Temat projektu: *Księgarnia*

1. Założenia projektowe i opis klas:

W projekcie przewiduje się powstanie 3 głównych klas o nazwach: `seller`, `client`, `book`. Dodatkowo przewiduje się napisanie klasy o nazwie `symulation`, która to będzie odpowiedzialna za tworzenie obiektów głównych klas oraz poprawne przeprowadzenie całej symulacji. Podział na klasy i relacje między nimi wygląda następująco:

	Atrybuty	Metody
seller	Identyfikator	Odpowiadanie na pytania klienta
	Imię	Wystawianie rachunku
	Nazwisko	Gettery, settery
	Dostępność	
	Id klienta obsługiwanego	

Sprzedawca obsługuje klienta. Sprzedawca zaprzyjaźniony z książką – będzie miał dostęp do jej atrybutów prywatnych. Na początku symulacji każdy z sprzedawców jest wolny.

	Atrybuty	**Metody**
client	Identyfikator	Pytanie (1)
	Imię	Zamówienie (2)
	Nazwisko	Zakup (3)
	Określony cel wizyty	Gettery i settery
	ISBN książki, którą klient jest zainteresowany	Zmiany stanu w zależności od aktualnego stanu klienta
	Id przydzielonego sprzedawcy	
	Stan: oczekujący/osługiwany	

Klient ma określony cel wizyty. Może on zadawać ogólne pytania o dane książki, dokonać zakupu lub złożyć zamówienie. Każda z metod zajmuje określony czas podczas symulacji – wartości podane w nawiasach. Klient zawsze zajmuje czas 1 sprzedawcy. Klienci obsługiwani są w kolejności swoich numerów id, które to symbolizują czas przyjścia – mogą tworzyć się kolejki oczekujących.

	Atrybuty	Metody
book	Autor	Gettery settery
	Tytuł	
	Uproszczony identyfikator ISBN	
	Gatunek	
	Cena	
	Dostępność	

	Atrybuty	Metody
magazine	Autor	
	Tytuł	
	Uproszczony identyfikator ISBN	
	Gatunek	
	Cena	
	Dostępność	
	Data wydania	

	Atrybuty	Metody
ebook	Autor	
	Tytuł	
	Uproszczony identyfikator ISBN	
	Gatunek	
	Cena	
	Dostępność	
	Format	

Książkę oprócz oczywistych atrybutów, cechuje także dostępność. Dostępność oznacza tak naprawdę stan danej książki. Po symulacji będą wyświetlane statystyki pokazujące, którymi książkami klienci byli zainteresowani

Czasopisma i ebooki wykorzystują dziedziczenie i polimorfizm z klasy podstawowej książka.

	Atrybuty	**Metody**
simulation		do_simulation()

Symulacja przeprowadzana jest w pętli, jeden obrót pętli odpowiada jednej jednostce czasu, długość działania księgarni ustawia się poprzez podanie argumentu przy wywołaniu programu.

W księgarni nie przewiduje się limitu książek. Zakłada się, że w ciągu 1 symulacji książek wystarczy dla każdego klienta w danym dniu, o ile tylko zapyta o książkę z wczytywanej z pliku do programu listy książek.

Dodatkowo w programie zostaną zaimplementowane klasy będące kontenerami wskaźników dla wszystkich obiektów książek, klientów oraz sprzedawców. W zależności od typu kontenera pojawią się tam takie metody jak: dodawanie nowych obiektów, usuwanie istniejących już obiektów, wyszukiwanie obiektów, listowanie obiektów w okienku terminalowym, zmiany atrybutów obiektów, współpraca z pozostałymi klasami.

Oprócz powyżej omówionych klas pojawią się jeszcze klasy odpowiedzialne za poprawne wczytywanie danych z plików tekstowych – klasa `file_operation`. Dodatkowo zostaną zaimplementowane klasy obsługujące wyjątki.

2. Opis działania symulacji

Aby uruchomić symulację należy w terminalu w folderze z aplikacją napisać polecenie:

```
proi_23l_201_ksiegarnia x y z t
```

gdzie:
- x to parametr określający żądany czas symulacji,
- y to liczba sprzedawców w księgarni,
- z to liczba klientów, która się zgłosi podczas symulacji,
- t to nazwa pliku .txt z danymi książkami, o które klienci mogą pytać, zamawiać lub kupować.

Dane należy wpisywać po spacjach, a na końcu wcisnąć enter.

Do przebiegu symulacji wykorzystano takie mechanizmy jak dziedziczenie czy polimorfizm. Dziedziczenie użyto przy wprowadzeniu podziału na książki – ogólne(klasa bazowa `book`) oraz ebooki czy magazyny, które stanowiły klasy pochodne. Dla każdej z klas bazowych ustalono inny sposób liczenia ceny za daną pozycję, a także inny sposób wyświetlania danego rodzaju pozycji w terminalu. Metody wyliczania ceny książek i wyświetlania informacji o książkach w terminalu są metodami

wirtualnymi, korzystającymi z polimorfizmu. Polimorfizm ten wykorzystano do operacji na wskaźnikach do klas bazowych – klasa `bookcollection`.

Wynik symulacji jest wyświetlany w czasie rzeczywistym w okienku terminalowym oraz zapisywany do pliku `simulation_results.txt` oraz `simulation_start.txt`. Do pierwszego z nich zapisywane są wyniki symulacji, a do drugiego z nich parametry startowe symulacji, podane przez użytkownika podczas wywoływania programu.

Przed rozpoczęciem symulacji każdemu klientowi przydziela się losowo książkę, którą jest zainteresowany. Dla każdego klienta dodatkowo wyznacza się akcję, tzn. czy dany klient chce kupić czy zamówić czy jedynie spytać się o cenę danej książki. Każda z tych czynności podczas trwania symulacji będzie trwała inną ilość czasu jednostkowego, co w konsekwencji ma wpływ na całkowity czas obsługiwania klientów i długość kolejki.

Po wykonaniu symulacji program zwraca informacje, o aktualnym stanie książek, aby pod koniec dnia księgarnia mogła uzupełnić braki magazynowe. Dodatkowo wyświetlany jest dzienny przychód księgarni na podstawie zakupów, których dokonali klienci (założono, że złożenie zamówienia nie daje jeszcze przychodu księgarni).

3. Wykorzystane elementy biblioteki STL

W projekcie wykorzystano następujące elementy z biblioteki STL:

std::list – kontener ten wykorzystano przy budowaniu kolekcji książek – klasa `bookcollection`, zbioru sprzedawców – klasa `seler_list` oraz zbioru klientów – klasa `client_list`. R

std::begin() / std::end() – iterator rozpoczynający oraz kończący sekwencję. Wykorzystany w klasach stanowiących kolekcje obiektów, w pętlach podczas odwoływania się do poszczególnych metod danego obiektu.

4. Sytuacje wyjątkowe i ich obsługa

Zbiór sytuacji wyjątkowych:

Problem:	Rozwiązanie:
Klienci zostaną obsłużeni przed końcem czasu trwania symulacji	Przerwanie symulacji, wyświetlenie w terminalu komunikatu, że wszyscy klienci zostali obsłużeni.
Symulacja skończy się przed obsługą wszystkich klientów	Część klientów nie zostanie obsłużona, brak błędu.
Nieudany odczyt z pliku	Pojawienie się komunikatu: `can't open given file`. Przerwanie działania programu.
Wprowadzenie na listę klientów wskaźnika do istniejącego już klienta	Wyrzucenie wyjątku: `AlreadyExistingClientException`
Ponowne wprowadzenie książki o podanym wcześniej numerze ISBN do listy książek	Komunikat o próbie dodania już istniejącej książki, symulacja przebiega dalej bez zmian bez dodania tej książki do listy
Wprowadzenie na listę sprzedawców istniejącego już sprzedawcę	Wyrzucenie wyjątku: `AlreadyExistingSellerException`
Wczytanie z pliku ceny książki o ujemnej wartości	Wyrzucenie wyjątku: `ExceptionNegativePrice`

Prośba klienta o nieistniejącą książkę	Sytuacja taka nie wystąpi, ponieważ klienci mogą pytać jedynie o książki z wczytywanej listy książek

5. Opis przeprowadzonych testów

W celu weryfikacji poprawności działania programu przeprowadzono szereg testów jednostkowych. Sprawdzono w nich poprawne działanie metod zaimplementowanych klas. Zbadano przede wszystkim poprawne ustawianie wartości parametrów oraz zachowanie klas w przypadkach niestandardowych takich jak np. odwołanie do nieistniejących elementów. Wyniki testów przedstawiono poniżej: