Aplikacja do zarządzania biznesem świeczkowym

Wygenerowano przez Doxygen 1.10.0

Rozdział 1

Indeks hierarchiczny

1.1 Hierarchia klas

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

Rozdział 2

Indeks klas

2.1 Lista klas

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

Rozdział 3

Indeks plików

3.1 Lista plików

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

Rozdział 4

Dokumentacja klas

4.1 Dokumentacja klasy BOM

Klasa dziedzicząca BOM reprezentująca element struktury materiałowej.

`#include <Data.h>`

Dziedziczy Item.

Metody publiczne

- **BOM** ()

  Konstruktor domyślny klasy BOM.
- BOM (int id___, const string &n__, int q__, const string &u__, const string &m, double c)

  Konstruktor parametryczny klasy BOM.
- ∼**BOM** ()

  Destruktor klasy BOM.
- string getMaterial ()

  Pobiera materiał elementu BOM.
- void setMaterial (const string &a)

  Ustawia materiał elementu BOM.
- double getCost ()

  Pobiera koszt elementu BOM.
- void setCost (double a)

  Ustawia koszt elementu BOM.
- void display () override

  Wyświetla informacje o elemencie BOM. Implementuje funkcję `display` z klasy `Item`.

Metody publiczne dziedziczone z Item

- **Item** ()

  Konstruktor domyślny klasy `Item`.
- Item (int id_, const string &n, int q, const string &u)

  Konstruktor parametryczny klasy `Item`.
- ~**Item** ()

  Destruktor klasy `Item`.
- int getId ()

  Pobiera identyfikator przedmiotu.
- void setId (int a)

  Ustawia identyfikator przedmiotu.
- string getName ()

  Pobiera nazwę przedmiotu.
- void setName (const string &a)

  Ustawia nazwę przedmiotu.
- int getQuantity ()

  Pobiera ilość przedmiotu.
- void setQuantity (int a)

  Ustawia ilość przedmiotu.
- string getUnit ()

  Pobiera jednostkę miary przedmiotu.
- void setUnit (const string &a)

  Ustawia jednostkę miary przedmiotu.

4.1.1 Opis szczegółowy

Klasa dziedzicząca `BOM` reprezentująca element struktury materiałowej.

Klasa `BOM` rozszerza klasę `Item` o dodatkowe pola `material` i `cost` oraz implementuje funkcję `display`.

4.1.2 Dokumentacja konstruktora i destruktora

4.1.2.1 BOM()

```
BOM::BOM (
            int id___,
            const string & n__,
            int q__,
            const string & u__,
            const string & m,
            double c )
```

Konstruktor parametryczny klasy `BOM`.

Parametry

id_↩ ___	Identyfikator elementu BOM.
n_↩ _	Nazwa elementu BOM.
q_↩ _	Ilość elementu BOM.
u_↩ _	Jednostka miary elementu BOM.
m	Materiał elementu BOM.

4.1.3 Dokumentacja funkcji składowych

4.1.3.1 display()

```
void BOM::display ( )  [override], [virtual]
```

Wyświetla informacje o elemencie BOM. Implementuje funkcję `display` z klasy `Item`.

Implementuje Item.

4.1.3.2 getCost()

```
double BOM::getCost ( )
```

Pobiera koszt elementu BOM.

Zwraca

Koszt elementu BOM.

4.1.3.3 getMaterial()

```
string BOM::getMaterial ( )
```

Pobiera materiał elementu BOM.

Zwraca

Materiał elementu BOM.

4.1.3.4 setCost()

```
void BOM::setCost (
            double a )
```

Ustawia koszt elementu BOM.

Parametry

a	Nowy koszt elementu BOM.

4.1.3.5 setMaterial()

```
void BOM::setMaterial (
            const string & a )
```

Ustawia materiał elementu BOM.

Parametry

a	Nowy materiał elementu BOM.

Dokumentacja dla tej klasy została wygenerowana z plików:

- C:/Users/majak/CLionProjects/untitled1/Data.h
- C:/Users/majak/CLionProjects/untitled1/Data.cpp

4.2 Dokumentacja klasy FileUtilise

Klasa odpowiedzialna za operacje na plikach, takie jak wczytywanie i zapisywanie danych magazynu.

`#include <Logic.h>`

Metody publiczne

- **FileUtilise** ()

  Konstruktor domyślny klasy `FileUtilise`.
- ∼**FileUtilise** ()

  Destruktor klasy `FileUtilise`.
- void loadFromFile (const string &filename, vector< Item * > &items)

  Wczytuje dane magazynu z pliku.
- void saveToFile (const string &filename, const vector< Item * > &items)

  Zapisuje dane magazynu do pliku.

4.2.1 Opis szczegółowy

Klasa odpowiedzialna za operacje na plikach, takie jak wczytywanie i zapisywanie danych magazynu.

4.2.2 Dokumentacja funkcji składowych

4.2.2.1 loadFromFile()

```
void FileUtilise::loadFromFile (
            const string & filename,
            vector< Item * > & items )
```

Wczytuje dane magazynu z pliku.

Parametry

filename	Nazwa pliku, z którego mają być wczytane dane.
items	Referencja do wektora przechowującego wskaźniki do obiektów `Item`.

4.2.2.2 saveToFile()

```
void FileUtilise::saveToFile (
            const string & filename,
            const vector< Item * > & items )
```

Zapisuje dane magazynu do pliku.

Parametry

filename	Nazwa pliku, do którego mają być zapisane dane.
items	Referencja do wektora przechowującego wskaźniki do obiektów `Item`.

Dokumentacja dla tej klasy została wygenerowana z plików:

- C:/Users/majak/CLionProjects/untitled1/Logic.h
- C:/Users/majak/CLionProjects/untitled1/Logic.cpp

4.3 Dokumentacja klasy Interface

Klasa odpowiedzialna za interakcję z użytkownikiem.

```
#include <Interface.h>
```

Metody publiczne

- void run ()

  Główna funkcja uruchamiająca interfejs użytkownika i kontrolująca przepływ programu.
- void displayMenu ()

  Funkcja odpowiedzialna za wyświetlenie menu głównego programu.

4.3.1 Opis szczegółowy

Klasa odpowiedzialna za interakcję z użytkownikiem.

Klasa `Interface` zarządza główną pętlą programu, wyświetla menu oraz przetwarza wybory użytkownika. Umożliwia komunikację między użytkownikiem a funkcjami programu, takimi jak zarządzanie produktami i operacje na plikach.

4.3.2 Dokumentacja funkcji składowych

4.3.2.1 displayMenu()

```
void Interface::displayMenu ( )
```

Funkcja odpowiedzialna za wyświetlenie menu głównego programu.

Metoda `displayMenu()` prezentuje użytkownikowi dostępne opcje programu, takie jak dodawanie produktu, usuwanie produktu, dodawanie BOM do odpowiedniego produktu, aktualizacja stanu magazynowego, wczytywanie i zapisywanie danych oraz wyjście z programu.

4.3.2.2 run()

```
void Interface::run ( )
```

Główna funkcja uruchamiająca interfejs użytkownika i kontrolująca przepływ programu.

Metoda `run()` inicjuje pętlę główną programu, wyświetla menu i przetwarza wybory użytkownika, wywołując odpowiednie funkcje z klas `ItemUtilise` i `FileUtilise`.

Dokumentacja dla tej klasy została wygenerowana z plików:

- C:/Users/majak/CLionProjects/untitled1/Interface.h
- C:/Users/majak/CLionProjects/untitled1/Interface.cpp

4.4 Dokumentacja klasy Item

Klasa bazowa `Item` dla produktów i elementów BOM.

```
#include <Data.h>
```

Dziedziczona przez BOM i Product.

Metody publiczne

- **Item** ()

  Konstruktor domyślny klasy `Item`.
- Item (int id_, const string &n, int q, const string &u)

  Konstruktor parametryczny klasy `Item`.
- ~**Item** ()

  Destruktor klasy `Item`.
- int getId ()

  Pobiera identyfikator przedmiotu.
- void setId (int a)

  Ustawia identyfikator przedmiotu.
- string getName ()

  Pobiera nazwę przedmiotu.
- void setName (const string &a)

  Ustawia nazwę przedmiotu.
- int getQuantity ()

  Pobiera ilość przedmiotu.
- void setQuantity (int a)

  Ustawia ilość przedmiotu.
- string getUnit ()

  Pobiera jednostkę miary przedmiotu.
- void setUnit (const string &a)

  Ustawia jednostkę miary przedmiotu.
- virtual void display ()=0

  Czysta funkcja wirtualna do wyświetlania informacji o przedmiocie. Funkcja ta musi być zaimplementowana w klasach dziedziczących.

4.4.1 Opis szczegółowy

Klasa bazowa `Item` dla produktów i elementów BOM.

Klasa `Item` definiuje wspólne właściwości i metody dla wszystkich produktów i elementów BOM. Z tej klasy dziedziczą klasy `Product` oraz `BOM`.

4.4.2 Dokumentacja konstruktora i destruktora

4.4.2.1 Item()

```
Item::Item (
            int id_,
            const string & n,
            int q,
            const string & u )
```

Konstruktor parametryczny klasy `Item`.

Parametry

*id_*	Identyfikator przedmiotu.
n	Nazwa przedmiotu.
q	Ilość przedmiotu.
u	Jednostka miary przedmiotu.

4.4.3 Dokumentacja funkcji składowych

4.4.3.1 display()

```
virtual void Item::display ( )  [pure virtual]
```

Czysta funkcja wirtualna do wyświetlania informacji o przedmiocie. Funkcja ta musi być zaimplementowana w klasach dziedziczących.

Implementowany w Product i BOM.

4.4.3.2 getId()

```
int Item::getId ( )
```

Pobiera identyfikator przedmiotu.

Zwraca

Identyfikator przedmiotu.

4.4.3.3 getName()

```
string Item::getName ( )
```

Pobiera nazwę przedmiotu.

Zwraca

Nazwa przedmiotu.

4.4.3.4 getQuantity()

```
int Item::getQuantity ( )
```

Pobiera ilość przedmiotu.

Zwraca

Ilość przedmiotu.

4.4.3.5 getUnit()

```
string Item::getUnit ( )
```

Pobiera jednostkę miary przedmiotu.

Zwraca

Jednostka miary przedmiotu.

4.4.3.6 setId()

```
void Item::setId (
            int a )
```

Ustawia identyfikator przedmiotu.

Parametry

a	Nowy identyfikator przedmiotu.

4.4.3.7 setName()

```
void Item::setName (
            const string & a )
```

Ustawia nazwę przedmiotu.

Parametry

a	Nowa nazwa przedmiotu.

4.4.3.8 setQuantity()

```
void Item::setQuantity (
            int a )
```

Ustawia ilość przedmiotu.

Parametry

a	Nowa ilość przedmiotu.

4.4.3.9 setUnit()

```
void Item::setUnit (
            const string & a )
```

Ustawia jednostkę miary przedmiotu.

Parametry

a	Nowa jednostka miary przedmiotu.

Dokumentacja dla tej klasy została wygenerowana z plików:

- C:/Users/majak/CLionProjects/untitled1/Data.h
- C:/Users/majak/CLionProjects/untitled1/Data.cpp

4.5 Dokumentacja klasy ItemUtilise

Klasa odpowiedzialna za zarządzanie produktami i operacjami magazynowymi.

`#include <Logic.h>`

Metody publiczne

- **ItemUtilise** ()

  Konstruktor domyślny klasy `ItemUtilise`.
- ∼**ItemUtilise** ()

  Destruktor klasy `ItemUtilise`. Usuwa dynamicznie alokowane obiekty przechowywane w wektorze `items`.
- void addItem (Item ∗item)

  Dodaje nowy produkt do magazynu.

- void removeItem (int id)

  Usuwa produkt z magazynu na podstawie jego identyfikatora.
- void updateQuantity (int id, int quantity)

  Aktualizuje ilość produktu w magazynie.
- void **displayInventory** ()

  Wyświetla zawartość magazynu. Przechodzi przez wszystkie produkty w magazynie i wywołuje ich metodę `display()`.
- void **checkStockLevels** ()

  Sprawdza poziomy zapasów produktów w magazynie. Informuje, które produkty wymagają uzupełnienia zapasów.
- void **checkOrders** ()

  Przetwarza zamówienia klientów i aktualizuje ilości produktów w magazynie. Pozwala użytkownikowi wprowadzać identyfikatory produktów i ilości, aż do zakończenia wprowadzania danych.
- vector< Item * > & getItems ()

  Zwraca referencję do wektora `items`.

4.5.1 Opis szczegółowy

Klasa odpowiedzialna za zarządzanie produktami i operacjami magazynowymi.

4.5.2 Dokumentacja funkcji składowych

4.5.2.1 addItem()

```
void ItemUtilise::addItem (
            Item * item )
```

Dodaje nowy produkt do magazynu.

Parametry

item	Wskaźnik do obiektu `Item`, który ma być dodany.

4.5.2.2 getItems()

```
vector< Item * > & ItemUtilise::getItems ( )
```

Zwraca referencję do wektora `items`.

Zwraca

Referencja do wektora przechowującego wskaźniki do obiektów `Item`.

4.5.2.3 removeItem()

```
void ItemUtilise::removeItem (
            int id )
```

Usuwa produkt z magazynu na podstawie jego identyfikatora.

Parametry

id	Identyfikator produktu, który ma być usunięty.

4.5.2.4 updateQuantity()

```
void ItemUtilise::updateQuantity (
            int id,
            int quantity )
```

Aktualizuje ilość produktu w magazynie.

Parametry

id	Identyfikator produktu.
quantity	Nowa ilość produktu.

Dokumentacja dla tej klasy została wygenerowana z plików:

- C:/Users/majak/CLionProjects/untitled1/Logic.h
- C:/Users/majak/CLionProjects/untitled1/Logic.cpp

4.6 Dokumentacja klasy Product

Klasa dziedzicząca `Product` reprezentująca produkt.

`#include <Data.h>`

Dziedziczy Item.

Metody publiczne

- **Product** ()

  Konstruktor domyślny klasy `Product`.
- Product (int id__, const string &n_, int q_, const string &u_, double p)

  Konstruktor parametryczny klasy `Product`.
- ~**Product** ()

  Destruktor klasy `Product`.
- double getPrice ()

  Pobiera cenę produktu.
- void setPrice (double a)

  Ustawia cenę produktu.
- void display () override

  Wyświetla informacje o produkcie. Implementuje funkcję `display` z klasy `Item`.

Metody publiczne dziedziczone z Item

- **Item** ()

  Konstruktor domyślny klasy `Item`.
- Item (int id_, const string &n, int q, const string &u)

  Konstruktor parametryczny klasy `Item`.
- ~**Item** ()

  Destruktor klasy `Item`.
- int getId ()

  Pobiera identyfikator przedmiotu.
- void setId (int a)

  Ustawia identyfikator przedmiotu.
- string getName ()

  Pobiera nazwę przedmiotu.
- void setName (const string &a)

  Ustawia nazwę przedmiotu.
- int getQuantity ()

  Pobiera ilość przedmiotu.
- void setQuantity (int a)

  Ustawia ilość przedmiotu.
- string getUnit ()

  Pobiera jednostkę miary przedmiotu.
- void setUnit (const string &a)

  Ustawia jednostkę miary przedmiotu.

4.6.1 Opis szczegółowy

Klasa dziedzicząca `Product` reprezentująca produkt.

Klasa `Product` rozszerza klasę `Item` o dodatkowe pole `price` i implementuje funkcję `display`.

4.6.2 Dokumentacja konstruktora i destruktora

4.6.2.1 Product()

```
Product::Product (
            int id__,
            const string & n_,
            int q_,
            const string & u_,
            double p )
```

Konstruktor parametryczny klasy `Product`.

Parametry

*id_*	Identyfikator produktu.
*n_*	Nazwa produktu.
*q_*	Ilość produktu.
*u_*	Jednostka miary produktu.
p	Cena produktu.

4.6.3 Dokumentacja funkcji składowych

4.6.3.1 display()

```
void Product::display ( )  [override], [virtual]
```

Wyświetla informacje o produkcie. Implementuje funkcję `display` z klasy `Item`.

Implementuje Item.

4.6.3.2 getPrice()

```
double Product::getPrice ( )
```

Pobiera cenę produktu.

Zwraca

Cena produktu.

4.6.3.3 setPrice()

```
void Product::setPrice (
            double a )
```

Ustawia cenę produktu.

Parametry

a	Nowa cena produktu.

Dokumentacja dla tej klasy została wygenerowana z plików:

- C:/Users/majak/CLionProjects/untitled1/Data.h
- C:/Users/majak/CLionProjects/untitled1/Data.cpp

Rozdział 5

Dokumentacja plików

5.1 Data.h

```
00001 #pragma once
00002 #include <iostream>
00003 #include <vector>
00004 #include <string>
00005 #include <fstream>
00006
00007 using namespace std;
00008
00017 class Item {
00018
00019 private:
00023     int id;
00027     string name;
00031     int quantity;
00035     string unit;
00036
00037 public:
00038
00042     Item();
00050     Item(int id_, const string& n, int q, const string& u);
00054     ~Item();
00055
00060     int getId();
00065     void setId(int a);
00066
00071     string getName();
00072
00077     void setName(const string& a);
00078
00083     int getQuantity();
00084
00089     void setQuantity(int a);
00090
00095     string getUnit();
00100     void setUnit(const string& a);
00101
00106     virtual void display() = 0;
00107 };
00108
00115 class Product : public Item {
00116
00117 private:
00121     double price;
00122 public:
00123
00127     Product();
00136     Product(int id__, const string& n_, int q_, const string& u_, double p);
00140     ~Product();
00141
00146     double getPrice();
00151     void setPrice(double a);
00152
00157     void display() override;
00158 };
00159
00160
00168 class BOM : public Item {
00169
```

```
00170 private:
00174     string material;
00178     double cost;
00179 public:
00180
00184     BOM();
00194     BOM(int id___, const string& n__, int q__, const string& u__, const string& m, double c);
00198     ~BOM();
00199
00204     string getMaterial();
00209     void setMaterial(const string& a);
00210
00215     double getCost();
00220     void setCost(double a);
00225     void display() override;
00226 };
```

5.2 Interface.h

```
00001 #pragma once
00002 #include "Logic.h"
00003
00012 class Interface
00013 {
00019     ItemUtilise iu;
00025     FileUtilise fu;
00026
00030     int choice;
00031
00032 public:
00039     void run();
00046     void displayMenu();
00047 };
```

5.3 Logic.h

```
00001 #pragma once
00002 #define LOGIC_H
00003
00004 #include "Data.h"
00005 #include <iostream>
00006 #include <vector>
00007 #include <string>
00008 #include <fstream>
00009 using namespace std;
00010
00015 class ItemUtilise {
00016 private:
00020     vector<Item*> items;
00021 public:
00025     ItemUtilise();
00030     ~ItemUtilise();
00031
00036     void addItem(Item* item);
00037
00042     void removeItem(int id);
00048     void updateQuantity(int id, int quantity);
00053     void displayInventory();
00058     void checkStockLevels();
00063     void checkOrders();
00068     vector<Item*>& getItems();
00069
00070 };
00071
00076 class FileUtilise
00077 {
00078 public:
00079
00083     FileUtilise();
00087     ~FileUtilise();
00088
00094     void loadFromFile(const string& filename, vector<Item*>& items);
00100     void saveToFile(const string& filename, const vector<Item*>& items);
00101 };
00102
```

Skorowidz