Symulacja banku

Jakub Robaczewski, Oskar Bartosz

Opis przyjętych założeń:

Przy tworzeniu naszej symulacji założyliśmy, że w banku jest zarejestrowanych m klientów, oraz n pracowników pracujących przy różnych okienkach, które realizują różne potrzeby klientów: okienka dotyczące konta, okienka dotyczące płatności i okienka informacyjne. W banku znajdują się również bankomaty i wpłatomaty, które są samoobsługowe. Klientów banku można podzielić na indywidualnych i biznesowych, rozróżnia ich początkowa ilość pieniędzy. Do każdego okienka ustawiona jest kolejka, przy czym zakładamy, że klienci raz stając w kolejce nie zmieniają jej do czasu obsłużenia (nie mogą „przeskakiwać" do mniejszej kolejki). Klienci wybierają okienka w oparciu o swoje potrzeby oraz aktualną długość kolejki przy nich. Jeśli nie rozstrzyga to wyboru ustawiają się do okienka o mniejszym indeksie. Operacje przy okienkach różnią się czasem w zależności od dokonywanej operacji. Symulacja rozpoczyna się i kończy o podanych godzinach, ale jej krok to zawsze 1 minuta.

Algorytm symulacji:

1. Początek symulacji. Aktualny czas = czas początkowy
2. Sprawdź, czy aktualny czas nie jest czasem końcowym. Jeśli tak to idź do 9.
3. Losuj, czy nowy klient przyjdzie do banku.
4. Jeżeli przyszedł nowy klient to ustaw do odpowiedniej kolejki.
5. Wykonaj operacje przy okienkach, jeśli upłynął czas ich wykonania oraz usuń takich klientów z banku.
6. Przesuń kolejki w całym banku.
7. Zwiększ aktualny czas o 1 minutę.
8. Wróć do 2.
9. Zakończ symulację.

Elementy biblioteki STL:

W naszym programie wykorzystujemy następujące elementy biblioteki STL:

- Wektory – do przechowywania okienek w banku, zarejestrowanych klientów, pracowników, imion i nazwisk do losowania, dostępnych operacji w okienkach
- Kolejki – do implementacji kolejki przed okienkiem

Testowanie programu:

Symulację banku można uruchomić na 3 sposoby:

Uruchomienie pliku bez podania żadnych parametrów:

Podczas uruchomiania programu bez podanych żadnych parametrów pobiera on ustawienia symulacji z pliku settings.txt. Jeśli program nie znajdzie odpowiedniego pliku, wyświetli odpowiedni komunikat o błędzie.

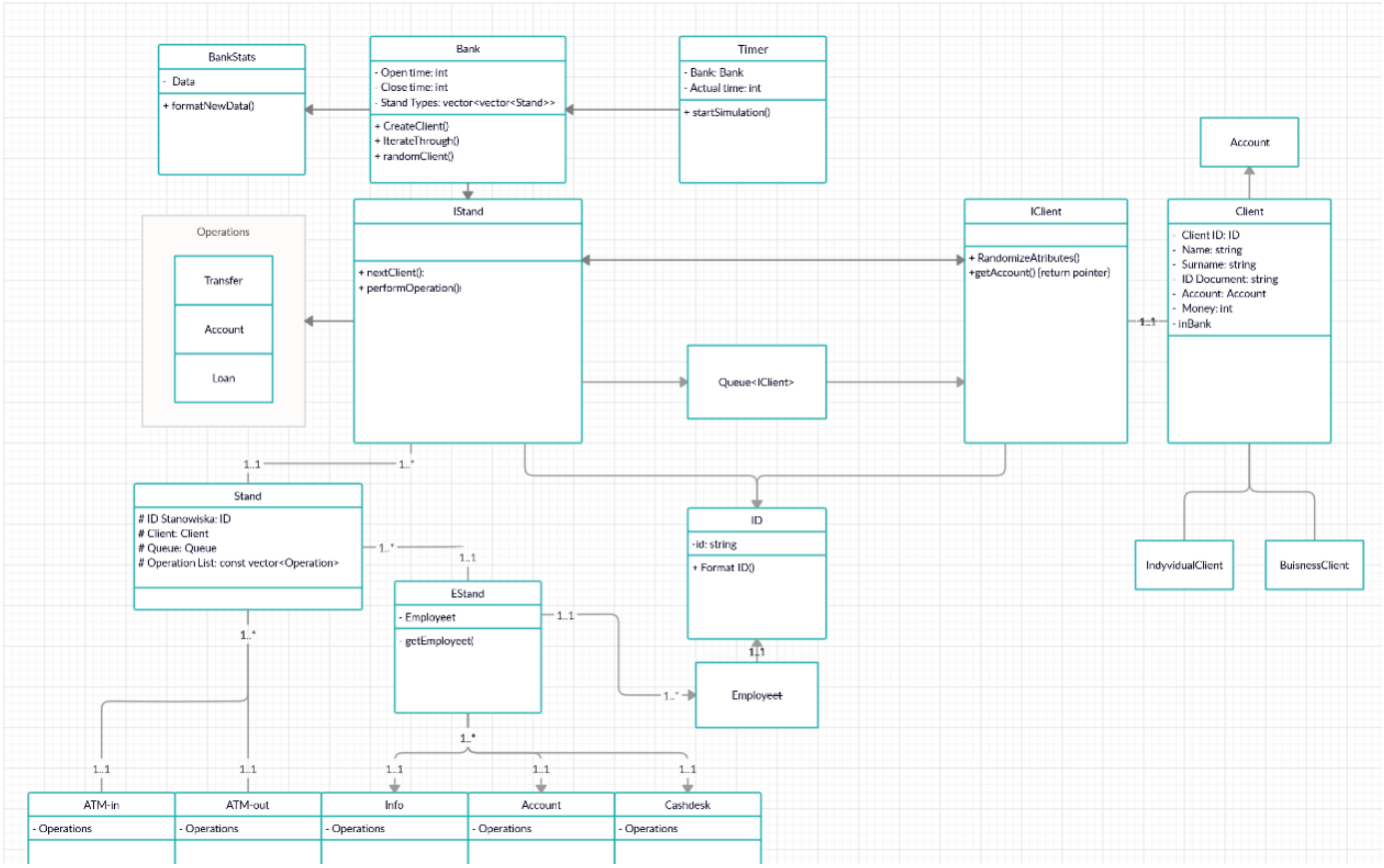
Uruchomienie pliku z podaniem lokalizacji pliku konfiguracyjnego:

Podczas tego sposobu uruchomiania programu użytkownik podaje lokalizację pliku, którego symulacja pobierze parametry. Jeśli program nie znajdzie odpowiedniego pliku lub dane będą podane w błędny sposób, symulacja wyświetli odpowiedni komunikat o błędzie.

Uruchomienie pliku z podaniem parametrów symulacji:

Podczas tego sposobu uruchomiania programu użytkownik podaje wszystkie 10 parametrów symulacji, gdy dane są podane błędnie lub zostanie podana ich błędna liczba, symulacja wyświetli komunikat o błędzie.

Hierarchia klas i ich relacji:



Skrót działania klas:

Stand i klasy pochodne:

Jest to klasa odpowiedzialna za symulację pojedynczego stanowiska. Najważniejszą częścią tej klasy jest metoda performOperation(), która różni się dla każdej klasy pochodnej, w zależności od tego jakie operacje można wykonać przy różnych typach stanowiska.

Client i Account:

Są to klasy symulujące jednego klienta i jego konto. Każdy klient ma jedno konto. Przechowuje informacje, które w rzeczywistości miałby klient wchodzący do Banku. Klient zawsze wie czy jest w banku i w kolejce do którego stanowiska czeka.

Bank:

Klasa zbierająca w całość okienka i klientów. Ważnym elementem jest dwuwymiarowy wektor, w którym każda kolumna jest różnym typem okienek w banku. Bank początkowo tworzy listę wszystkich swoich pracowników, klientów i okienek w zależności od parametrów podanych przez użytkownika w pliku. Posiada klasę pomocniczą BankStats, która dodatkowo przechowuje statystyki banku z danego dnia, takie jak ilość klientów, którzy założyli konta.

Timer:

Klasa dokonująca symulacji banku. Sama klasa Bank jest niezależna od Timera, i nie wie o jego istnieniu. Wyniki symulacji są zawsze zapisywane do pliku log.txt. Wyniki są także wypisywane do konsoli, a prędkość ich wypisywana jest zależna od parametrów programu.

Używanie Programu:

Program symuluje pracę banku podczas jednego dnia. Użytkownik może dowolnie zmieniać parametry symulacji, aby uzyskać wyniki jak najbardziej odpowiadające realnemu przypadkowi, który bada. Parametry w programie ustawia się w pliku settings.txt. Poniżej pokazane jest przykładowe uzupełnienie parametrów.

Parametry symulacji:

```
settings.txt
10000 <- Ilość zatrudnionych pracowników
10000 <- Ilość zapisanych klientów
1     <- Ilość wpłatomatów
1     <- Ilość bankomatów
1     <- Ilość okienek do wypłat
1     <- Ilość okienek informacji
1     <- Ilość okienek do obsługi kont
420   <- Czas rozpoczęcia pracy (w minutach: 420 - 8:00)
1320  <- Czas zakończenia pracy (w minutach: 1320 - 22:00)
3     <- Czas pauzy po wydarzeniu, podczas wyświetlania w konsoli (w sekundach)
```

Po uzupełnieniu parametrów i uruchomieniu symulacji zauważymy różne typy informacji o wewnętrznych wydarzeniach w banku:

- Monity o obecnej godzinie (pojawiają się jedynie, gdy o tej godzinie stało się coś wartego odnotowania).
- Informacje o nowym kliencie w banku.
- Informacja o okienku, które wybrał klient.
- Informacje o potrzebie nowo przybyłego klienta.
- Informacje o długości kolejki w okienku.
- Informacje o zakończeniu pracy pracownika z klientem.
- Informacje, o osobach które nie zdążyły załatwić swoich potrzeb.

Po zakończeniu symulacji pojawia się odpowiednia informacja.

Obsługa sytuacji wyjątkowych:

Ponieważ program posiada dość skromną interakcję z użytkownikiem, wykrywanie i obsługa błędów jest głównie skupiona na wczytywaniu danych. Program sprawdza, ile parametrów zostało mu podanych oraz czy wartości są odpowiednie (tj. numeryczne oraz nieujemne), dodatkowo porównywana jest liczba pracowników z ilością okienek przez nich obsługiwanych. W przypadku wykrycia błędu zwracane są wyjątki: BadOperation. Dodatkowo wykrywane są również błędy wynikające z braku istotnych plików, takich jak settings.txt lub pliki generatora imion i nazwisk. Zwracają one wyjątek FileNotFound, który jest wyłapywany w pliku Main.cpp.