# Case study

## Setting

This case study puts you in the position of a Java backend developer at trivago. There has been a decision at the management level to re-implement the backend from scratch and see whether the new solution can replace the legacy system which has grown over the past several years.

We suggest to spend anywhere between 2 hours to max a day on the case study.

## Your task

First take a look at the interface `HotelSearchEngine`. Your task will be to implement the two given functions in the concrete implementation in `HotelSearchEngineImpl`:

- `initialize`: this function is used to read the .csv (comma separated value) files initially. Take a look at the database schema section for further details.
- `performSearch`: executes the actual search. You are given a concrete search and can use the `OfferProvider` interface to retrieve offers from the advertisers.

The `initialize` function will be called one time during startup, whereas the `performSearch` function will be called multiple times. Make sure to do the initializations in the former function and execute the latter as quickly as possible. Also, every call to `OfferProvider.getOffersFromAdvertiser` retrieves offers from advertisers via a network request. It is very costly, so don't call it unnecessarily!

## Database schema

The data for your search engine is stored in csv files:

- `advertisers.csv`: contains an `id` and the name of the advertiser
- `cities.csv`: contains ten cities with a name and an `id`
- `hotel_advertiser.csv`: models the m-n relationship between hotels and advertisers; every advertiser might have offers for m hotels and each hotel might be offered by n advertisers. Contains two columns, referencing the ids of `advertisers` and `hotels`
- `hotels`: the given hotels for your search engine. Contains an `id` field and the foreign key to the `cities` table. Next, you have five columns that further describe the hotel: the number of clicks and impressions over the last week, its name, the user rating (0 - 100), and the hotel star rating (1 - 5).

## Important Notes

### Scope

- Focus on implementing a minimum viable product, that is, implement the class `HotelSearchEngineImpl` in a way that correct offers and hotels are returned for a given search query.
- We respect your time, and expect the whole case study to only take about 2 hours. Don't spend your whole weekend trying to impress us with anything fancy.
- Add comments in plain text or pseudocode where you can imagine improvements or have some general ideas.

### Hints

- You can use the given unit tests, but beware that they are not exhaustive (feel free to add more).
- If necessary, you can extend all classes with additional methods, but you should not change the method signatures in `HotelSearchEngine`.

## Building

We have provided a Gradle build file as well as Eclipse and Idea project files for convenience, but we use Gradle internally. Here are some useful gradle commands for this project:

- `gradle` / `gradle build`: compile and test
- `gradle clean`: delete all of the build artifacts
- `gradle classes` / `gradle testClasses`: compile source/test Java classes
- `gradle test`: run unit tests

## Evaluation criteria

We are looking at the following points when you are completing the case study:

- Pragmatic use of data structures/collections
- Code legibility
- Code reuse
- Thoughtfull use of concurrency
- If you believe that things are out of scope, that is totally fine. Please add TODO comments on things that should be improved in the future