UDACITY

Return to Classroom

DISCUSS ON STUDENT HUB

Hotel Reservation Application

REVIEW

CODE REVIEW 3

HISTORY

Requires Changes

2 specifications require changes

Greetings Student,
You did a good job on this submission. Your `Room` class implements the `IRoom` interface properly. Also, your `FreeRoom` class extends the `Room` class. And, Both of your `Customer` and `Reservation` classes apply encapsulation. However, there should be at least one example of the model classes overriding the equals and hashcode methods. Please, take the time and address the changes required.

Pro Tips

Let me share with you some good documents that talk about the interesting points of this project.

- Controlling Access to Members of a Class
- Encapsulation in Java
- final keyword in java

Object-Oriented Programming

The hotel reservation application contains the IRoom interface , which is implemented by the `Room` class.

Your `Room` class implements the `IRoom` interface properly.

The `FreeRoom` class extends the `Room` class.

A `FreeRoom` is a special type of room that inherits the `Room` class. In this way, your code is not redundant and is reusable. Nice work!

There is at least one example of the model classes (`Room` , `Customer` , Reservation `) using data encapsulation.

Both of your `Customer` and `Reservation` classes apply encapsulation. This is good code practice here!

Suggestions

Check this link for more on encapsulation and accessibility.

There is at least one example of the model classes (`Room` , `Customer` , `Reservation`) overriding the `toString` method.

All of your model classes successfully override the `toString()` method and you provide clear and descriptive messages to the user.

There is at least one example of the model classes (`Room` , `Customer` , `Reservation`) overriding the `equals` and `hashcode` methods.

There should be at least one example of the model classes (Room, Customer, Reservation) overriding the `equals` and `hashcode` methods. It's also recommended to override them in the `Reservation` class as well since it could help us to avoid storing duplicate reservations.

Suggestions

Check out this post that explains the need for overriding these methods and provides.

- Why do I need to override the equals and hashCode methods in Java?

The application contains at least one example of using each of the following access modifiers: 'public', 'private' and 'final'.

Processing and Storing Data

Collections are used to store data for:

- Room
- Customer
- Reservation

The collection type chosen for rooms ensures that two rooms cannot be booked at the same time.

Please, make sure that the `equals` and `hashcode` methods are overridden in the Reservation model before getting a feedback here.

All of the service classes use `static` references to create singleton objects.

The `ReservationService` contains `for` loops that are used to iterate over and process data. That's neat!

The `ReservationService` contains `for` or `while` loops that are used to iterate over and process data in order to do the following:

- Search for available rooms
- Search for recommended rooms

Well done using static references to create singletons in your application.

Suggestions

Check out this link for more details about singletons.

The `ReservationService` contains at least one example of using each of the following method access modifiers:

- `public`
- `private`
- `default`

Your `ReservationService` class used public, private access modifiers.

Suggestions

More reading on default methods could be found here.

Core Java Concepts

The `Customer` class should contain at least one example of validating a String to ensure that it has valid email address syntax.

You are validating the customer email correctly. Good job!

The application contains the enumeration class `RoomType`.

Excellent work! The application contains the enumeration class `RoomType`. Keep it up!

The Reservation class uses Date objects for check-in date and check-out date.

Well done, the `Reservation` class uses Date objects for check-in date and check-out date.

The application contains at least one example of using `Exceptions` to validate input and `try` and `catch` blocks to handle error flow without crashing the application.

You have used `try` and `catch` blocks correctly. That means you showcased your understanding of throwing and catching exceptions.

The application uses different Java types (String, Double and Dates) to store data on objects.

Your application uses different Java types to store data on objects.

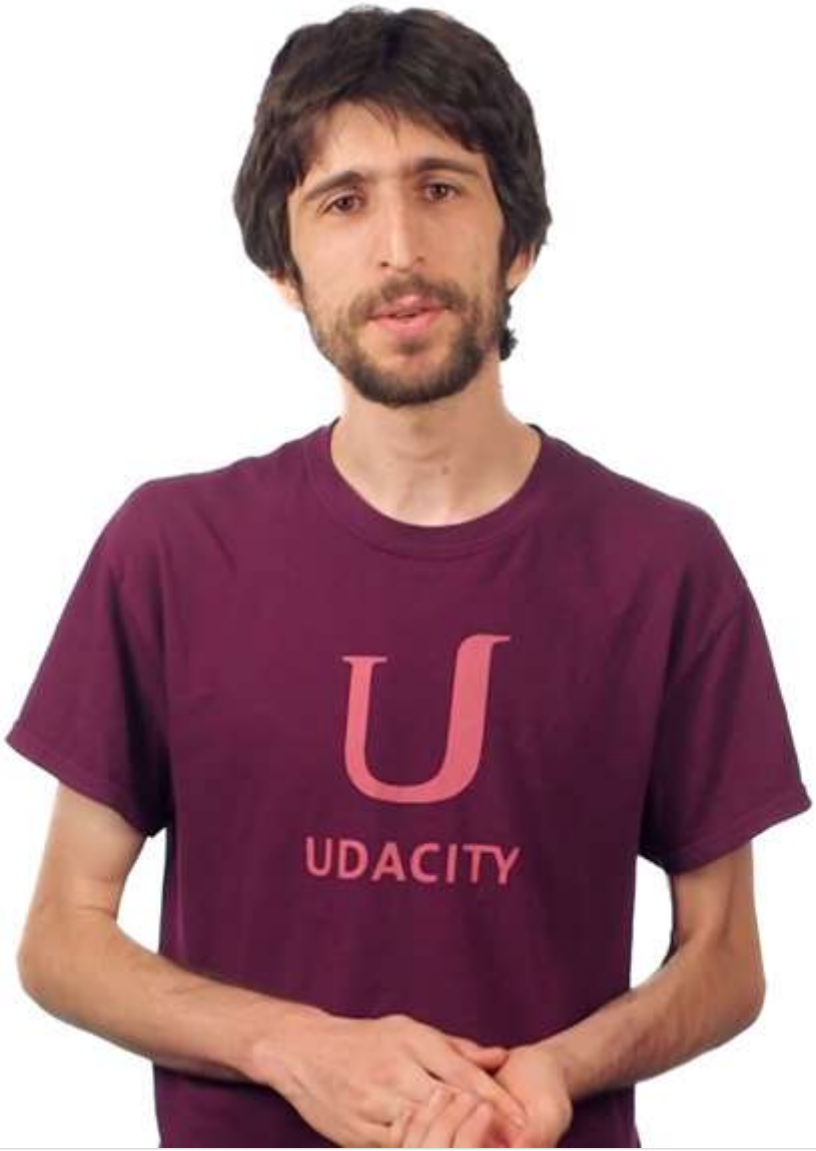
The application UI uses a `switch` statement to handle the user input flow.

Great work using a `switch` statement to handle the user input using a switch instead of using multiple if statements.

RESUBMIT

DOWNLOAD PROJECT

3 CODE REVIEW COMMENTS ›


U
UDACITY

Best practices for your project resubmission

Ben shares 5 helpful tips to get you through revising and resubmitting your project.

Watch Video (3:01)

RETURN TO PATH

Rate this review

START