Object-Oriented Programming

DAT3

Jens Tinggaard
Study No. 2022xxxx

January 9, 2024

1 Reservation class

This class is found in the file `Reservation.java`.

2 FreeSlot class

This class is found in the file `FreeSlot.java`.

3 Room class

This class is found in the file `Room.java`.

4 Group class

This class is found in the file `Group.java`.

5 Modified Reservation class

Also in the file `Reservation.java`.

6 BookingSystem class

This class is found in the file `BookingSystem.java`.

The required methods are implemented and working as intended, however due to restrictions in the usage of `System.in` in a `Scanner()`, running multiple of the methods will cause the program to crash. This could be solved by passing around the `Scanner` object, but I deemed this irrelevant to the assignment.

Furthermore, I inded up not using the `FreeSlot` class, as any time not occupied in the calendar is considered free.

7 Reservation Periods

Given a reservation period where group oop-01 has reserved room 1.1.01 next Monday, Wednesday, and Friday from 9:00 until 11:00, group oop-02 has reserved 1.1.02 next Tuesday from 13:00 until 15:00, and group oop-03 hasn't reserved any room, the following requests are made:

- *group oop-01 requests the reservation of room 1.1.03 on Thursday from 9:00 until 11:00.*
- *group oop-02 requests the reservation of room 1.1.03 on Thursday from 8:30 until 10:30.*
- *group oop-03 requests the reservation of room 1.1.03 on Thursday from 9:30 until 10:30.*

When these requests are handled, which of them would be approved by your system? Justify your answer and possible deviations from what is described in this assignment.

The last request would be approved, as group oop-03 does not have any confirmed reservations. This is fulfilled by implementing the `Comparable<T>` interface in the `Group` class, like such:

```
@Override
public int compareTo(Group other) {
    int reservedDiff = minutesReserved - other.minutesReserved;
    return reservedDiff == 0 ? noMembers - other.noMembers : reservedDiff;
}
```

In the `handleRequests()` method in the `BookingSystem`, all groups are compared to one another in case they overlap, and the one with the highest priority is selected.

8 Design Patterns

Identify at least one design pattern that you used in your solutions to the previous tasks. If you didn't use any design pattern, propose changes to your code to follow at least one design pattern (you do not have to update the code). Describe in a concise but precise manner the use of the design pattern, possible advantages and limitations.

The `Reservation` and `FreeSlot` classes are both implementations of the `Comparable<T>` interface, making them follow the composite pattern. The advantage of this, is that any element in a Room's calendar, can be compared, regardless of weather they are a `Reservation` or a `FreeSlot`, as they are treated uniformly.

9 Consistency

Assess how well your system ensures reservation consistency, i.e., rooms can only be reserved when they are available. If your system doesn't ensure reservation consistency, propose changes to your code to ensure reservation consistency (you do not have to update the code). Describe in a concise but precise manner the way your code (or proposed changes) prevents that inconsistent reservations are accepted and the possible feedback given to the client code and user interacting with the system.

It ensures consistency.

When all the reservation requests are handeled, the ones with the lowest minutes reserved (if the same, the one with the most members), have first claim over the others. The minutes reserved are dynamicaly updated, so when the next request is processed (even during the same `handleRequests()` call), is made, the consistency is ensured.

It is not possible to reserve a Room that has already been booked, even if the group making the reservation has lower minutes reserved, as the booked room is final. In that case, the the reservation is simply discarded.

10 Principles in the Object-Oriented Paradigm

This question is rather odd, as the `showCalendars()` method is so few lines. But the method does use the inherited `toString()`, which also uses overloading beheind the curtain, making it follow some principles from the object-oriented paradigm.