Ada-C8 / hotel
forked from AdaGold/hotel

No description, website, or topics provided.

19 commits | 3 branches | 0 releases | 2 contributors

Branch: master | New pull request | Create new file | Upload files | Find file | Clone or download

This branch is even with AdaGold:master. Pull request Compare

droberts-ada committed on GitHub Merge pull request #4 from AdaGold/dpr/reqs2 … Latest commit 391b9d8 a day ago

.github	copy-pasta PR template and feedback form from Ride Share	2 days ago
lib	add more details and three waves. also lib and spec folders.	6 days ago
specs	add more details and three waves. also lib and spec folders.	6 days ago
.gitignore	Add standard ruby gitignore file	5 days ago
README.md	Respond to PR comments	a day ago
feedback.md	Respond to PR comments	a day ago

README.md

Hotel

Learning Goals

Reinforce and practice all of the Ruby and programming concepts we've covered in class so far:

- Design a system using object-oriented principles
- Create and instantiate classes with attributes
- Create class and instance methods within our classes
- Write pseudocode and create tests to drive the creation of our code

This is a stage 3, individual project.

Introduction

Your company has been contracted to build a booking system for a small hotel. This system will be used by employees working at the front desk, and will not be available to the general public.

This system will have two parts: a user interface that runs in the terminal, and a module full of business logic, classes and methods that keep track of which rooms are reserved when. **Your job is to implement the business logic only.** You do **not** need to build a CLI for this project.

Instead, you will use tests to verify your part of the system works as intended.

Expectations

This project is both a culmination of our Intro to Ruby unit and our first stage 3 project. This means the requirement are more open-ended and ambiguous than previous projects you have worked on. This is intentional. You will be expected to:

- Make decisions on how to structure your classes and methods
- Ask questions when you need clarification
- Understand that the way you implement something may be different than the way your neighbor implements it

It is possible you will not be able to complete all requirements. Though all 3 waves will be visible at the beginning of the project, they are organized by difficulty and relevance to the learning goals, and should be tackled in order.

Setup

We will use the same project structure we used for the previous project. Classes should be in files in the `lib` folder, and tests should be in files in the `specs` folder. You should utilize a spec helper file. You will run tests by executing the `rake` command, as configured in a Rakefile.

1. Fork this repository in GitHub
2. Clone the repository to your computer
3. Create/copy a rakefile to run your tests
4. Create the `specs/spec_helper.rb` file to load your classes and start `simplecov`. This file will load all the required gems and source files your spec files need so they only need to require the helper.
   - Each of your spec files should `require_relative` the spec helper file.
5. Create a test to check the instantiation of one of your object types (**RED**)
6. Create the class for the object tested in the step above (**GREEN**)
7. Use `git add`, `commit` and `push` commands to push your initial code to GitHub

Process

You should use the following process as much as possible:

1. Write pseudocode
2. Write test(s)
3. Write code
4. Commit

You should have **95% code coverage** using `simplecov`.

Your git commit history should provide a clear description of how your code developed, letting the reader know what changed when and why.

Wave Zero: Project Design

This will be an in-class activity.

Wave One: Tracking Reservations

Remember that your job is only to build the classes that store information and handle business logic, and the tests to verify they're behaving as expected. Building a user interface is not part of this project!

User Stories

- As an administrator, I can access the list of all of the rooms in the hotel
- As an administrator, I can reserve a room for a given date range
- As an administrator, I can access the list of reservations for a specific date
- As an administrator, I can get the total cost for a given reservation

Constraints

- The hotel has 20 rooms, and they are numbered 1 through 20
- Every room is identical, and a room always costs $200/night
- The last day of a reservation is the checkout day, so the guest should not be charged for that night
- For this wave, any room can be reserved at any time, and you don't need to check whether reservations conflict with each other (this will come in wave 2!)

Error Handling

- Your code should raise an error when an invalid date range is provided

Hints

- You might want to investigate Ruby's `Date` gem.

Wave Two: Room Availability

User Stories

- As an administrator, I can view a list of rooms that are not reserved for a given date range
- As an administrator, I can reserve an available room for a given date range

Constraints

- A reservation is allowed start on the same day that another reservation for the same room ends

Error Handling

- Your code should raise an exception when asked to reserve a room that is not available

Wave Three: Blocks of Rooms

If you are not familiar with what a block of hotel rooms, here is a brief description:

> A Block Booking refers to a group of rooms set aside for a specific group of customers for a set period of time.
>
> Room blocks are commonly created for large events like weddings or conventions. A number of rooms are set aside, and are made available for reservation by certain customers at a discounted rate. These rooms are not available to be reserved by the general public.

User Stories

- As an administrator, I can create a block of rooms
  - To create a block you need a date range, collection of rooms and a discounted room rate
  - The collection of rooms should only include rooms that are available for the given date range
  - If a room is set aside in a block, it is not available for reservation by the general public, nor can it be included in another block
- As an administrator, I can check whether a given block has any rooms available
- As an administrator, I can reserve a room from within a block of rooms

Constraints

- A block can contain a maximum of 5 rooms
- When a room is reserved from a block of rooms, the reservation dates will always match the date range of the block
- All of the availability checking logic from Wave 2 should now respect room blocks as well as individual reservations

Optional Enhancements

You should not be working on these (or even thinking about them) until you have fully completed wave 3.

- Allow a user to set different rates for different rooms
- Read/write CSV files for each piece of data that your system is storing
- Create a CLI to interact with your hotel system