Wydział Elektroniki
i Technik Informacyjnych

POLITECHNIKA WARSZAWSKA

Programowanie obiektowe – dokumentacja wstępna

Temat:

Symulacja działania kina

Prowadzący:

mgr inż. Agnieszka Malanowska

Zespół:

Jakub Przesmycki 314241
Maksymilian Banach 314013

1. Założenia projektowe

- Kino będzie składać się z różnej liczby sal o określonych liczbach miejsc oraz kas.
- Klienci mogą przeglądać dostępne filmy oraz godziny seansów.
- Mogą wybierać miejsca w sali kinowej i dokonywać rezerwacji biletów.
- Kasjerzy mogą obsługiwać klientów, którzy chcą zakupić bilety.
- Muszą mieć możliwość wyboru odpowiedniego rodzaju biletu (normalny, ulgowy) oraz wybrania odpowiedniego filmu i godziny seansu.
- Klient ma możliwość kupna kilku biletów.
- Personel kina może aktualizować repertuar, dodając nowe filmy i określając godziny ich projekcji na kolejny tydzień.
- Mogą także usuwać filmy z repertuaru, jeśli zakończyły one swoje projekcje.
- System musi śledzić dostępność miejsc w poszczególnych seansach oraz rezerwacje dokonane przez klientów.
- Powinien zapewnić, żeby nie sprzedano więcej biletów, niż jest dostępnych miejsc w danej sali kinowej.
- System powinien uwzględniać różne rodzaje ulg, takie jak dla dzieci, studentów, seniorów (normalne, ulgowe, darmowe).
- Musi być możliwość obsługi sytuacji wyjątkowych tj. reklamacje (anulowanie rezerwacji, modyfikacja rezerwacji, zwrot biletów).
- System musi zapewnić bezpieczne przechowywanie danych osobowych klientów, takich jak imiona, nazwiska, adresy e-mail, a także danych finansowych, takich jak numery kart płatniczych.

2.1. Scenariusz pracownika

- Rozpoczęcie pracy o danej godzinie.
- Przypisanie kasy do pracownika (tylko do pracownika kas).
- Zarejestrowanie klienta.
- Wygenerowanie bądź nie wygenerowanie biletu(w zależności od wieku).
- Obsługa kolejnych klientów.

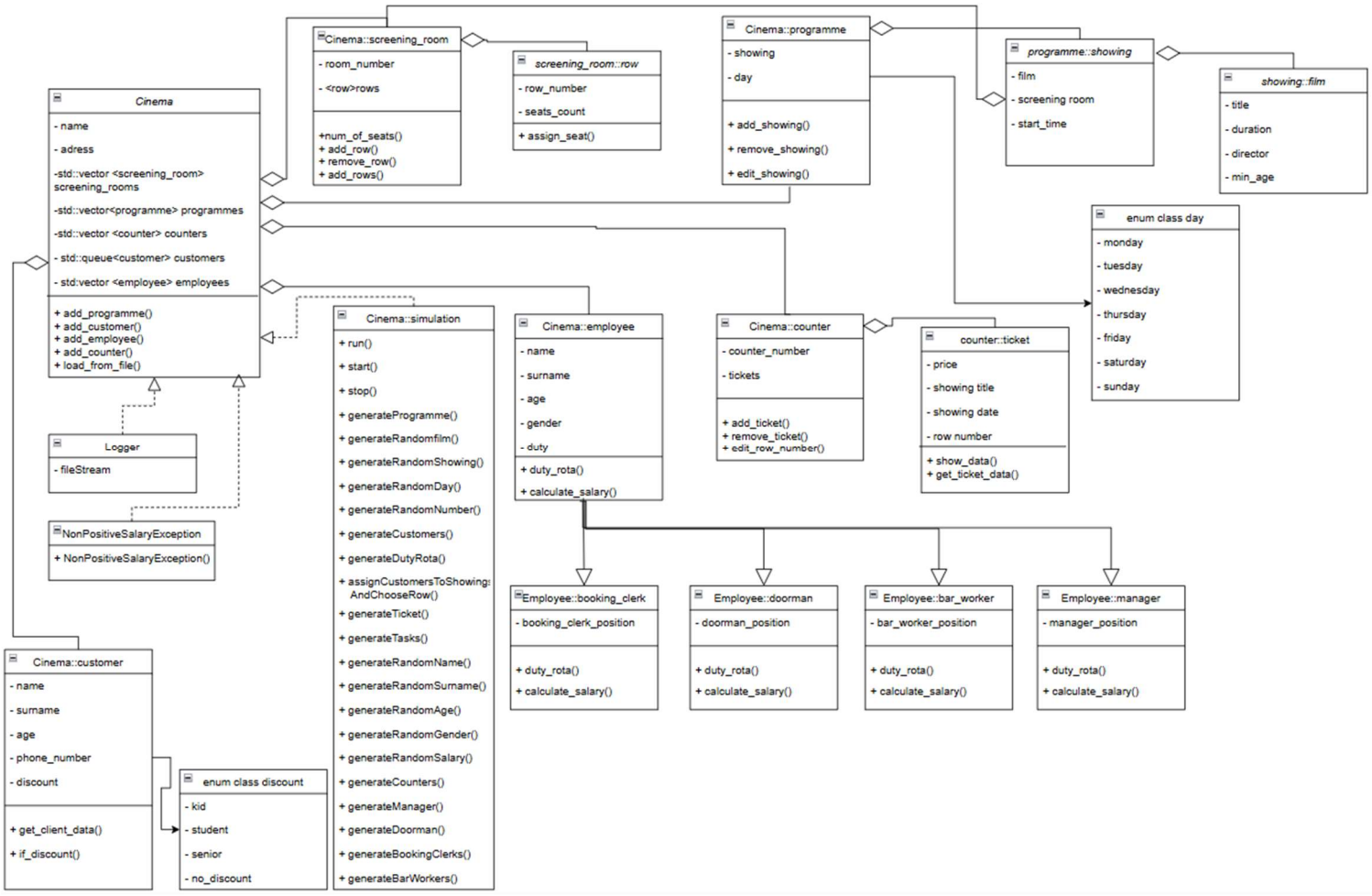
2.2. Scenariusz klienta

- Pojawienie się klienta.
- Wybór filmu.
- Wybór rzędu siedzenia.
- Odbiór biletu.
- Obejrzenie seansu.

2.3. Scenariusz kina

- Otwarcie kina.
- Stworzenie oraz wyświetlenie repertuaru.
- Otwarcie kas.
- Obsługa klientów.
- Zamknięcie kina.

3. Podział na klasy wraz z hierarchią klas oraz ich atrybuty i metody.



Opis metod:

- Counter:
  - Add_ticket() – metoda umożliwiająca edytowanie wygenerowanego już biletu.
  - Remove_ticket() – umożliwia usunięcia danego biletu.
  - Edit_row_number() – umożliwia edytowanie danych biletu.
- Customer:
  - If_discount() – sprawdzanie czy dany klient ma możliwość zniżki.
  - Get_client_data() – umożliwia zwrócenie danych klienta.
- Employee:
  - Calculate_salary() – możliwość obliczenia pensji pracowników
- Employee_list:
  - Add_employee() – możliwość dodania pracownika do wektora
  - Add_bar_worker() – możliwość dodania pracownika baru do wektora
  - Add_booking_clerk() – możliwość dodania kasjera do wektora

- Add_doorman() – możliwość dodania biletera do wektora
- Add_manager() – możliwość dodania managera do wektora
- Calculate_salary() – możliwość obliczenia sumy pensji pracowników
- Remove_employee() – możliwość usunięcia pracownika z wektora

- Cinema:
  - add_programme() – dodanie repertuaru do kina.
  - add_customer() – dodanie klientów do kina.
  - add_employee() – dodanie pracowników do kina.
  - add_counter() - dodanie kas do kina.
  - load_from_file() – metoda umożliwiająca załadowanie danych z pliku txt.

- Screening_room:
  - Num_of_seats() – obliczenie ilości miejsc w danej Sali.
  - Add_row() – dodanie rzędu do sali.
  - Add_rows() – dodanie kilku rzędów o tej samej ilości miejsc.
  - Remove_row() – usuwanie rzędu.
  -

- Row:
  - Assign_seat() – zajęcia miejsca w danym rzędzie.

- Programme:
  - Add_showing() – dodanie nowego seansu do repertuaru.
  - Remove_showing() – usunięcie seansu z repertuaru.
  - Edit_showing() – edycja seansów znajdujących się w repertuarze.

- Ticket:
  - Show_data() – umożliwia wyświetlenie danych biletu.
  - Get_ticket_data() – umożliwia zwrócenie danych biletu.
- Simulation:
  - Assign_customers_to_showings_and_choose_row() – przypisanie klientowi losowego seansu z danego dnia oraz miejsca w losowym rzędzie.
  - Generate_programme() – wygenerowanie losowego repertuaru na dany dzień.
  - Generate_random_film() – wygenerowanie losowego filmu
  - Generate_random_showing() – wygenerowanie losowego spektaklu.
  - Generate_random_day() – wygenerowanie losowego dnia
  - Generate_random_number() – generowanie losowej liczby
  - Generate_customers() –wygenerowanie losowych klientów.
  - Generate_duty_rota() – wygenerowanie obowiązków dla pracowników
  - Generate_ticket() – wygenerowanie biletu.
  - Generate_tasks() – pomocnicze generowanie obowiązków dla pracowników
  - Generate_random_name() – wygenerowanie losowego imienia.
  - Generate_random_surname() – wygenerowanie losowego nazwiska.
  - Generate_random_age() – wygenerowanie losowego wieku.
  - Generate_random_gender() – wygenerowanie losowej płci.

- Generate_random_salary() – wygenerowanie losowej wielkości pensji
- Generate_counters() – wygenerowanie losowych kas.
- Generate_manager() – wygenerowanie losowo managera
- Generate_doorman() – wygenerowanie losowo biletera
- generate_booking_clerks() – wygenerowanie losowo kasjera
- generate_bar_workers() – wygenerowanie losowo pracownika baru

4. Podział obowiązków

Maksymilian Banach – counter, ticket, customer, screening_room, row, ½ simulation

Jakub Przesmycki – Cinema, employee, programme, film, showing, ½ simulation

5. Opis symulacji

- Struktura argumentów wywołania program:

  - "3" → ilość dni symulacji.
  - "output1.txt." → plik .txt, w którym zostanie zapisana symulacja.
  - "cinema_data.txt" → plik .txt, z którego pobierane są dane do utworzenia obiektu kina.

- Struktura pliku wejściowego:

Plik wejściowy wykorzystywany jest do utworzenia obiektu kina. Plik składa się z 3 wierszy:

1) „Name" -> nazwa kina,
2) „Address" -> adres kina

Pozostałe atrybuty niezbędne do stworzenia obiektu kina generowane są losowo podczas rozpoczęcia symulacji.

- Opis symulacji:

1) Zdefiniowanie ilości dni wykonywania symulacji.
2) Zdefiniowanie danych kina.
3) Otwarcie kina.
4) Wygenerowanie oraz wyświetlenie repertuaru dla aktualnego dnia, a w tym:
   a. Wygenerowanie filmów,
   b. Wygenerowanie sal w których będą wyświetlane filmy,
   c. Wygenerowanie rzędów w salach.
5) Wygenerowanie pracowników kina, a w tym:
   a. Menadżera,
   b. Biletera,
   c. Pracowników kas,

    d. Pracowników baru.
6) Wygenerowanie kas oraz przypisanie ich do pracowników kas.
7) Wygenerowanie dzisiejszego harmonogramu zadań dla poszczególnych grup pracowniczych.
8) Wyświetlenie sumy wynagrodzeń pracowników (pokazanie działania polimorfizmu).
9) Wygenerowanie klientów.
10) Dla każdego klienta:
    a. Pokazanie, do której kasy podszedł.
    b. Wyświetlenie wieku poszczególnego klienta, filmu, który wybrał oraz wieku minimalnego tego filmu.
    c. Jeśli wiek klienta spełnia wymagania filmu, to wyświetlany jest wybrany rząd w sali oraz generowany jest bilet.
    d. Jeśli wiek klienta nie spełnia wymagań filmu, to wyświetlany jest komunikat, iż bilet nie może zostać wygenerowany ze względu na wiek klienta.
11) Po obsłużeniu wszystkich klientów, kino jest zamykane.
12) Symulacja powtarza się dla kolejnych dni.

6. Wykorzystane biblioteki STL

- **Random**: Używana do generowania liczb losowych, przydatna w symulacjach.
- **Ranges**: Przydatna do manipulacji i filtrowania danych w kontenerach, np. wektorach i listach.
- **Iostream**: Wykorzystywana do obsługi strumieni wejścia i wyjścia, np. do czytania i zapisywania danych z/do plików.
- **Algorithm**: Idealna do stosowania algorytmów sortowania, wyszukiwania i innych operacji na danych.
- **Fstream**: Używana do operacji wejścia/wyjścia na plikach, np. odczyt/zapis danych do plików tekstowych.
- **Sstream**: Przydatna do operacji wejścia/wyjścia na ciągach znaków, np. parsowanie danych z ciągów znaków.
- **Queue**: Wykorzystywana do implementacji kolejek, np. do zarządzania zdarzeniami w systemach czasu rzeczywistego.
- **Vector**: Idealna do przechowywania i manipulowania danymi w dynamicznej tablicy.
- **String**: Używana do operacji na ciągach znaków, np. manipulacja tekstem i analiza danych tekstowych.
- **Map**: Przydatna do przechowywania danych w postaci klucz-wartość, np. do mapowania identyfikatorów na dane.
- **Numeric**: Wykorzystywana do obliczeń numerycznych, np. obliczanie statystyk lub operacje matematyczne na danych.
- **List**: Używana do implementacji listy dwukierunkowej, przydatna, gdy potrzebne są częste operacje wstawiania/usuwania elementów na środku listy.
- **Memory**: Zapewnia funkcje do zarządzania pamięcią, np. alokacja i dealokacja pamięci dynamicznej.
- **Chrono**: Używana do operacji na czasie i dacie, np. pomiar czasu wykonania algorytmów.

- **Stdexcept**: Definiuje typy wyjątków standardowych, przydatna do obsługi wyjątków w programach.
- **Iomanip**: Przydatna do manipulacji strumieniami danych, np. formatowanie danych wyjściowych.
- **Thread**: Wykorzystywana do obsługi wielowątkowości, np. równoległe przetwarzanie danych dla poprawy wydajności.

7. **Sytuacje wyjątkowe i ich obsługa**
   - Non_positive_salary_exception – wyjątek, który wyrzucany jest w przypadku, kiedy wygenerowana pensja pracownika jest mniejsza niż 0.
   - Błąd otwarcia pliku do zapisu symulacji.
   - Brak wygenerowania biletu, gdy wiek klienta jest mniejszy od wieku minimalnego seansu.
   - Brak miejsca w rzędzie – wyrzucenie komunikatu o braku miejsca w rzędzie i wybór nowego rzędu.

8. **Przeprowadzone testy**

W ramach sprawdzenia utworzonej aplikacji, wykonano testy jednostkowe z użyciem biblioteki Catch2.

Na zdjęciu załączonym powyżej, możemy zauważyć, że wykonano 146 testów jednostkowych, które testują metody poszczególnych klas. Wszystkie testy są prawidłowe.

Testy klasy Ticket:

- Sprawdzono poprawność getterów i setterów dla ceny, tytułu seansu, daty seansu i numeru rzędu.
- Zweryfikowano, czy metoda show_data poprawnie wyświetla dane biletu.

Testy klasy Counter:

- Przetestowano dodawanie, pobieranie i usuwanie biletów.
- Zweryfikowano, czy możliwa jest edycja numeru rzędu.
- Sprawdzono, czy próba dodania biletu o istniejącym numerze wywołuje odpowiedni wyjątek.

Testy klasy Row:

- Sprawdzono poprawność getterów dla numeru rzędu i liczby miejsc.

- Zweryfikowano, czy możliwe jest przypisanie miejsca.

Testy klasy ScreeningRoom:

- Przetestowano dodawanie i usuwanie rzędów.
- Sprawdzono, czy możliwe jest pobieranie rzędu na podstawie indeksu.
- Zweryfikowano, czy możliwe jest pobranie liczby miejsc w sali.

Testy klasy Customer:

- Sprawdzono poprawność getterów i setterów dla danych klienta.
- Zweryfikowano, czy metoda if_discount zwraca poprawne wartości dla różnych typów zniżek.

Testy klasy Film:

- Przetestowano poprawność getterów dla tytułu, reżysera, długości filmu i minimalnego wieku.
- Zweryfikowano poprawność operatorów równości i strumieniowego.

Testy klasy Programme:

- Sprawdzono poprawność getterów dla seansów i dnia tygodnia.
- Zweryfikowano, czy możliwe jest dodawanie i usuwanie seansów, oraz edycja istniejącego seansu.
- Przetestowano poprawność operatora strumieniowego.

Testy klas związanych z pracownikami:

- Sprawdzono, czy możliwe jest poprawne tworzenie obiektów pracowników (barmani, pracownicy obsługi, doormani, menedżerowie) z odpowiednimi danymi, oraz czy obliczana jest poprawna pensja.

Testy klasy EmployeeList:

- Zweryfikowano, czy możliwe jest dodawanie pracowników różnych typów do listy pracowników.
- Sprawdzono, czy obliczana jest poprawna suma pensji pracowników.
- Zweryfikowano, czy możliwe jest usuwanie pracowników z listy.

Testy klasy Cinema:

- Sprawdzono, czy możliwe jest poprawne tworzenie obiektów `Cinema` z odpowiednimi danymi, takimi jak sale projekcyjne, programy, liczniki, klienci i pracownicy.
- Zweryfikowano, czy metody getterów (`get_name`, `get_address`, `get_screening_rooms`, `get_programmes`, `get_counters`, `get_customers`, `get_employees`) zwracają poprawne wartości.

- Zweryfikowano, czy możliwe jest dodawanie liczników do kina.
- Sprawdzono, czy po dodaniu licznika metoda `get_counters` zwraca poprawny rozmiar wektora liczników.
- Sprawdzono, czy możliwe jest poprawne wczytywanie danych kina (nazwa i adres) z pliku tekstowego.
- Zweryfikowano, czy po wczytaniu danych z pliku, metody `get_name` i `get_address` zwracają poprawne wartości.