float pPrice)
- void **setSalesPrice** (float sPrice)
- **Item** & **operator+=** (unsigned int ui)
- **Item** & **operator-=** (unsigned int ui)
- void **show** (const std::streamsize descriptionWidth, const std::streamsize quantityWidth, const std::streamsize spriceWidth, const std::streamsize vatWidth) const
- std::istream & **read** (std::istream &is)
- std::ostream & **write** (std::ostream &os) const
- void **writeXML** ([XMLDoc](#) &doc) const
- void **writeNettoXML** ([XMLDoc](#) &doc) const
- void **readXML** ([XMLDoc](#) &doc)

Atrybuty prywatne

- std::string [description](#)
Description of [Item](#).
- int **quantity**
- int **vat**
- float **salesPrice**
- float [purchasePrice](#)
Brutto prices.

Przyjaciele

- bool **operator==** (const [Item](#) &i1, const [Item](#) &i2)
- bool **operator<** (const [Item](#) &i1, const [Item](#) &i2)

7.5.1 Opis szczegółowy

Selling item data.

7.5.2 Dokumentacja funkcji składowych

7.5.2.1 show()

```
void Item::show (
    const std::streamsize descriptionWidth,
    const std::streamsize quantityWidth,
    const std::streamsize spriceWidth,
    const std::streamsize vatWidth ) const
```

Parametry

<i>descriptionWidth,quantityWidth,spriceWidth,vatWidth</i>	Set width of specified column
--	-------------------------------

Dokumentacja dla tej klasy została wygenerowana z plików:

- D:/nauka/polsl/PK4/PK4/Projekt/project2/project2/item.hpp
- D:/nauka/polsl/PK4/PK4/Projekt/project2/project2/item.cpp

7.6 Dokumentacja klasy Party

Class with party data.

```
#include <party.hpp>
```

Metody publiczne

- void [createParty](#) ()
Set all members from user.
- std::istream & **read** (std::istream &is)
- std::ostream & **write** (std::ostream &os) const
- void **writeXML** ([XMLDoc](#) &doc) const

Atrybuty prywatne

- `std::string name`
- `std::string NIP`
- `std::string address`
- `std::string postcode`
- `std::string city`

Przyjaciele

- `std::ostream & operator<< (std::ostream &os, const Party &p)`

7.6.1 Opis szczegółowy

Class with party data.

Dokumentacja dla tej klasy została wygenerowana z plików:

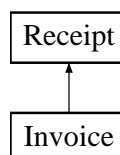
- `D:/nauka/polsl/PK4/PK4/Projekt/project2/project2/party.hpp`
- `D:/nauka/polsl/PK4/PK4/Projekt/project2/project2/party.cpp`

7.7 Dokumentacja klasy Receipt

Class with receipt data.

```
#include <receipt.hpp>
```

Diagram dziedziczenia dla Receipt



Metody publiczne

- `Receipt (const Party *owner)`
- `double getSum ()`
- `double getPTUSum ()`
- `void pushItem (const Item &item, unsigned int quantity, float price)`
Push item to items.
- `auto cbegin ()`
const_iterator for items forwarding
- `auto cend ()`
const_iterator for items forwarding
- `virtual bool createDocument () const`
Create XML document.

Metody chronione

- void **addSellerDataToDocument** (XMLDoc &doc) const
- void **addItemsToDocument** (XMLDoc &doc) const
- void **addSumToDocument** (XMLDoc &doc) const
- void **writeSellerXML** (XMLDoc &doc) const
Writing seller to XML.
- template<typename Function >
Function **for_eachItem** (XMLDoc &doc, Function fn) const
Apply writing type Items to XML.
- void **writeSumXML** (XMLDoc &doc) const
Writing sum to XML.
- void **writeNettoSumXML** (XMLDoc &doc) const

Atrybuty prywatne

- const Party * **owner**
Const pointer to owner data.
- std::vector< Item > **items**
- double **sum**
- double **PTUSum**

7.7.1 Opis szczegółowy

Class with receipt data.

7.7.2 Dokumentacja konstruktora i destruktor

7.7.2.1 Receipt()

```
Receipt::Receipt (
    const Party * owner ) [inline]
```

Parametry

<i>owner</i>	pointer to owner data
--------------	-----------------------

7.7.3 Dokumentacja funkcji składowych

7.7.3.1 for_eachItem()

```
template<typename Function >
Function Receipt::for_eachItem (
    XMLDoc & doc,
    Function fn ) const [inline], [protected]
```

Apply writing type Items to XML.

Zwraca

fn

Dokumentacja dla tej klasy została wygenerowana z plików:

- D:/nauka/polsl/PK4/PK4/Projekt/project2/project2/receipt.hpp
- D:/nauka/polsl/PK4/PK4/Projekt/project2/project2/receipt.cpp

7.8 Dokumentacja klasy XMLDoc

Tinyxml library managment.

```
#include <XMLDoc.hpp>
```

Komponenty

- class [XMLException](#)
Exceptions thrown from read documents.

Metody publiczne

- void **newDoc** (const char *name)
- template<typename T >
void [addElement](#) (const char *TextNewElement, const T SetText)
*See [addElement\(const char *TextNewElement\)](#)*
- void [addElement](#) (const char *TextNewElement)
- void [insertChild](#) ()
*See [addElement\(const char *TextNewElement\)](#)*
- bool [saveXML](#) (const char *docName)
- bool [loadFile](#) (const char *docName)
- bool **childElement** (const char *text)
- bool **nextElement** (const char *text)
- std::string [getText](#) (const char *text) const
- int [getInt](#) (const char *text) const
- float [getFloat](#) (const char *text) const

Atrybuty prywatne

- tinyxml2::XMLDocument **doc**
- tinyxml2::XMLNode * **pRoot**
- std::stack< tinyxml2::XMLElement * > **pElements**
Stack of pointers to element.

7.8.1 Opis szczegółowy

Tinyxml library managment.

7.8.2 Dokumentacja funkcji składowych

7.8.2.1 addElement()

```
void XMLDoc::addElement (
    const char * TextNewElement ) [inline]
```

Uwaga

Need [insertChild\(\)](#) after and some [addElement\(const char *TextNewElement, const T SetText\)](#) between.

Example:

```
XML::addElement(Śuma
);
XML::addElement(Śuma_PLN
, "21.37
);
XML::insertChild();
```

7.8.2.2 getFloat()

```
float XMLDoc::getFloat (
    const char * text ) const
```

Wyjątki

<code>XMLException("XML_ERROR_PARSING_ELEMENT")</code>	invalid file content
--	----------------------

7.8.2.3 getInt()

```
int XMLDoc::getInt (
    const char * text ) const
```

Wyjątki

<code>XMLException("XML_ERROR_PARSING_ELEMENT")</code>	invalid file content
--	----------------------

7.8.2.4 getText()

```
std::string XMLDoc::getText (
    const char * text ) const
```

Wyjątki

<code>XMLException("XML_ERROR_PARSING_ELEMENT")</code>	invalid file content
--	----------------------

7.8.2.5 loadFile()

```
bool XMLDoc::loadFile (
    const char * docName )
```

Wyjątki

<code>XMLException("XML_ERROR_PARSING_TEXT")</code>	invalid file content
---	----------------------

Zwracane wartości

<code>false</code>	file doesnt exist or is empty
--------------------	-------------------------------

7.8.2.6 saveXML()

```
bool XMLDoc::saveXML (
    const char * docName )
```

Zwracane wartości

<code>false</code>	Cant save file.
--------------------	-----------------

Dokumentacja dla tej klasy została wygenerowana z plików:

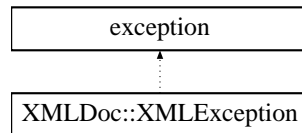
- D:/nauka/polsl/PK4/PK4/Projekt/project2/project2/XMLDoc.hpp
- D:/nauka/polsl/PK4/PK4/Projekt/project2/project2/XMLDoc.cpp

7.9 Dokumentacja klasy XMLDoc::XMLException

Exceptions thrown from read documents.

```
#include <XMLDoc.hpp>
```

Diagram dziedziczenia dla XMLDoc::XMLException



Metody publiczne

- **XMLException** (const char *message)
- char const * **what** () const override

Atrybuty prywatne

- char **msg** [80]
Buffer for warning text.

7.9.1 Opis szczegółowy

Exceptions thrown from read documents.

Dokumentacja dla tej klasy została wygenerowana z pliku:

- D:/nauka/polsl/PK4/PK4/Projekt/project2/project2/XMLDoc.hpp

Rozdział 8

Dokumentacja plików

8.1 Dokumentacja pliku

D:/nauka/polsl/PK4/PK4/Projekt/project2/project2/main.cpp

```
#include <iostream>
#include <cstring>
#include "interface.hpp"
```

Funkcje

- void `help` ()
Called by -h or -help.
- int `main` (int argc, char *argv[])

8.1.1 Opis szczegółowy

Autor

Krzysztof Doniec

Data

2020

8.2 Dokumentacja pliku D:/nauka/polsl/PK4/PK4/↵ Projekt/project2/project2/windowsFunctions.h

Funkcje

- void `ClearScreen` (void)
Clear CMD or console screen.

Indeks

- addElement
 - XMLDoc, [26](#)
- checkItem
 - DataManager, [17](#)
- D:/nauka/polsl/PK4/PK4/Projekt/project2/project2/main.cpp,
[29](#)
- D:/nauka/polsl/PK4/PK4/Projekt/project2/project2/windowsPartitions.h,
[29](#)
- DailyRaport, [15](#)
- DataManager, [16](#)
 - checkItem, [17](#)
 - DataManager, [17](#)
 - loadFromXMLInvoice, [17](#)
 - remove, [18](#)
- date
 - helpfulness, [11](#)
- for_eachItem
 - Receipt, [24](#)
- getFloat
 - XMLDoc, [26](#)
- getInt
 - XMLDoc, [26](#)
- getText
 - XMLDoc, [27](#)
- helpfulness, [11](#)
 - date, [11](#)
 - hour, [12](#)
- hour
 - helpfulness, [12](#)
- Interface, [18](#)
 - Interface, [19](#)
- Invoice, [20](#)
 - Invoice, [20](#)
- invoiceMenu
 - menusStrings, [12](#)
- Item, [21](#)
 - show, [22](#)
- loadFile
 - XMLDoc, [27](#)
- loadFromXMLInvoice
 - DataManager, [17](#)
- logo
 - menusStrings, [13](#)
- mainMenu
 - menusStrings, [13](#)
- menusStrings, [12](#)
 - invoiceMenu, [12](#)
 - logo, [13](#)
 - mainMenu, [13](#)
 - receiptMenu, [13](#)
- Party.h,
[28](#)
- Receipt, [23](#)
 - for_eachItem, [24](#)
 - Receipt, [24](#)
- receiptMenu
 - menusStrings, [13](#)
- remove
 - DataManager, [18](#)
- saveXML
 - XMLDoc, [27](#)
- show
 - Item, [22](#)
- XMLDoc, [25](#)
 - addElement, [26](#)
 - getFloat, [26](#)
 - getInt, [26](#)
 - getText, [27](#)
 - loadFile, [27](#)
 - saveXML, [27](#)
- XMLDoc::XMLException, [28](#)