04.05.2021r.

Projekt aplikacja sklepowa

Programowanie komputerów 4

Jakub Domerecki inf. Gr.5

1. Analiza tematu

Moja aplikacja jest programem ułatwiającym pracę osobie zarządzającej sklepem, pracownikowi a także klientowi sklepu. Każda z poszczególnych osób może logować się do systemu i wykonywać odmienne czynności. Całość opcji będzie opisana w specyfikacji wewnętrznej. Jest to aplikacja konsolowa napisana obiektowo w C++. Klasy w programie zostały dobrane tak, aby każda odpowiadała za daną osobę, przykładowo Client klientowi itd. Dodatkowo stworzyłem kilka innych klas które były pomocne, przykładowo klasa application obsługująca cały program, db agregująca dane. Najbardziej użyteczne do przechowywania danych okazały się kontenery STL, a różne przeszukiwanie wypisywanie lub wstawianie do nich spełniły dobrze w moim programie operatory związane z nimi. Wykorzystałem również wzorzec projektowy singleton, który okazał się wspaniałym narzędziem do dostępu do danych z klasy db i używanym w całym programie co ułatwiało znacznie napisanie projektu. W skład aplikacji wchodzą następujące pliki:

- Źródłowe

  `main.cpp client.cpp application.cpp db.cpp Worker.cpp Manager.cpp`
- Nagłówkowe

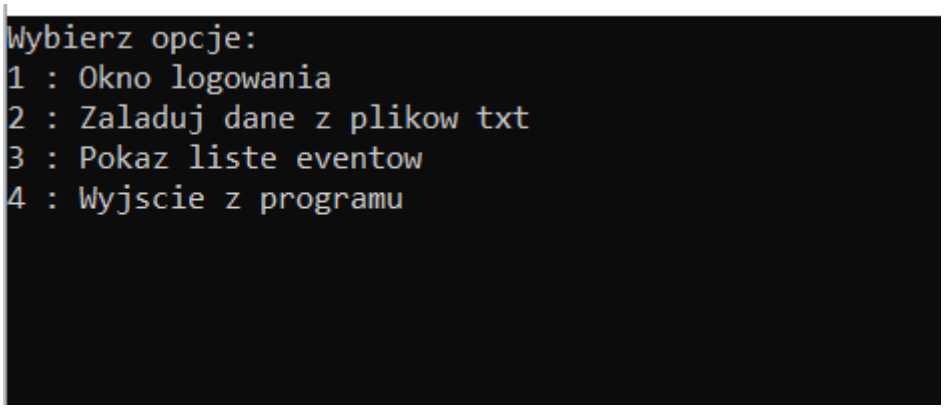
  `Person.h Client.h application.h Product.h Worker.h Manager.h db.h`

- Tekstowe (dane dla aplikacji)

  `client.txt worker.txt product.txt`

2. Specyfikacja zewnętrzna

Użytkownikowi po skompilowaniu programu wyświetlą się konsola z danymi niżej opcjami:



Oczywiście jest tu możliwość dowolnego wyboru poprzez wpisanie cyfry 1,2,3 lub 4, wpisanie czegokolwiek innego wyświetli błąd

Po wybraniu 1 wyskoczą nam opcje do logowania trzeba podać login i hasło istniejącego konta, oczywiście po wcześniejszym wczytaniu danych z plikow txt(2)

Tak właśnie działa całość programu. Polega to na wybieraniu opcji jakich chcemy i oczywiście jakie mamy prawo wykonać zależnie od tego czy jesteśmy zalogowani jako klient, pracownik lub menadżer.

3. Specyfikacja wewnętrzna

1. Application

Klasa ta jest główną klasą aplikacji. W punkcie wejściowym w funkcji main wołamy metodę run(), która jest odłożona na stosie przez cały czas użytkowania z aplikacji. W tej metodzie mamy nieskończoną pętle, którą możemy przerwać wybierając opcję z zamknięcia z menu. Poniżej interfejs użytkownika jaki oferuje metoda run().

1 : Okno logowania
2 : Zaladuj dane z plikow txt
3 : Pokaz liste eventow
4 : Wyjscie z programu

Wybranie opcji 1 pozwoli nam zalogować się na konto istniejące w aplikacji (Klient/Pracownik/Manager). Wybranie opcji 2 pozwoli nam załadować dale z plików txt do aplikacji. Wybranie opcji 3 pozwoli nam wyświetlenie wszystkich eventów jakie się wydarzyły w aplikacji, takich jak dokonanie zamówienia, złożenie skargi, rozpatrzenie skargi, utworzenie konta klienta, zatrudnienie nowego pracownika. Istnieje możliwość filtrowania eventów po ich rodzaju.

2. DB

Klasa ta jest klasą agregującą dane aplikacji. W poniższych obiektach:

```
std::list<Product> shop;
std::map<std::string, std::unique_ptr<Client>> clients;
std::map<std::string, std::unique_ptr<Worker>> workers;
Manager manager;
std::multimap<std::string, std::string> complaints;
std::multimap<EventType, std::string> event;
int income;
```

Klasa ta została zaprojektowana jako singleton z leniwą inicjalizacją instancji klasy. Klasa zawiera wewnętrzny enum EventType. Klasa posiada metodę pointującą wszystkie dostępne aktualnie produkty.

3. Person

Jest to klasa abstrakcyjna będącą bazą, dla innych klas takich jak Worker, Client. Klasa posiada czysto wirtualną metodę run(), zaimplementowaną metode do zmiany hasla. W sklad klasy wchodzą dwa pola std::string login i std::string password.

4. Client

Klasa dziedziczy publicznie po klasie Person. Główną metodą jest metoda run().Poniżej interfejs użytkownika jaki oferuje metoda run().

Konto Klienta: <login clienta>
1 : Wyloguj
2 : Zloz zamowienie
3 : Zloz skarge
4 : zmien haslo

Opcja 1 wyloguje nas z konta klienta, wyjdzie z metody run. Opcja 2 wyświetli listę oferowanych produktów i pozwoli dokonać zakupu. Opcja 3 zawoła metodę pozwalającą

złożyć skargę, skarga zostanie umieszczona w DB:: std::multimap<std::string, std::string> complaints i będzie ona mogla być rozpatrzona przez pracownika (instancje klasy Worker). W skład klasy wchodzi pole int numOfOrders reprezentującą ilość złożonych zamówień przez zalogowanego klienta.

5. Worker

Klasa dziedziczy publicznie po klasie Person. Główną metodą jest metoda run().Poniżej interfejs użytkownika jaki oferuje metoda run().

Konto Pracownika: <login Workera>
1 : Wyloguj
2 : Pokaz wszystkich klientow
3 : Stworz konto nowego klienta
4 : Przeczytaj nastepna skarge
5 : zmien haslo
6 : Dodaj towar do sklepu
7 : Pokaz zawartosc sklepu

Opcja 1 wyloguje nas z konta workera, wyjdzie z metody run. Opcja 2 wyświetli listę istniejących kont klientów. Opcja 3 pozwoli utworzyć konto w aplikacji dla nowego klienta. Opcja 4 służy do zachowania metody służącej obsłudze skarg złożonych przez klientów. Opcja 6 pozwala dodać produkt do listy dostępnych produktów w sklepie. Opcja 7 pozwala nam zobaczyć jakie produkty aktualnie są w sklepie. W skład klasy wchodzą pola int salary – reprezentuje pensje pracownika, oraz pole std::string position reprezentujące stanowisko pracownika.

6. Manager

Klasa dziedziczy publicznie po klasie Worker. W aplikacji zostaje stworzony jeden predefiniowane konto managera login: root hasło: root. Główną metodą jest metoda run().Poniżej interfejs użytkownika jaki oferuje metoda run().

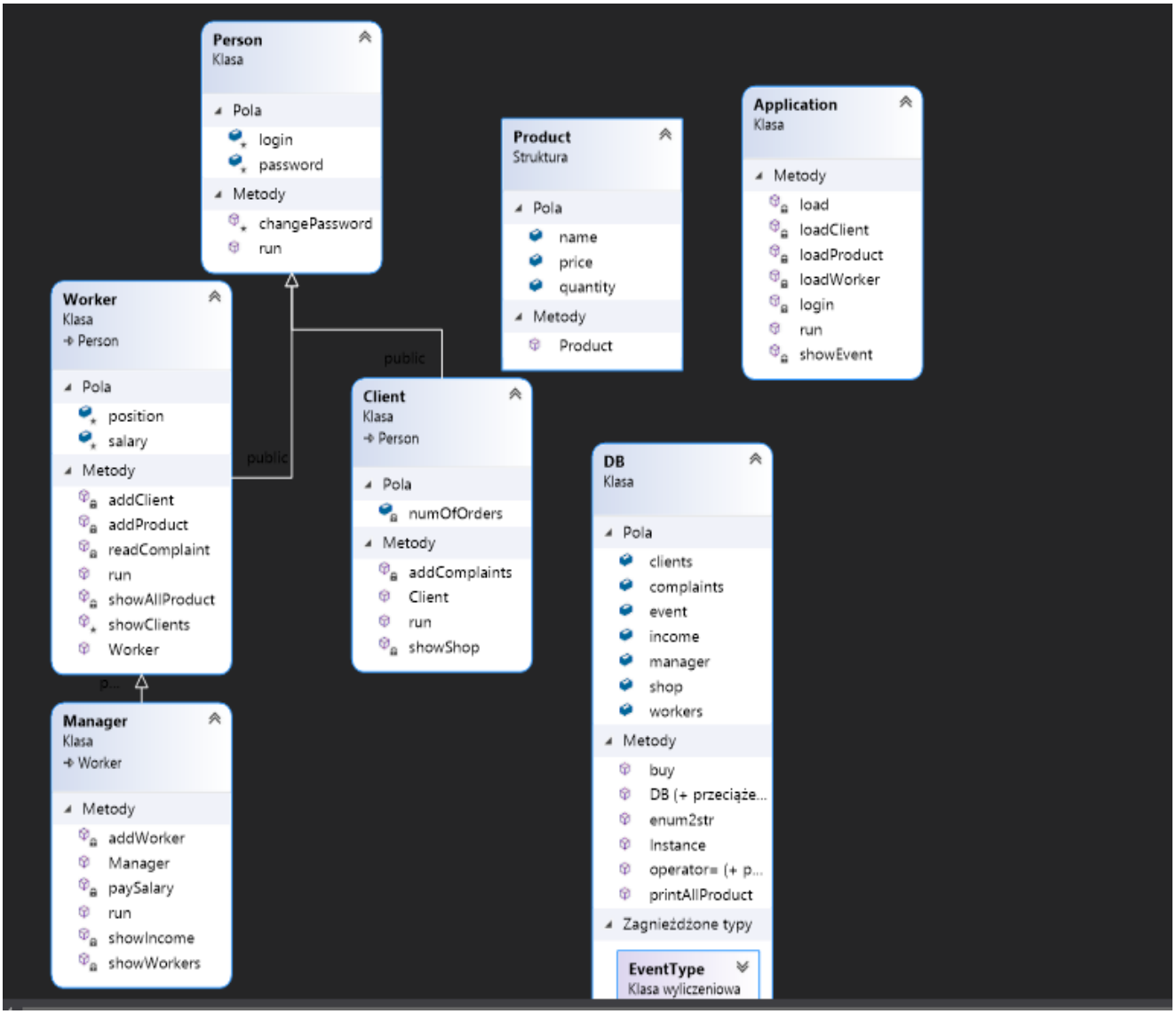
Konto Managera: <login Managera>
1 : Wyloguj
2 : Pokaz wszystkich klientow
3 : Pokaz wszystkich pracownikow
4 : Wyplac pensje
5 : zmien haslo
6 : Zatrudni pracownika
7 : Pokaz przychod

Opcja 1 wyloguje nas z konta Managera, wyjdzie z metody run. Opcja 2 wyświetli listę istniejących kont klientów. Opcja 3 pozwoli wyświetli listę istniejących kont workerów. Opcja 4 pobierze z przychodu wartość sumy pensji wszystkich pracowników. Opcja 6 pozwala dodać konto dla nowo zatrudnionego pracownika. Opcja 7 pozwala wyświetlić aktualny przychód, pole DB::income(początkowo ustawione na 100000).

7. Product

Klasa stanowiąca reprezentację produktów znajdujących się w sklepie. Klasa posiada trzy pola std::string name – nazwa produktu, int quantity – ilość sztuk znajdujących się w sklepie, float price – cena za jedną sztukę produktu.

Diagram klas



4. Testowanie i uruchamianie

W trakcie pisania programu natknąłem się na wycieki pamięci. Początkowo nie wiedziałem z czym one były związane, lecz ostatecznie udało się wykryć źródło. Singleton wykorzystujący dane z db udostepniając je w różnych miejscach w programie powodował takie wycieki. Udało to się skorygować po dodaniu poniższego fragmentu kodu :

```
DB(DB const&) = delete;
DB& operator=(DB const&) = delete;
DB(DB&&) = delete;
DB& operator=(DB&&) = delete;
```

Poza tym problemem natknąłem się również na inne ale to były już takie mniej istotne, które szybko udało się wykryć i zreperować. Przykładowo źle napisana pętla lub zła implementacja operatora STL itp.

5. Wnioski

Wybrany przeze mnie temat okazał się bardzo ciekawy. Myślę, że takie pisanie aplikacji związanych z logowaniem jest aktualnie bardzo istotne i każdy większy lub bardziej znany sklep, firma, przedsiębiorstwo musi mieć własną stronę z możliwością logowania i wykonywania różnych czynności związanych z danym kontem. Obiektowe pisanie w przypadku takiego programu jest na pewno zdecydowanie lepsze oraz szybsze i łatwiejsze od strukturalnego. Pozwala ono na łatwa rozbudowę na przykład konta klienta i jego opcji lub jakiejkolwiek innej części programu. Jest duże prawdopodobieństwo że w przyszłości będę pisał coś podobnego lub coś zawartego w tym projekcie i będę mógł to jakoś wykorzystać.