Dokumentacja symulatora apteki

Marcin Białek

Założenia

- Parametry symulacji wczytywane są z pliku *config.json* i są nimi:
    - Liczba cykli na sekundę,
    - Czas trwania symulacji (w liczbach cykli),
    - Nazwa pliku, do którego będą zapisywane komunikaty symulacji,
    - Prawdopodobieństwo pojawienia się nowego klienta w danym cyklu symulacji,
    - Maksymalna liczba leków, które jednorazowo może kupić klient,
    - Minimalna liczba cykli pomiędzy dostawami leków do apteki,
    - Liczba pracowników apteki,
    - Liczba okienek w aptece, przy których mogą być obsługiwani klienci,
    - Liczba każdego leku w aptece na początku symulacji,
    - Minimalna liczba leków jaka może zostać zamówiona,
    - Ilość pieniędzy w aptece na początku symulacji,
    - Podatek i marża od sprzedaży leków,
    - Minimalna liczba osób w kolejce potrzebna, aby aptekarz otworzył nowe okienko,
    - Grupy leków,
    - Leki (nazwa, grupa, cena, czy wymaga recepty).

- Klienci wchodzą do apteki w losowych momentach symulacji i trafiają do wspólnej kolejki oczekując na obsłużenie.

- Są dwa typy klientów: indywidualny i biznesowy. Klient biznesowy posiada numer NIP. Klient może chcieć kupić leki bez recepty, leki na receptę lub zasięgnąć rady od aptekarza.

- W momencie kiedy w kolejce czeka klient i żadne z okienek nie jest otwarte, wolny pracownik otwiera nowe stanowisko.

- Kiedy co najmniej jedno z okienek jest otwarte, liczba osób potrzebna, aby wolny aptekarz otworzył kolejne, musi przekroczyć ustaloną liczbę.

- Przy jednym okienku może być obsługiwany tylko jeden klient na raz. Czas obsługi klienta jest losowy.

- Jeśli wszyscy klienci zostali obsłużeni, aptekarze zamykają okienka.

- Po pomyślnym obsłużeniu klienta, który kupował leki, otrzyma on paragon (klient indywidualny) lub fakturę (klient biznesowy).

- Paragon zawiera: nazwy zakupionych leków, ich liczbę, cenę brutto oraz sumę.

- Faktura zawiera: NIP, nazwy zakupionych leków, ich liczbę, stawkę podatku, cenę netto, cenę brutto, kategorie zakupionych leków, sumę netto i sumę brutto.

- Leki, które się skończyły zostają dopisane do listy leków do zamówienia.

- Dostawa leków przyjeżdża okresowo i tylko po określonym czasie. Zawiera ona leki wpisane do listy leków do zamówienia. Za dostawę wystawiana jest faktura, która pokrywana jest z budżetu apteki.

- Dostawa musi zostać rozpakowana. Robią to pracownicy, którzy nie obsługują klientów. Czas potrzebny na rozpakowanie leku przez jednego pracownika jest losowy.

Hierarchia klas

- **Json** - klasa reprezentująca format JSON, w którym jest zapisany plik konfiguracyjny.

- **Json::Value** - reprezentacja wartości w obiekcie Json
    - **Json::String**
    - **Json::Integer**
    - **Json::Double**
    - **Json::Object**
    - **Json::Array**
    - **Json::Bool**

- **JsonParser** - klasa parsująca plik tekstowy *.json*

- **Logger** - klasa ze statycznymi metodami do wypisywania komunikatów oraz do zapisywania ich do pliku.

- **Random** - klasa ze statycznymi metodami do losowania liczb i zmiennych typu prawda/fałsz.

- **Exception** - klasa wyjątków.
    - **Json::JsonError**
    - **Json::InvalidCasting**
    - **JsonParser::ParserError**
        - **JsonParser::Invalid**
        - **JsonParser::Expected**
        - **JsonParser::Unexpected**

- **Identifiable** - klasa nadająca obiektom unikalny identyfikator.
    - **Bill** - klasa reprezentująca rachunek za produkty.
        - **Receipt** - klasa reprezentująca paragon.
        - **Invoice** - klasa reprezentująca fakturę.
    - **Person** - klasa reprezentująca osobę.
        - **Employee** - klasa reprezentująca pracownika apteki.
        - **Customer** - klasa reprezentująca klienta apteki.
            - **BusinessCustomer** - klasa reprezentująca klienta biznesowego.
    - **Prescription** - klasa reprezentująca receptę na leki.
    - **Window** - klasa reprezentująca okienko w aptece do obsługi klientów.

- **Product** - klasa reprezentująca produkt na rachunku.

- **MedicineDatabase** - klasa reprezentująca bazę z istniejącymi lekami.

- **Medicine** - klasa reprezentująca wpis leku w bazie leków.

- **Pharmacy** - klasa reprezentująca aptekę.

- **EventEmmiter** - generator klientów.

Testowanie działania programu

Program został przetestowany poprzez podanie różnych plików konfiguracyjnych oraz analizę wyświetlanych komunikatów pod względem ich poprawności oraz logiczności. Nie stwierdzono nieprawidłowości tzn. nie wystąpiły sytuacje kiedy pracownik robił dwie rzeczy jednocześnie, klient chciał kupić nieistniejący lek, klient nie został obsłużony, itp.

Wykorzystane elementów biblioteki STL

Klasa	Użycie w programie
std::vector	Lista leków do kupienia przez klienta, Lista leków na recepcie, Lista pracowników apteki, Lista okienek w aptece
std::map	Produkty na rachunku, Grupy leków w bazie leków, Leki w bazie leków, Dostępne lekarstwa w aptece, Lekarstwa wymagające odpakowania
std::queue	Kolejka klientów w aptece, Kolejka komunikatów do zapisania do pliku
std::set	Lista leków do zamówienia

Sytuacje wyjątkowe

Wyjątki pojawiają się w momencie podania błędnego pliku konfiguracyjnego lub zawierającego ustawienia, które powodują nieprawidłowe działanie programu, na przykład kiedy liczba pracowników apteki zostanie ustawiona na 0. Dzieje się tak równie wtedy, gdy nie udało się otworzyć pliku konfiguracyjnego lub pliku do zapisywania komunikatów.

Każdy wyjątek zostaje zasygnalizowany i kończy działanie programu.