

# Selectra Technical Challenge

Welcome to Selectra's technical tests suite. In this game, you will be building a simple billing service.

## Instructions

- Clone this repository
- Solve as many levels as you can, and in ascending order
- Commit your code at the end of each level
- Give `part2` a try, even if you have not fully finished `part1`
- Email [rh+tech@selectra.info](mailto:rh+tech@selectra.info) when you are done, with a link to your GitHub repository!

Levels become more complex over time, so you will probably have to re-use some code and adapt it to the new requirements.

## Requirements

- Pick either **PHP** or **Ruby**
- Clean, readable and robust code
- Follow Software Development best practices

## Part 1

### Level 1

In this first level, you will be writing the formula to calculate the yearly electricity bill for our customers.

#### Instructions

- `users` contains the yearly consumption (in kWh) and a reference to the current provider
- `providers` is the list of providers, with related kWh price fee

Write code that generates `output.json` from `data.json`

### Level 2

Sometimes, contracts have special discounts, depending on their length.

#### Instructions

In this level we have added a new entity, `Contract`. We are making good progress on our product and every user can have multiple contracts. On the other side, a contract is associated with a provider.

The discount rules are as follows (`contract_length` is a positive non-zero integer): - if `contract_length` is less or equal than 1 year -> 10% discount - if `contract_length` is greater than 1 and less or equal than 3 -> 20% discount - if `contract_length` is greater than 3 -> 25% discount

Write code that generates `output.json` from `data.json`

## Level 3

Providers want to get paid! And... providers pay us too.

### Instructions

From the billed amount, the insurance applies a 0.05€/day commission. The remaining amount goes directly to the provider, which is then responsible to give us the 12.5%.

Write code that generates `output.json` from `data.json`

## Level 4

Green offers are on their way, hurray!

### Instructions

We want to give contracts the option to be green. In other words, for every kWh produced, your bill is reduced by, let's say, 0.05€

Write code that generates `output.json` from `data.json`

## Level 5

Customers are always on the lookout for the best deals. So let's meet their expectations.

### Instructions

First of all, we have decided to replace `contract_length` with `start_date` and `end_date`

A customer can then change its contract's provider, and here are the steps: - the yearly fee must be updated accordingly - the cancellation fee (50€) is immediately applied to the `provider_fee`

To make things easier, you can assume the difference between `start_date` and `end_date` is always in terms of years (day and month are the same).

Write code that generates `output.json` from `data.json`

## Level 6

During the year, providers apply additional season costs.

## Instructions

In this level, you are going to essentially override the `yearly_consumption` value provided as input by `data.json`. The steps are as follows: - Read the value from the input file - Calculate the values for each season of which it consists of. For example: if the contract starts in August 2017 and ends in August 2019, you will need to compute each season from August 2017 until Spring 2019 - Sum up the season values to obtain the new overall consumption - This will be the new starting value from which all other operations will occur (i.e. discount, cancellation fee etc...)

Season consumptions are calculated as follows: - Spring: +1% - Summer: -1.5% - Autumn: average of +0.7% with average of previous spring and summer. If spring and/or summer are not present, only consider the +0.7% result - Winter: no additional cost

Finally, you can make the following assumptions: - Spring: 1st March - 31st May - Summer: 1st June - 31st August - Fall: 1st September - 30th November - Winter: 1st of December - 28th February

Write code that generates `output.json` from `data.json`

## Part 2

Congrats! You have reached the end of the test! :) In this final part, we are going to test your creativity.

## Instructions

Create a CLI tool to interact with what you have built so far. Some ideas could be: - Feed your application with data (a JSON file) - An user-friendly print of the output - Export output in different formats

Keep in mind that we are looking for clean, readable and robust code.