1 General information

This documentation is specifically about the open-source program made for packing "Boxes" in "Containers". These terms will be used in the whole program. Keep in mind, that the boxes are generated from columns and security margins are included in the returned dimensions.

1.1 Program structure

The project exists of 3 parts. The back-end, rest service and web visualization. We chose to write the program in Java so we could use OO-approaches. It is suggested that you use Java version 8+, which can be configured in the maven "pom.xml" file.

1.1.1 The back-end

The back-end exists of 2 important algorithms and a "Facade" class which interacts with both algorithms. The first algorithm places the boxes into containers. After that, the result is returned and used by the second algorithm to calculate the best amount of different containers.

The back-end is managed through maven. You can see that dependencies are imported in the 'pom.xml' file. The back-end will also be used as a dependency for another project (using maven). This makes the program extensible. Technical limitations, opportunities, etc.. will be mentioned in the "Technical Specifications" document.

To compile the back-end, keep in mind that you first have to "install" the back-end with maven (to put it in the local maven repository) and then it can be imported in your rest-service. Otherwise the rest service will look online for the dependency, which won't work.

1.1.2 The rest-service

This "layer" in-between connects the back-end with our front-end. Communication is entirely in JSON format. The front-end (which can be a website, application, desktop program, Excel, ...) sends a request to the rest-service, which will respond in JSON format. This allows the front-end and back-end to be developed at the same time, and even in different languages.

As mentioned above, this rest-service implements the back-end through a maven dependency specified in the "pom.xml" file. This allows the possibility to implement other projects or calculations and they can be linked together in this layer.

As example, you can calculate the containers necessary (back-end A) and when you press a button on the front-end, it will automatically calculate the transport prices(back-end B).

1.1.3 The front-end

The front-end has 2 responsibilities. One is to send HTTP Requests to the rest-service, regardless of the back-end, it will get a JSON response. The second responsibility is to display the JSON content.

Important calculations to display the information is all done in the "printer.js" file. the JavaScript file will display the placement method for each container, volume utilization and container dimensions.