

HOTEL

Hubert Gołębiowski (313804)

Wojciech Grunwald (311566)



1 Wstęp

Wszystkie pliki stworzone na potrzeby tego projektu znajdują się w folderze *PROJ_21Z_103_HOTEL* na gitlabie. Program zaimplementowaliśmy w *C++*, korzystając ze środowiska *Visual Studio Code*.

Projekt został zbudowany na podstawie konwencji programowania obiektowego - abstrakcji, hermetyzacji, dziedziczenia, polimorfizmu.

Program wywoływany jest z argumentami:

1. argv[1] - time_interval
2. argv[2] - guests.txt
3. argv[3] - number_of_guests
4. argv[4] - rooms.txt
5. argv[5] - number_of_rooms
6. argv[6] - workers.txt
7. argv[7] - number_of_workers

Przykładowa kompilacja:

```
PS E:\Studia\Informatyka\2 semestr\Programowanie Obiektowe\Hotel\proi_21z_103_hotel> g++ main.cpp group_of_guests.cpp hotel.cpp room.cpp hotel_interface.cpp waiter.cpp croupier.cpp receptionist.cpp room_service.cpp worker.cpp data.cpp Invalid_File_Exception.cpp Invalid_File_Header_Exception.cpp -o main
PS E:\Studia\Informatyka\2 semestr\Programowanie Obiektowe\Hotel\proi_21z_103_hotel> ./main 30 guests.txt 15 rooms.txt 20 workers.txt 4
```

Zawartości plików guests.txt, rooms.txt i workers.txt wyglądają według schematu:

```
id size acc_length cash
0 10 2 12457
1 5 4 10
2 2 4 2453
3 4 12 5055
```

Rysunek 1: guests.txt

```
id type number_of_beds area fee additional_furniture
0 normal 5 49 78 0
1 studio 5 58 95 1
2 normal 2 28 64 0
3 normal 3 40 71 0
4 apartament 1 24 57 1
5 normal 5 56 93 0
6 apartament 3 36 62 0
7 studio 5 48 74 1
```

Rysunek 2: rooms.txt

```
id profession cash
0 0 644
1 1 731
2 2 533
3 3 354
4 0 812
```

Rysunek 3: workers.txt

2 Wstępna dokumentacja

TEMAT PROJEKTU: **HOTEL**

AUTORZY: **Wojciech Grunwald, Hubert Gołębiowski**

1. W hotelu są dostępne różne rodzaje pokoi: jedno-, dwu-, trzy-, czteroosobowe, studia, apartamenty itp. Od rodzaju pokoju zależy liczba łóżek, powierzchnia, cena wynajmu, dostępność dodatkowych mebli itp.
2. W hotelu pracuje pewna liczba pracowników, wśród których można wyróżnić recepcjonistów, pokojówki, kucharzy, kelnerów itp. Każda grupa pracowników ma inne obowiązki.
3. Do hotelu przyjeżdżają i z hotelu wyjeżdżają goście. Gość może wybrać dowolną długość pobytu. Każdy gość musi się zameldować i wymeldować z hotelu.
4. Gość może przedłużyć pobyt, zamówić posiłek do pokoju lub udać się do hotelowej restauracji, zamówić taksówkę za pośrednictwem recepcji, poprosić o dodatkowe sprzątanie lub budzenie, skorzystać z hotelowych atrakcji itp.
5. Pokoje sprzątane są codziennie.
6. Należy zapewnić obsługę rozliczania klientów.

Podział pracy:

Hubert - **Czerwone** klasy(+**Różowe**)

Wojciech - **Zielone** klasy

Main wspólnie

Założenia:

Każdy apartament jest 4-osobowy

Każde Studio jest 2-osobowe

Obiekt typu Guests jest grupą od 1 do 4 osób, wszystkie działania podejmują wspólnie

Podział na klasy:

Klasa Hotel:

Pola:

```
string name
vector<*Room> rooms
vector<*Worker> workers
vector<*Guests> guests
double capital // kapitał
```

Metody:

```
void simulate
```

Klasa Room:

Pola:

```
string rank {2_person, 3_person, 4_person, Apartment, Studio}
int number_of_beds
double area
double fee
bool additional_furniture
Guests guests
```

Metody:

```
void change_number_of_guests
void get_cleaned
void next_day // metoda po upływie dnia aktualizuje length_od_accomodation
```

gości i pobiera opłatę jeśli goście kończą pobyt

Klasa Guests:

Pola:

int id
int length_of_accomodation
double cash

Metody:

void extend_the_time_of_accomodation
void order_the_meal
void go_to_the_restaurant
void demand_waking_up
void go_to_the_casino

Klasa Worker:

Pola:

int id
double cash
double salary

Metody:

virtual void work = 0;
void get_paid

Klasa Janitor: public Worker

Klasa Waiter: public Worker

Klasa Receptionist: public Worker

Klasa Cook: public Worker

3 Ostateczna dokumentacja

TEMAT PROJEKTU: **HOTEL**

AUTORZY: **Wojciech Grunwald, Hubert Gołębiowski**

Podział pracy:

Hubert - **Czerwone** klasy(+**Różowe**)

Wojciech - **Zielone** klasy

Main wspólnie

Założenia:

Obiekt typu Guests jest grupą od 1 do 5 osób (w przykładzie jest też użyta liczba 10, aby pokazać pełne działanie funkcji accomodation), wszystkie działania podejmują wspólnie

Podział na klasy:

Klasa Hotel: **Public Hotel_Interface**

Pola:

```
string name
vector<Room*> rooms
map<string, vector<Worker*>> workers
vector<Guests*> guests
int capital // kapitał
```

Metody:

```
void Add_Room
void Add_Worker
void Add_Guests
void Accomodation
void Simulate
void Info
```

Klasa Room:

Pola:

```
string rank {normal, apartment, studio}
int number_of_beds
int area
int fee
bool additional_furniture
GroupOfGuests& guests
```

Metody:

```
void change_guests
void checkout
```

Klasa GroupOfGuests

Pola:

```
int id
int size
int length_of_accomodation
```

```
int cash
int room_id
Metody:
int extend_the_time_of_accomodation
int order_the_meal
int go_to_the_restaurant
bool demand_taxi
int go_to_the_casino
int give_tip
```

Klasa Worker:

```
Enum Profession {croupier, room_service, waiter, receptionist}
```

Pola:

```
int id
int cash
int salary
Profession profession
```

Metody:

```
virtual void work = 0;
void get_paid
void receive_tip
```

Klasa Croupier: public Worker

Metody:

```
void work
```

Klasa Waiter: public Worker

Metody:

```
void work
```

Klasa Receptionist: public Worker

Metody:

```
void work
```

Klasa RoomService: public Worker

Metody:

```
void work
void clean_room
```

Klasa Data:

Pola:

```
string g_address
string r_address
string w_address
```

Metody:

```
void loadGuests
void loadRooms
void loadWorkers
```

Klasa Hotel_Interface:

```
virtual void Add_Room = 0  
virtual void Add_Worker = 0  
virtual void Add_Guests = 0  
virtual void Accomodation = 0  
virtual void Simulate = 0  
virtual void Info = 0
```

Klasa Invalid_File_Exception: invalid_argument

Klasa Invalid_File_Header_Exception: invalid argument

4 Symulacja

Symulacja wywoływana jest przez publiczną metodę `Simulate()` z klasy `Hotel`. Polega ona na tym, że w każdej iteracji pętli na ekran wypisywana jest informacja o numerze dnia, stanie kapitału hotelu, liście gości, pokoi i pracowników.

Następuje operacja zakwaterowania dla każdego z gości, którym, jeśli jest to możliwe, przypisywany jest pokój o najmniejszej dla nich cenie. Następnego dnia każdej grupie gości losowana jest funkcja, korzystając z biblioteki `chrono` i `random`, spośród kilku dostępnych w klasie *GroupOfGuests*:

1. `extend_the_time_of_accomodation`
2. `order_the_meal`
3. `go_to_the_restaurant`
4. `demand_taxi`
5. `go_to_the_casino`

Do każdej z tych funkcji, oprócz pierwszej, wywoływana jest także funkcja `give_tip`, która losuje, czy pracownik otrzyma od gościa napiwek czy nie. Na koniec dnia sprzątane są pokoje, wypłacana jest pensja pracownikom i wymeldowane są osoby, które nie spełniają już kryteriów potrzebnych do zamieszkania.

4.1 Przykładowy wycinek symulacji

```
=====
Welcome to our Hotel Paradise.
Hotel will work in the span of 30 days.
=====
When you get into a hotel room, you lock the door, and you
know there is a secrecy, there is a luxury, there is a fantasy
There is comfort. There is reassurance.
=====

=====
DAY 1
=====
$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$
CAPITAL OF THE HOTEL: $10000000
$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$
```


Guests no. 6 from room 11 went to casino and won 2428\$
Croupier 0 served guests no. 6 in casino
Guests no. 6 from room 11 decided to give no tip

Guests no. 7 from room 2 want hotel receptionist to call for a taxi
This course will cost them 88\$
Receptionist 3 called for taxi for guests no. 7
Guests no. 7 from room 2 decided to give 6\$ to worker 3

Guests no. 8 from room 18 want hotel receptionist to call for a taxi
This course will cost them 92\$
Receptionist 3 called for taxi for guests no. 8
Guests no. 8 from room 18 decided to give no tip

Guests no. 9 from room 7 ordered meal worth 45\$
Room service 1 delivered food to room 7
Guests no. 9 from room 7 decided to give 11\$ to worker 1

Guests no. 10 from room 12 decided to extend their stay by 7 days

Symulacja kończy się po przekroczeniu określonej przez time_interval liczby dni.

Guests no. 6 from room 11 went to casino and won 2428\$
Croupier 0 served guests no. 6 in casino
Guests no. 6 from room 11 decided to give no tip

Guests no. 7 from room 2 want hotel receptionist to call for a taxi
This course will cost them 88\$
Receptionist 3 called for taxi for guests no. 7
Guests no. 7 from room 2 decided to give 6\$ to worker 3

Guests no. 8 from room 18 want hotel receptionist to call for a taxi
This course will cost them 92\$
Receptionist 3 called for taxi for guests no. 8
Guests no. 8 from room 18 decided to give no tip

Guests no. 9 from room 7 ordered meal worth 45\$
Room service 1 delivered food to room 7
Guests no. 9 from room 7 decided to give 11\$ to worker 1

Guests no. 10 from room 12 decided to extend their stay by 7 days

Została także zaimplementowana obsługa wyjątków dla wczytywania plików:

```
PS E:\Studia\Informatyka\2 semestr\Programowanie Obiektowe\Hotel\proi_21z_103_hotel> ./main 30 error.txt 15 rooms.txt 20 workers.txt 4
Złapano wyjątek typu Invalid_File_Exception: Invalid file name: error.txt
```

```
id sizeERROR acc_length cash
0 10 2 12457
1 5 4 10
2 2 4 2453
```

```
PS E:\Studia\Informatyka\2 semestr\Programowanie Obiektowe\Hotel\proi_21z_103_hotel> ./main 30 guests.txt 15 rooms.txt 20 workers.txt 4
Złapano wyjątek typu Invalid_File_Header_Exception: Invalid file header: id sizeERROR acc_length cash
PS E:\Studia\Informatyka\2 semestr\Programowanie Obiektowe\Hotel\proi_21z_103_hotel>
```