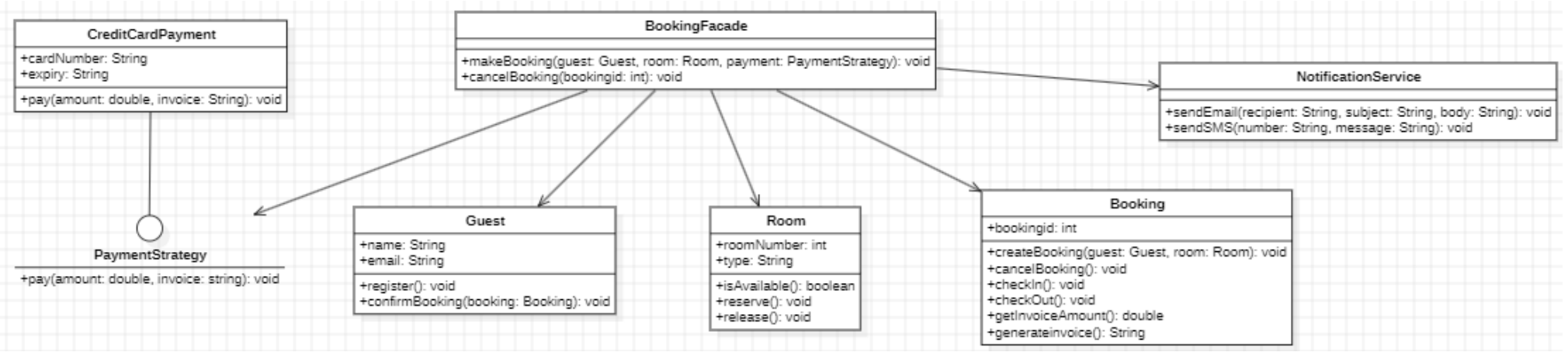
**Façade design pattern of the Booking process (Daron Delvina, Orgest Baçova):**




We have chosen to use this pattern for a couple of reasons. Although the original layout of the classes in the original class diagram are correct, the usage of the façade pattern offers a few more advantages.

Operations like creating a new booking or cancelling an existing one would require deep understanding of multiple classes and the relationships between them making the code prone to errors and difficult to maintain.

This pattern can encapsulate interactions between the Guest, Room, Booking and payment classes. In doing so, the complexity of the operations is reduced.

System-wide changes are made more manageable. For example, if we want to add a new workflow, we would only need to make changes to the façade implementation.

Also, in the original class diagram, a complete booking process would require direct interaction with more classes, meanwhile the façade pattern reduces it to a single method call.