Kawiarnia - dokumentacja projektu

PROI 22L
Wojciech Zarzecki
Bartosz Kisły

Opis przyjętych założeń

Program realizujący niniejszy projekt ma możliwość pracy w dwóch głównych trybach:

1. **Tryb symulacji** - przeprowadzenie symulacji działania kawiarni trwającej zadaną ilość cykli. Użytkownik ma możliwość wprowadzenia informacji o długości przerwy pomiędzy wyświetlaniem kolejnych informacji o symulacji, częstotliwości pojawiania się nowych klientów, nazwie pliku wyjściowego, do którego zapisana będzie cały wynik symulacji, a także nazw plików z danymi kawiarni do wczytania.
2. **Tryb administratora** - umożliwia modyfikację informacji o kawiarni (tj. menu, pracowników i stolików). Użytkownik wybiera nazwy plików, na których ma zamiar operować.

Podział programu na klasy

Cała architektura programistyczna projektu opiera się o wzorzec projektowy ***Model-View-Controller (MVC)***, co pozwala na ścisłą separację zadań pomiędzy różnymi fragmentami programu.

Model - odpowiada za logikę programu, przechowywanie oraz bezpośrednie operowanie na zbiorach danych.
Kontroler - odpowiada za komunikację z zasobami komputera, tj. bezpośrednie wczytywanie danych z plików/bazy danych lub zapisywanie do nich.
Widok - odpowiada za prezentację danych użytkownikowi oraz możliwość interakcji.

Klasy wchodzące w skład modelu:

- **MenuItem, Beverage, Dessert, Dish** - klasy przechowujące dane o produktach, które można zamówić w kawiarni. Klasa MenuItem jest klasą abstrakcyjną i stanowi bazę dla klas Beverage, Dessert i Dish.
- **Employee, Waiter, Cook** - klasy przechowujące dane o pracownikach kawiarni, a także udostępniające metody, które pozwalają wykonywać im

odpowiednie akcje w kawiarni. Analogicznie do punktu wyżej, klasa Employee jest abstrakcyjna i stanowi bazę dla klas Waiter i Cook.

- **Table, Customer, CustomersGroup** - klasy reprezentujące stolik, klienta i grupę klientów. Każdy stolik przechowuje kolekcję klas Customer, z kolei klienci, którzy mogą przychodzić do kawiarni całymi grupami, podczas ich tworzenia opakowywani są w klasę CustomersGroup, która tworzy z klientów jedną wspólną grupę o wspólnych preferencjach dot. towarzystwa innych klientów przy stoliku. Każdy klient posiada preferowany produkt do zamówienia.
- **Klasy bazodanowe** - hierarchia klas przechowująca zbiory obiektów
    - **IDatabase<T>** - interfejs, który znajduje się najwyżej w hierarchii klas bazodanowych. Zawiera zbiór metod czysto wirtualnych, które z założenia mają wykonywać takie akcje jak dodanie, usunięcie lub zwrócenie kolekcji przechowywanych elementów.
    - **TemplateDatabase<T>** - klasa będąca bazą dla tych baz danych, których obiekty wykonują akcje związane z symulacją oraz posiadają identyfikatory liczbowe. Umożliwia dodawanie obiektów z automatycznie przypisanym, unikatowym ID.
    - **MenuDatabase, CustomEmployeesDb, CustomTableDb** - właściwe klasy bazodanowe, które przechowują odpowiednio zbiór produktów, pracowników oraz stolików. Klasa MenuDatabase bezpośrednio implementuje interfejs IDatabase<T>, z kolei klasy CustomEmployeesDb oraz CustomTableDb dziedziczą po klasie TemplateDatabase<T>, która implementuje interfejs IDatabase<T>. Oprócz przechowywania samych danych, posiadają metody umożliwiające wykonanie zbiorczych operacji na całych kolekcjach obiektów.
- **Price** - pomocnicza klasa przechowująca cenę. Zawiera komplet przeciążonych operatorów, które pozwalają wykonywać różne obliczenia na cenach oraz porównywanie ich.
- **CafeModel** - główna klasa modelu kawiarni, która łączy wszystkie obiekty składające się na niego w całość. Przeprowadza ona właściwą symulację, kontroluje aktualizację stanów i bezpośrednio komunikuje się z kontrolerem.

Klasy wchodzące w skład kontrolerów:

- **DataService** - klasa obsługująca wczytywanie klas bazodanowych z plików i zapisywanie ich do plików. Bezpośrednio zarządza strumieniami plików.

Klasy wchodzące w skład widoku:

- **CafeView** - klasa obsługująca wyświetlanie wszystkich danych w konsoli. Obsługuje interakcję z użytkownikiem oraz wywołuje odpowiednie metody modelu.

Rola pliku main.cpp:
W pliku main.cpp znajduje się parser argumentów wywołania. Po ich odpowiedniej konwersji, inicjuje tryb symulacji lub tryb administratora (bezpośrednio wołając odpowiednie metody klasy CafeView).

Uruchomienie programu

Program rozpoczyna pracę w danym trybie po dodaniu odpowiednich argumentów wywołania:

- **`Cafe.exe -s`** - uruchomienie trybu symulacji. Dodatkowo, wymagane jest podanie argumentów kolejno:
  - **ilość cykli** symulacji do wykonania
  - **odstęp czasowy** pomiędzy wyświetlaniem kolejnych logów symulacji (w sekundach, wartość zmiennoprzecinkowa)
  - **częstotliwość pojawiania się nowych klientów** - w tym wypadku jest to ilość cykli, co które mają przychodzić nowi klienci
  - **nazwa pliku wyjściowego**
  - **nazwa pliku z informacjami o menu**
  - **nazwa pliku z informacjami o pracownikach**
  - **nazwa pliku z informacjami o stolikach**
- **`Cafe.exe -a`** - uruchomienie trybu administratora. Po uruchomieniu tego trybu, użytkownik ma możliwość podejrzenia zawartości wszystkich baz danych oraz interaktywnego wprowadzania nowych elementów oraz usuwania istniejących. Dane są odczytywane i zapisywane do tego samego pliku, który został podany przez użytkownika. Wymagane jest podanie kolejno argumentów:
  - **nazwa pliku z informacjami o menu**
  - **nazwa pliku z informacjami o pracownikach**
  - **nazwa pliku z informacjami o stolikach**

Bazy danych zapisują się do plików tekstowych w podfolderze textFiles. Znajdują się w nim pliki domyślne, które pozwalają na poprawne uruchomienie programu.

Przebieg symulacji

Kucharz, kelner oraz klient mają swoje grafy przejść pomiędzy poszczególnymi stanami, które są sprzężone ze sobą. Kelner odbiera zamówienia tylko od gotowych klientów, a przynosi rachunki tylko tym, którzy skończyli jeść. Czas przygotowania posiłków jest zależny od czasu przygotowania potrawy oraz ilości pracujących kucharzy, którym kelner przekazuje zamówienia.
Klienci są na bieżąco obsługiwani oraz dołączają nowy klienci, którzy mogą dołączyć do stolika lub wybrać nowy w zależności od chęci siadania razem.

Wykorzystane elementy biblioteki standardowej

Podstawową kolekcją był vector, umożliwia on swobodne dodawanie i usuwanie elementów oraz korzystanie z standardowych algorytmów np. wyszukania. W kolekcjach były przechowywane shared_ptr - wskaźniki do obiektów, umożliwiały wygodne przekazywanie obiektu, a nie kopii pomiędzy poszczególnymi obiektami, np. danie od klienta przez kelnera do kucharza. Do generowania liczb losowych wykorzystane zostały elementy z biblioteki <random> - standardowe dla języka C++.

Wyjątki

W programie mogą nastąpić różne sytuacje wyjątkowe, jak wczytanie nieprawidłowego pliku lub wprowadzenie niepoprawnych danych podczas dodawania dań/pracowników. Każda z tych sytuacji jest obsługiwana przez program poprzez wyświetlenie stosownego komunikatu o błędzie użytkownikowi - cały program jest odporny na różnego rodzaju błędy.

Dodatkowo, błąd, który powstaje podczas wprowadzania danych w trakcie dodawania nowych obiektów do baz danych, program nie kończy swojego działania - tryb administratora jest odpowiednio “zapętlony”, dlatego stanowi on praktyczną i wygodną dla użytkownika formę wprowadzania danych.

Podział pracy

Wojciech Zarzecki:

- utworzenie modelu pracowników oraz baz danych ich przechowujących
- utworzenie nadrzędnych klas bazodanowych
- utworzenie klasy bazodanowej dla stolików
- opracowanie przebiegu symulacji kolejnych stanów pracowników

- sprzężenie przebiegu stanów pracowników ze stanami klientów
- utworzenie klasy obsługującej pliki

Bartosz Kisły:

- utworzenie modelu produktów serwowanych w kawiarni oraz odpowiadającej im klasy bazodanowej
- utworzenie modelu klienta oraz stolika
- utworzenie klasy CafeModel
- utworzenie klasy CafeView
- obsługa argumentów wywołania

Cały projekt realizowany był z wykorzystaniem systemu Git, z uwzględnieniem odseparowania pracy na własnych gałęziach.