Kacper Klassa
Bartosz Kosiński
Jakub Kowieski

Link do repozytorium:
https://gitlab-stud.elka.pw.edu.pl/jkowiesk/seaside_hotel

PROI projekt: Hotel w nadmorskim kurorcie

1. Opis przyjętych założeń

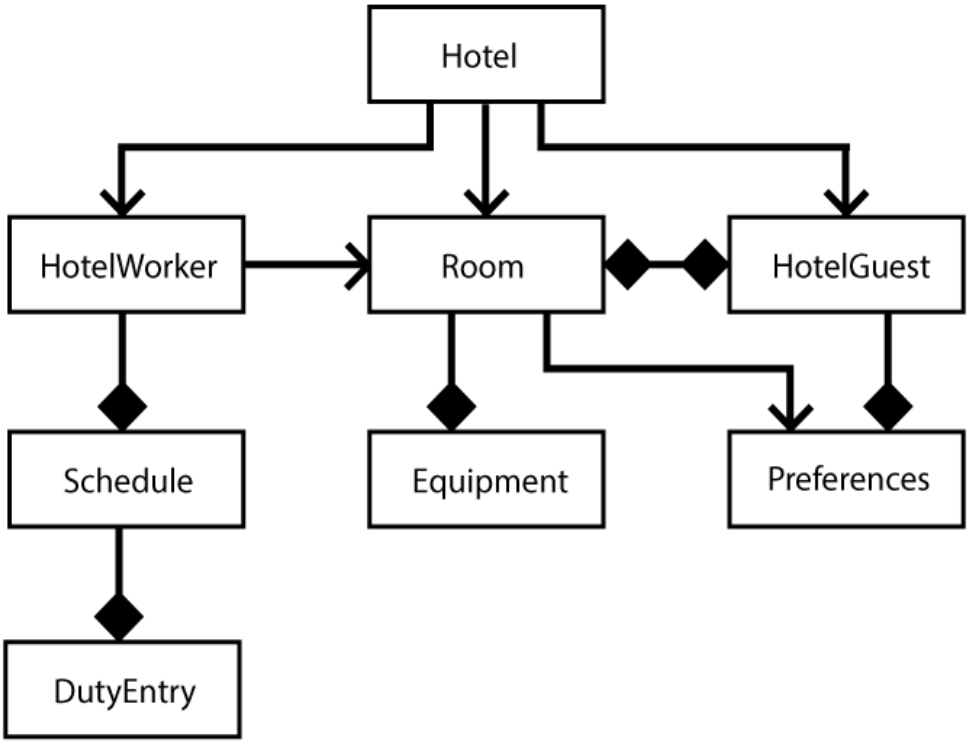
Projekt miał na celu symulację działania hotelu w nadmorskim kurorcie, przyjęliśmy następujące założenia:

- symulację działania będzie imitowało pętla, w której jeden pełny obrót będzie reprezentował jeden dzień
- podczas tego dnia wystąpi pewna ilość zdarzeń, podana przez użytkownika
- oprócz zdarzeń losowych nastąpią takie zdarzenia jak: przyjazd jednego gościa; posprzątanie pokoi; sprawdzenie czy rezerwacje gości są ważne danego dnia
- użytkownik oprócz ilości zdarzeń, poda jeszcze dwa parametry - długość symulacji (liczona w dniach) oraz ilości pokoi w hotelu
- w hotelu dostępne są pokoje o różnej pojemności - jedno, dwu, trzy i czteroosobowe pokoje oraz wspólne sypialnie dla większej ilości gości
- każdy pokój posiada swoje wyposażenie w tym gwarantowane łóżko, posiada on również informacje takie jak: numer pokoju, powierzchnia, piętro na którym się znajduje
- ilość pracowników hotelu zależy od aktualnej ilości używanych pokoi
- istnieją cztery typy pracowników - kelner, kucharz, sprzątacz i recepcjonista, działania wykonywane przez pracowników podczas symulacji są uzależnione od typu, do którego dany pracownik należy
- każdy z pracowników posiada własną pensję, która jest wypłacana co tydzień oraz terminarz, w którym opisane są godziny, o których wykonywane są konkretne czynności, przy czym jeden pracownik nie może wykonywać więcej niż jedną czynność na godzinę
- do hotelu przyjeżdżają różne typy gości, gdzie każdy z nich posiada unikatową wartości zniżki
- podczas symulacji gość może: zamówić taksówkę i zamówić dodatkowe umeblowanie do pokoju
- klienci rozliczani są podczas końca swojego pobytu

- imiona pracowników oraz gości są wczytywane z pliku tekstowego o formacie “<imię>;<nazwisko>\n”
- maksymalna wartość parametru “busyness” (ilość pracowników na liczbę pokoi) w symulacji wynosi 10

2. Opis hierarchii klas oraz relacji pomiędzy nimi

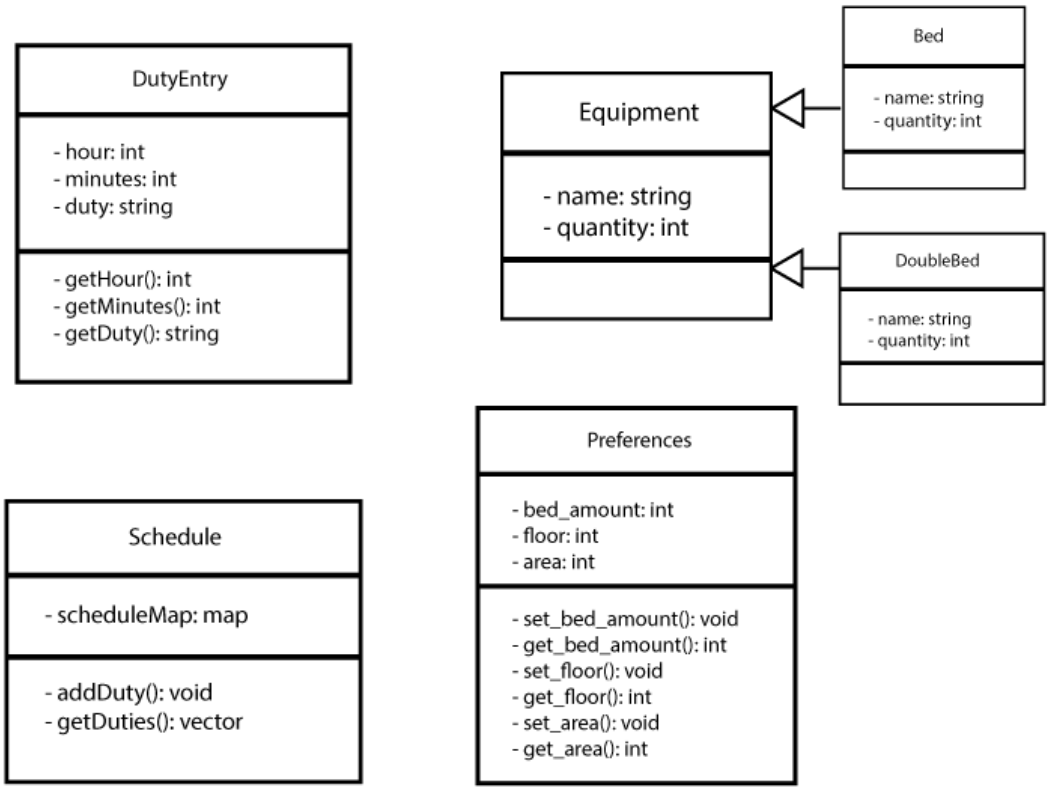
Ogólne relacje pomiędzy najważniejszymi klasami zawarte są poniżej (w diagramie zawarte są główne klasy wykorzystywane podczas symulacji, gdzie ich dokładny opis oraz lista klas dziedziczących po nich znajduje się w dalszej części dokumentacji)



Zaczynając od góry, klasa Hotel będzie wykorzystywać najważniejsze trzy klasy programu: pokój, pracownika oraz gościa hotelowego. Jej zadaniem jest ogólne zarządzanie pracą całego hotelu. Relacja klas gościa oraz pokoju polega na wzajemnej wymianie informacji czy dany gość rezerwuje jeszcze konkretny pokój. Pracownik wykorzystuje klasę pokoju do wykonania czyszczenia wszystkich pokoi. Klasa preferencji występuje jako atrybut klasy gościa - każdy z nich posiada własne upodobania co do atrybutów pokoju. Terminarz zawiera informacje na

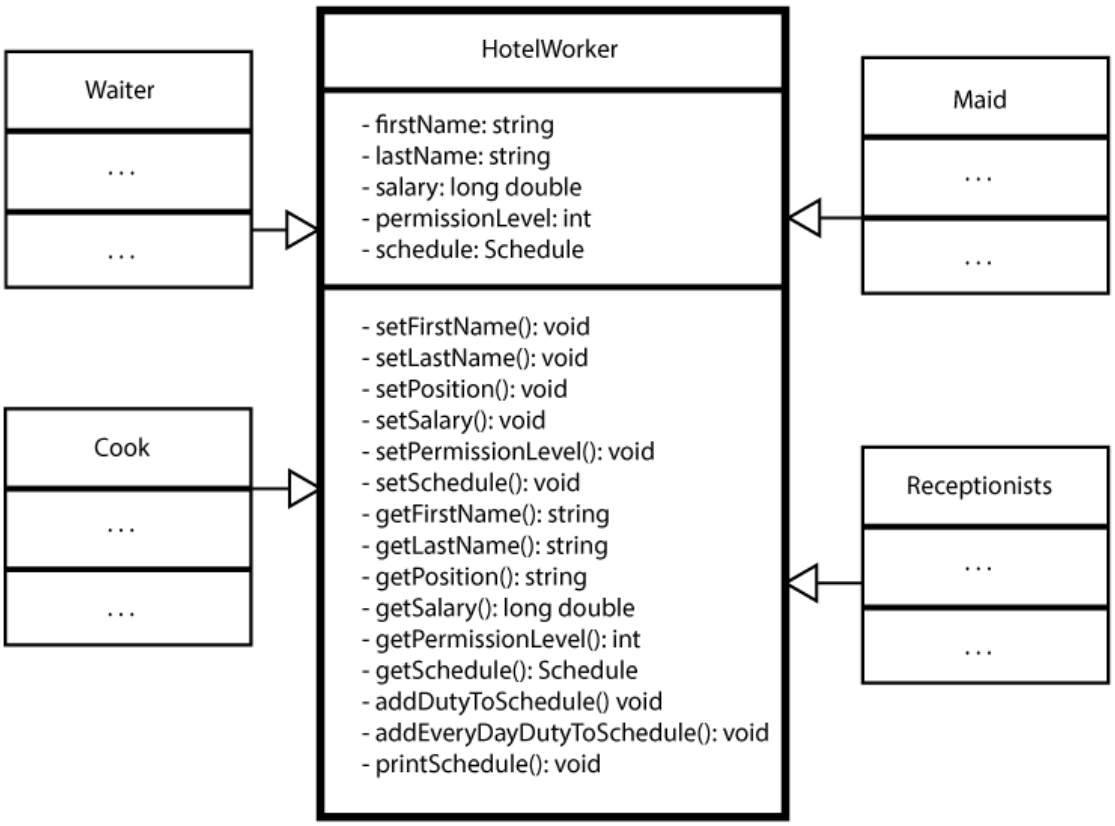
temat długości wykonywania czynność oraz ich godziny. Każdy z pracowników ma własny terminarz.

Diagram pokazujący szczegóły klas drugorzędnych:



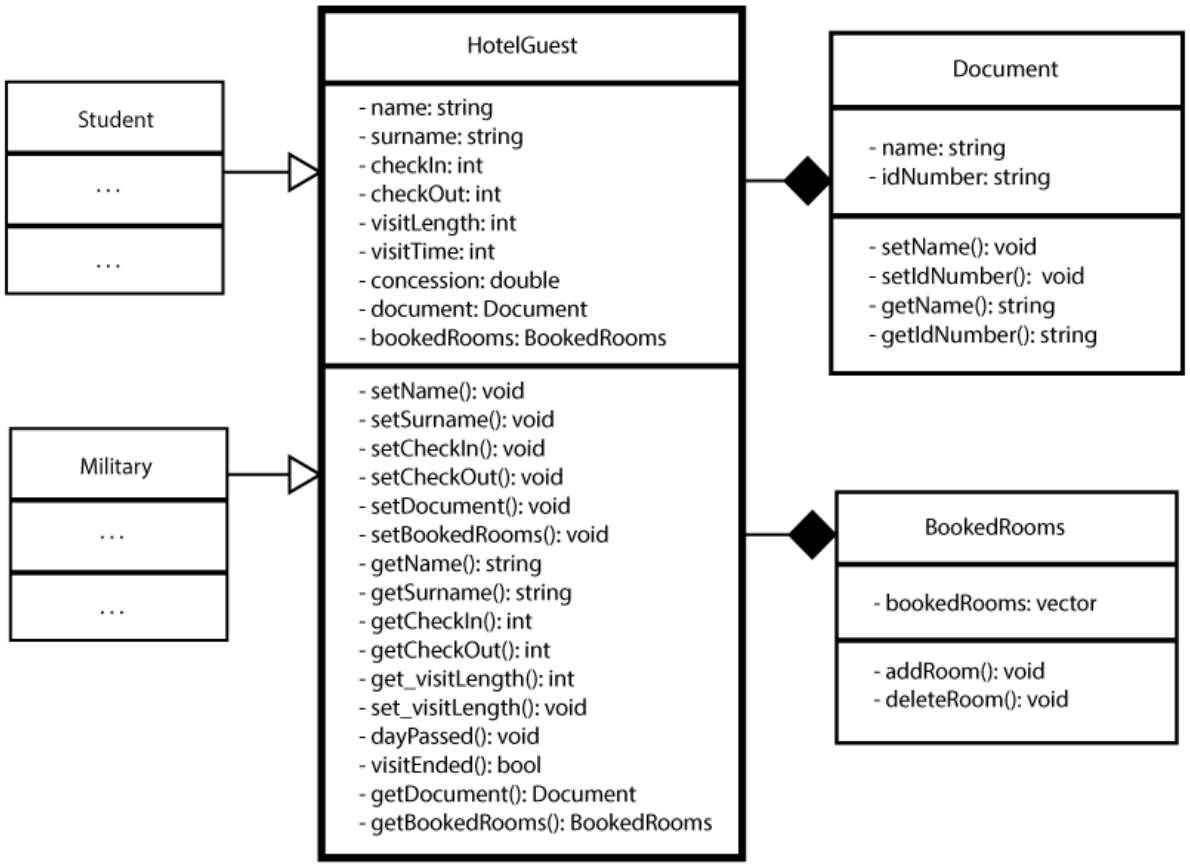
Wyposażenie pokoju (Equipment) posiada dwie klasy dziedziczące - łóżko jedno (Bed) oraz dwu-piętrowe (DoubleBed). Metody powyższych klas ograniczają się do zmiany oraz wyświetlenia ich atrybutów. Klasa preferencji posiada jedynie metody odpowiedzialne za wczytanie oraz zmienienie jej atrybutów podobnie jak klasa terminarzu (Schedule) oraz klasa DutyEntry, zawierająca informacje o długość wykonywanej czynności oraz czasu, w którym jest ona wykonywana.

Wykres klasy pracownika hotelowego:



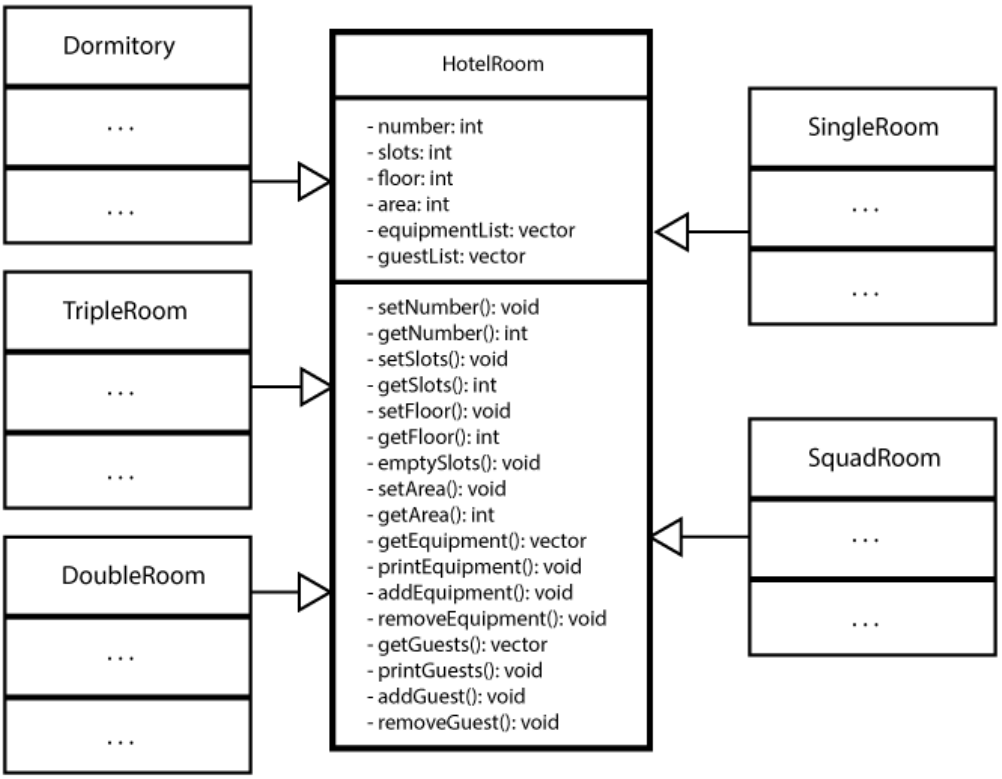
Tak jak wspomniano istnieją cztery klasy dziedziczące po pracowniku hotelowym - każda z nich posiada z góry ustawione czynności, które mają narzucone długości trwania oraz czas, w którym są wykonywane. Przykładowo klasa kucharza (Cook) posiada obowiązek “Preheat oven”, który może zostać wykonany o z góry ustalonej wcześniej godzinie. Głównymi metodami są również operacje dostępu oraz zmiany atrybutów klasy.

Diagram klasy gościa hotelowego:



Klasa gościa hotelowego posiada takie atrybut jak godzina zameldowania, wymeldowania oraz długość pobytu - informacje te będą kluczowe podczas przeprowadzania symulacji działania programu. Istnieją dwie klasy dziedziczące posiadające własną wartość atrybutu ‘concession’ - zniżki, oprócz tego główna klasa korzysta z dwóch pośrednich klas - dokumentu zawierającego nazwę oraz jego typ i ilości zarezerwowanych obecnie pokoi. Reszta metod tej klasy zawiera odczytanie lub zmianę jej atrybutów.

Wykres klasy pokoju hotelowego:

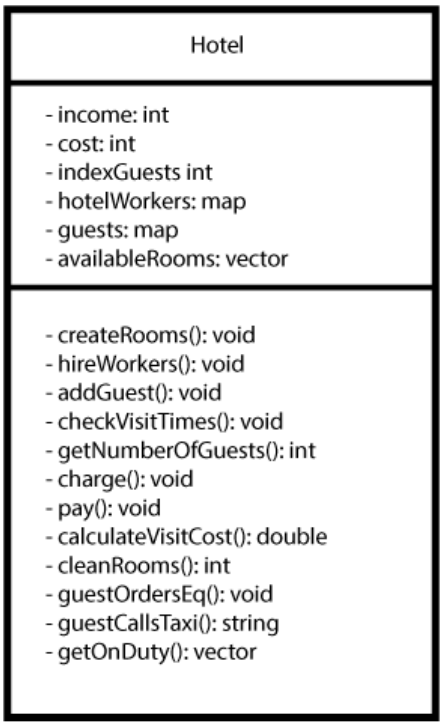


Tak jak napisano w założeniach ogólnych - istnieje pięć typów pokoi:

- jednoosobowe zawierają jedno jednoosobowe łóżko
- dwuosobowe zawierają dwa jednoosobowe łóżka
- trzyosobowe zawierają jedno jednoosobowe oraz jedno dwuosobowe łóżka
- czteroosobowe zawierają dwa jednoosobowe oraz jedno dwuosobowe łóżka
- dormitory - pokój dla większej ilości gości - zawiera dowolną ilość jednoosobowych łóżek

Ponad to główna klasa posiada atrybuty odpowiadające za: numer pokoju, ilość łóżek, piętro pokoju, wartość powierzchni, lista umeblowania pokoju oraz lista gości, którzy aktualnie rezerwują dany pokój

Diagram klasy hotel:



Klasa ta posiada atrybuty takie jak: przychód, cena wynajmu, listę gości, listę pracowników oraz listę aktualnie dostępnych pokoi. Metody tej klasy odpowiedzialne są za dodawanie nowych pokoi, nowych pracowników, gości; sprawdzanie czy goście nie skończyli swojego pobytu; zwiększenie przychodu; obliczanie kosztu pobytu gościa; sprzątanie pokoi; pobieranie dostępnych obowiązków pracowników na konkretną godzinę oraz realizowanie działań gościa - zamawianie taksówki oraz dodatkowego umeblowania do pokoju.

3. Opis sposobu testowania programu

Program został przetestowany na dwa sposoby:

1. Przeprowadzanie testowych symulacji dla wcześniej napisanych plików tekstowych z informacjami o pracownikach i gościach oraz o dowolnej liczbie zdarzeń wykonywanych podczas jednego dnia, ilości dni oraz pokoi. Wszystkie zdarzenia wykonywane podczas symulacji zapisywane zostają do danego pliku tekstowego oraz wyświetlają się w oknie konsoli, tak aby

użytkownik mógłby je zobaczyć. Program działa w oczekiwany sposób, dla dowolnych danych pobieranych od użytkownika. Ten sposób testowania programu okazał się najbardziej problematyczny, gdyż zanim został przeprowadzony wymagał on ujednolicenia całej pracy grupy oraz ewentualnego usunięcia błędów występujących w programie.

2. Przez testy jednostkowe - została wykorzystana biblioteka do testów jednostkowych Google Test, która umożliwia napisanie prostych testów sprawdzających poprawność działania konkretnych metod danej klasy bez bezpośredniego pisania klas testowych. Sprawdzenie polegało na założeniu oczekiwanej wartości zwracanej przez daną metodę, operującą na konkretnych danych. Takie testowanie dało pewność w dalszych modyfikacjach programu, gdyż można od razu sprawdzić czy po danych poprawkach przypadkiem nie została zmieniona dana funkcjonalność, która nie powinna zostać zmieniona.

4. Wykorzystane elementy biblioteki STL

Program wykorzystuje następujące biblioteki STL:

- <vector> do zaimplementowania prostych tablic jednowymiarowych zawierających np. ilość aktualnie dostępnych pokoi
- <map> do zaimplementowania tablic, w których elementy posiadają klucz oraz wartość, np. dla listy pracowników
- <string> do pracy z łańcuchami znaków
- <utils> zastosowano obiekt std::pair, pozwalający na grupowanie obiektów w pary
- <functional>
- <algorythms> wykorzystano metodę std::sort

5. Opis zidentyfikowanych sytuacji wyjątkowych i ich obsługi

Podczas testowania wynikały błędy, typu “Stack mashing”, ale udało się je rozwiązać poprzez debugowanie programu