Upon reviewing the requirements, I realized that in order to determine the best way to fit items into the package, I needed to find all possible combinations and then calculate the weight and cost of each item. To achieve this, I began by creating a validation method to ensure that the input file parameter existed and that I could read from it. Next, I created a validation method to parse each line item, ensuring that I could work with the data and throwing an APIException if any issues were encountered.

To achieve this, I implemented a function that removes any whitespace characters from the string and splits it using the chars ")(", creating a list of string line items. Then, I ran it through another method to ensure that "(" and ")" were trimmed from the values, leaving only "index, weight, cost". After parsing all the line items and ensuring that they met the constraints, I saved them as a list of packages.

Once the validation was complete, I began creating a 2D List containing a list of package items. This allowed me to easily find and calculate what I needed to do later on and remove any items that met any constraints. I used an iterative algorithm to populate this list instead of a recursive algorithm.

Next, I ran the list through another iterative algorithm to find the best combination of weight and cost and saved the current weight, cost, and combination. If I found another combination with the same cost, I checked if the weight was less, and if it was, I saved it as the new best combination, weight, and cost. Once the iteration was complete, I returned the best combination.

Finally, I checked if a best combination was found or not. If one was found, I joined the index values of the list using a comma to create a string that would read as "5,6". When a combination was not found, the string would read as "-".

Although I did not employ TDD, I incorporated test cases into my coding process to verify the correctness of the initial code. These test cases included both positive and negative scenarios, and were implemented to ensure that any refactoring I performed did not impact the logic that was already functioning properly.