Hotel Reservations™ Presentation

Nolan Cretney, Nathan Carmine,
Jackson Mediavilla, Monte Anderson

Introduction - Hotel Reservations™

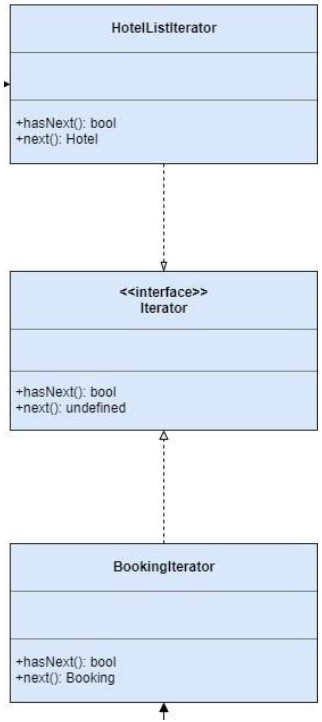
- Our application is a hotel reservation system
  - Reads in hotel names and locations
  - Lists out names, locations, rooms available, etc, from our database.

- The user can search for hotels by location
  - Reserve and cancel bookings
  - Our card verification system uses the Luhn algorithm

Primary Use Cases

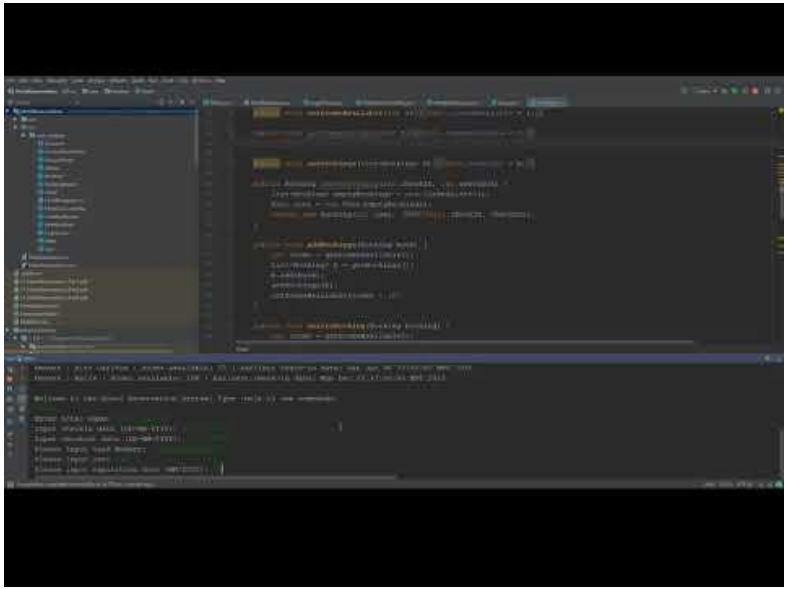
ID	Requirement	Actor
UR-001	User should be able to book a reservation	SearchUser
UR-002	User should be able to cancel a reservation	BookedUser
UR-003	User should be able to pay with credit card only	SearchUser, BookedUser

Design Pattern - **Iterators++**



- *“Provide a way to access the elements of an aggregate object sequentially without exposing its underlying representation.”* - Gang of Four
- We use `HotelListIterator` and `BookingIterator` to sift through `Hotel` and `Booking` lists
- Example: Finding hotel locations based on query
  - Old method: Create a discrete list of all hotel locations separate from hotel objects, go through that list (again) to find matches.
    - **2x list iteration**
  - New method: Iterate through hotel objects, get their location, and add to results list if it matches.
    - **1x list iteration** - results found in iteration process

Demo



https://youtu.be/rLhPdtQANNk

Unimplemented / Future Ideas

- **User Authentication / Administrators**
  - We would like to implement our original idea of having an administrator verify credentials, then be able to add/remove bookings from the actual database.
- **Searching/Sorting By Price Range, and Amenities**
  - In the future, we would like to add a way for the user to sort hotels by price, based on either location or name, as well as displaying amenities from the hotel.