## Task

In this exercise you should implement a Spring-Boot application that provides an API for a checkout system in a supermarket. It should offer its clients the possibility to register an arbitrary number of items in a shopping cart and calculate the cart's total price in the end.

The goods in a normal supermarket are identified by using Stock Keeping Units (SKU). For that, we will use individual letters of the alphabet (e.g. A, B, C) and the goods are priced individually. In addition to that some items get quantity-based discounts. For example: item A costs 40 Cent individually, but this week there is a special offer: Buy three A's and they will cost you 1 Euro.
The checkout should accept items in any order, so that if we scan a B, an A and another B, it will recognize the two B's and apply the according pricing rule automatically.

These are the pricing rules:

1. Item A costs 40 - Special Offer: 3 items of A cost 100
2. Item B costs 50 - Special Offer: 2 items of B cost 80
3. Item C costs 25
4. Item D costs 20

Because the pricing can change frequently, we need to be able to pass in a set of pricing rules each time we start handling a checkout transaction. So the main steps when using the API are something like the following:

```
1. create a checkout with pricing rules
2. scan an arbitrary number of items
3. calculate the total price
```

On https://start.spring.io/ a simple Spring project can be set up with the dependency "Spring Web". After downloading and unzipping this can be used as a starting point.
Please include a test set which you think is suitable for verifying the correctness of your code.

## General Conditions

In this exercise we're not necessarily looking for speed, completion or even the solution to a particularly difficult problem. It is more important for us to see how you get to the solution and that you are able to present it to us. The programming language must be Java. You can make use of a build tool (e.g. Maven) but don't have to. Architectural details that are not mentioned here can be freely decided by you. Keep in mind that you should be able to explain them, if needed.
You can spend as much time on this task as you like. However, we don't expect you to invest more than 4-6 hours. If you are not able to finish everything within this timeframe, please add a readme.md file that describes, what you would tell us about the current status of your work in a daily stand-up meeting. We'd love to know what parts you consider finished, what would be your next steps and which of the things you wrote already you'd still like to improve if there was more time.