Igor Kraszewski - Album: 310164

Adam Dąbkowski - Album: 310035

Daniel Szepietowski - Album: 310316

Symulacja księgarni w języku C++

Spis treści

Dokumentacja

1.1 Przyjęte założenia

W symulacji stosujemy założenia pozwalające naśladować sposób działania rzeczywistej księgarni. W księgarni pracuje pewna ilość sprzedawców, znajdują się tam klienci oraz ma w swoich zbiorach książki o różnej tematyce i cenach. Książki są sortowane poprzez umieszczenie ich w odpowiednim dziale tematycznym.

Przyjęte przez nas założenia:

- W księgarnia pracuje określona ilość sprzedawców
- W księgarni znajduje się określona ilość klientów
- Jedna iteracja programu jest równa czasowi, w którym sprzedawca obsługuje jednego klienta
- Każdy klient przeprowadza interakcje ze sprzedawcą
- Klient może wykonywać 5 akcji:
    1. Zapytać o książkę
    2. Kupić książkę
    3. Zamówić książkę
    4. Zdecydować się na kolejną książkę
    5. Zrezygnować
- Książki dzielone są na podstawie tematyki na pewną liczbę działów
- Sprzedawca wystawia klientowi paragon, gdy kupił lub zamówił on książkę

1.2 Działanie programu

Ilość sprzedawców, klientów oraz czas trwania symulacji jest wczytywany z pliku. Sprzedawcy oraz klienci są losowani w odpowiedniej ilości z obiektów stworzonych za pomocą wczytywania danych z odpowiednich plików csv. Za pomocą funkcji pochodzących z bibliotek <*random*> i <*chrono*> losowane jest wiele wartości. Są to klienci, sprzedawcy, książka o którą zapyta klient, czy akcja, którą wykona. Dodatkowo losowane są pewne atrybuty obiektów takie jak: ilość pieniędzy posiadanych przez klienta, czy ilość egzemplarzy oraz cena danej książki. Wynik symulacji jest wypisywany zarówno w terminalu jak i zapisywany do pliku tekstowego.

Pliki baz danych, które obsługujemy to:

- Books.csv – baza danych książek zawierająca ich tytuły, autorów, id oraz rodzaj
- Salesmen.csv – baza danych sprzedawców księgarni zawierająca ich imiona, nazwiska, id, uprawnienia, lata doświadczenia oraz działy, którymi się opiekują
- Customers.csv – baza danych klientów księgarni opisanych za pomocą imienia, nazwiska, id oraz preferencji dotyczących tematyki książek
- Sections.csv – baza danych sekcji księgarni zawierająca ich nazwy oraz symbole
- Simulation.csv – plik zawierający dane potrzebne do przeprowadzenia symulacji
- Simulation_result.txt – plik zawierający wynik przeprowadzonej symulacji

```
1   title,author,id,genre
2   Outopost,Dmitry Glukhovsky,1,fantasy
3   Pozniej,Stephen King,2,fantasy
4   Wiedzmin,Andrzej Sapkowski,3,fantasy
```

Rysunek 1 - fragment bazy danych książek

```
name,surname,worker_id,permissions,experience,departments
Mateusz,Muranski,1,kierownik,8,fantasy,sport
Marek,Markiewicz,2,asystent,3,sport,biznes
```

Rysunek 2 - fragment bazy danych sprzedawców

```
name,surname,cardId,preferences
Igor,Rogi,1,biznes,fantasy
Adam,Dab,2,informatyka
Daniel,Szepiks,3,informatyka
```

Rysunek 3 - fragment bazy danych klientów

```
1   name,symbol
2   fantasy,A
3   biznes,B
4   historia,C
```

Rysunek 4 - fragment bazy danych działów tematycznych

Jednym z założeń jest to dotyczące jaki czas opisuje jedna iteracja programu. W naszym przypadku jest to czas w jakim każdy sprzedawca obsługuje jednego klienta. Klienci podczas interakcji ze sprzedawcą pytają go o wylosowaną książkę, a następnie mogą wykonać jedną

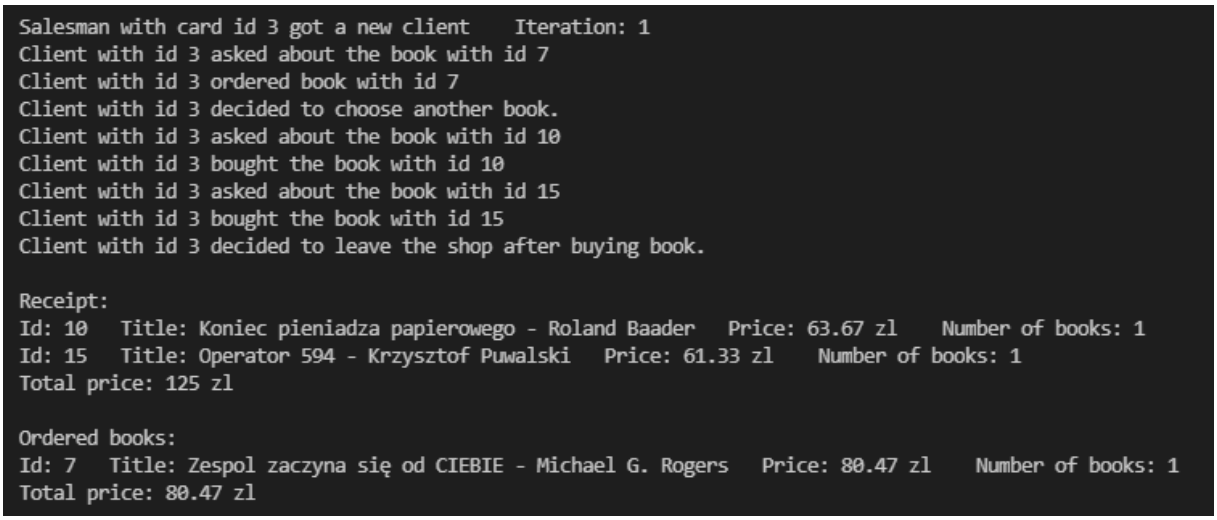
z poniższych akcji (wybrana akcja jest losowa):

- Kupić książkę (jeżeli jej ilość jest większa od 0)
- Zamówić książkę (jeżeli ilość egzemplarzy książki jest równa 0)
- Zrezygnować z zakupów

W przypadku chęci zakupu książki sprawdzamy, czy klient posiada wystarczającą liczbę pieniędzy aby ją kupić. Kiedy ma pieniądze, kupuje książkę po czym może wyrazić chęć kupienia kolejnej lub zrezygnować, a gdy brakuje mu pieniędzy może zrezygnować z zakupów lub zapytać o inną książkę.

Podobnie wyglądają możliwe interakcje dla sytuacji kiedy klient chce zamówić książkę pomijając przypadek niewystarczającej ilości pieniędzy, ponieważ zakładamy, że książka opłacana jest przy odbiorze.

Symulacja kończy się po upłynięciu wcześniej założonego czasu trwania lub w sytuacji, gdy wszyscy klienci opuszczą sklep przed końcem czasu.



Rysunek 5- Fragment przebiegu symulacj

1.3 Podział na pliki

- *book.h, book.cpp*

  Pliki te zawierają implementację klasy Book. Klasa ta zawiera takie atrybuty jak: *title(tytuł)*, *author(autor), genre(rodzaj), id, price(cena), number(ilość egzemplarzy).*

- *person.h*

  Plik ten zawiera implementację abstrakcyjnej klasy Person, która stanowi bazę dla klas Customer i Salesman. Posiada metodę wirtualną to_string() – jej implementacja różni się w każdej klasie podrzędnej. Posiada atrybuty: *name(imie), surname(nazwisko), cardId(numer Id).*

- *customer.h, customer.cpp*

  Pliki zawierające implementację klasy Customer, która dziedziczy po klasie abstrakcyjnej Person. Klasa ta posiada oprócz zmiennych odziedziczonych zmienne: *money(ilość pieniędzy), basket(koszyk na zakupione książki), ordered_book(koszyk na zamówione książki), preferences(preferencje klienta zapisane w wektorze z biblioteki STL).*

- *salesman.h, salesman.cpp*

  Pliki zawierające implementacje klasy Salesman. Atrybuty poza tymi z klasy nadrzędnej to: *departments(działy którymi się opiekuje zapisane za pomocą wektora z biblioteki STL), permissions(uprawnienia), experience(lata doświadczenia), customer(obiekt klasy Customer – klient obsługiwany przez sprzedawcę)*

Powyższe pliki opisują podstawowe klasy wykorzystywane w naszym projekcie.

Pozostałe pliki to:

- box.h, box.cpp
- basket.h, basket.cpp – Klasa Basket (koszyk na książki)
- books_col.h – Klasa BooksCol (przechowywanie książek)
- bookstore.h, bookstore.cpp – Klasa Bookstore (księgarnia)
- ordered_books.h, ordered_books.cpp – Klasa OrderedBooks (koszyk na zamówione książki)
- person_col.h – Klasa PersonCol – (szablon do przechowywania osób)

- section_col.h, section_col.cpp – Klasa SectionCol (przechowywanie działów tematycznych)
- section.h, section.cpp – Klasa Section (dział tematyczny)
- utils.h, utils.cpp – odczytywanie i zapisywanie do plików
- simulation.h, simulation.cpp – Klasa Simulation (operacje dotyczące przeprowadzenia symulacji)
- main.cpp

Klasa Box jest klasą abstrakcyjną, po której dziedziczą klasy: Basket, OrderedBooks, BooksCol. Klasa opiera się na mapie pochodzącej z biblioteki STL <map> w postaci zmiennej books. Inna klasą wyróżniającą się z pozostałych jest klasa PersonCol, ponieważ jest szablonem wykorzystywanym do przechowywania osób (Customer, Salesman).

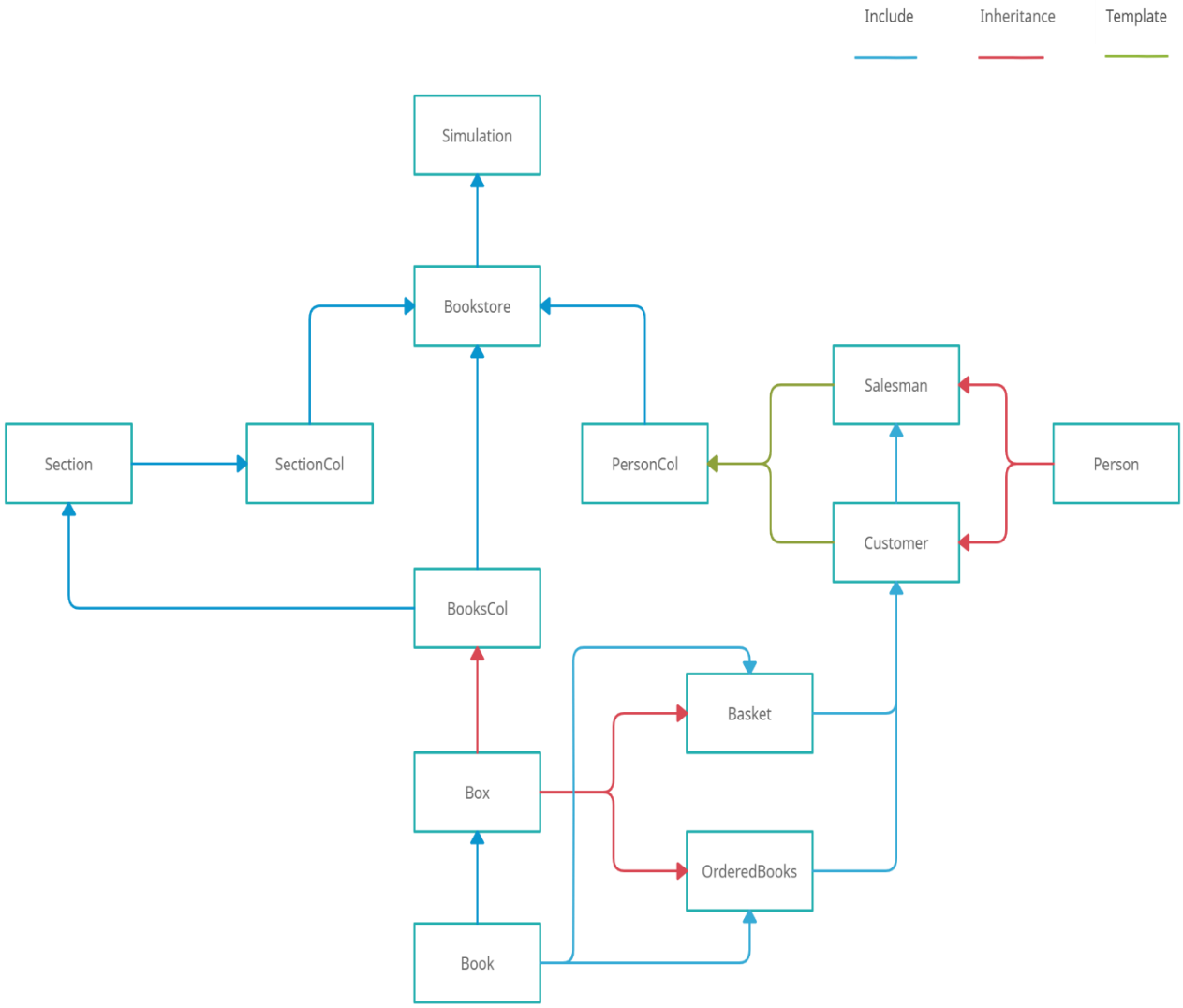
Klasa Bookstore jest połączeniem wcześniej zaimplementowanych części kodu w całość w strukturę księgarni, którą potem wykorzystujemy jako źródło danych do symulacji przeprowadzanej przez metody klasy Simulation.

1.4 Opis sytuacji wyjątkowych

Najbardziej narażonym na błędy jest proces wczytywania danych z plików. To właśnie tam za pomocą przechwytywania i wyrzucania wyjątków jesteśmy w stanie sprawdzić czy pobierane dane są poprawne. W tym celu korzystamy z biblioteki <stdexcept>. Jako błędy traktujemy nieodpowiedni typ danych oraz ich nieprawidłowa ilość.

Aby umożliwić płynność działania programu w przypadku podania argumentów startowych programu wykraczających poza zakresy naszych baz danych są one konwertowane do maksymalnych możliwych do zrealizowania wartości. Wystąpienie takiej sytuacji jest komunikowane za pomocą odpowiedniego tekstu.

1.5 Hierarchia i powiązania klas



Praca została wykonana wspólnie za pomocą narzędzia LiveShare programu Visual Studio Code. Cały napisany kod konsultowaliśmy i wykonywaliśmy wspólnie. Dlatego wkład w projekt jest rozłożony równomiernie na wszystkich członków grupy 1.7.