

Lab 5.3.2 Flight booking system: part 2

Objectives

Familiarize the student with:

- modelling real-world entities with classes and objects;
- limiting acceptable input range.

Scenario

Let's continue working on our booking system.

You may remember in our last task that we had no limit on the number of reserved seats.

Airlines often allow the overbooking of flights, expecting that some passengers will not make it.

Modify the constructor so that it will not allow more than 105% reservation of the total capacity.

Also for a negative number of reservations, set the number to 0.

We might also want to be able to add new reservations or cancel them. Provide a way for the user to do this via a simple interface.

- The command "add [n]" will try to add n reservations to the flight.
- The command "cancel [n]" will try to cancel n reservations from the flight.
- If an operation fails for any reason, the program will issue the message "Cannot perform this operation"
- The command "quit" will stop execution of the program.

Use the code from the previous lab as a starting point.

```

#include <iostream>

class FlightBooking {
public:
    FlightBooking(int id, int capacity, int reserved);
    void printStatus();
    bool reserveSeats(int number_ob_seats);
    bool cancelReservations(int number_ob_seats);
private:
    int id;
    int capacity;
    int reserved;
};
// ...
FlightBooking::FlightBooking(int id, int capacity, int reserved)
{
    // Save data to members according to limits
}

bool FlightBooking::reserveSeats(int number_ob_seats)
{
    // try to add reservations and return 'true' on success
    // keep the limits in mind
    return false;
}

bool FlightBooking::cancelReservations(int number_ob_seats);
{
    // try to cancel reservations and return 'true' on success
    // keep the limits in mind
    return false;
}

int main() {
    int reserved = 0,
        capacity = 0;
    std::cout << "Provide flight capacity: ";
    std::cin >> capacity;

    std::cout << "Provide number of reserved seats: ";
    std::cin >> reserved;

    FlightBooking booking(1, capacity, reserved);

    std::string command = "";
    while (command != "quit")
    {
        booking.printStatus();
        std::cout << "What would you like to do?: "
        std::cin.getline(command);

        // handle the command
    }

    return 0;
}

```

Example input

```
100
50
add 4
cancel 200
quit
```

User prompts were omitted in the output

Example output

```
Flight 1 : 50/100 (50%) seats reserved
Flight 1 : 54/100 (54%) seats reserved
Cannot perform this operation
```

Example input

```
180
200
add 1
quit
```

User prompts were omitted in the output

Example output

```
Flight 1 : 189/180 (105%) seats reserved
Cannot perform this operation
```