Programming Exercise: Pick and Ship

Description:

You are a software developer working for an online retailer. Your boss, Tom, has recently done an internal audit, and discovered that he is spending much too much money on shipping costs. He has decided that he would like you to make a new pick/ship program to minimize the number of boxes used for shipping orders.

Tom has exported the interesting components of the inventory database to a simple text file. The format of the inventory file is as follows:

```
INVENTORY START
ITEM START
CODE: <String>
NAME: <String>
WEIGHT: <double>
ITEM END
. . . .
ITEM START
CODE: <String>
NAME: <String>
WEIGHT: <double>
ITEM END
INVENTORY END
```

Tom has contracted with an independent web developer named Sandy to modify the website. Sandy has configured the website, so that each order will be placed in a simple text file. At the end of the day, Tom will run your program against each order file, which will generate a pick/ship list for each order.

The format of each order file is as follows:

```
ORDER START
ORDER NUMBER: <Integer>
CUSTOMER CODE: <String>
ITEM: <code>, <qty>
ITEM: <code>, <qty>
ITEM: <code>, <qty>
…
ITEM: <code>, <qty>
ORDER END
```

As mentioned earlier, shipping costs are expensive, and Tom wants to minimize the number of

boxes used to ship an order. He only ships with one kind of box, and it holds a maximum of 10 lbs. He wants your program to be “very good” at filling each box shipped to capacity. For example, if the order has an estimated shipping weight of 17 lbs, and it can be shipped in 2 boxes, don’t ship it in 3.

Here is the file format for the Pick/Ship list:

```
PICK SHIP START
ORDER NUMBER: <Integer>
TOTAL SHIP WEIGHT: <double>
BOX START: <Integer>  /* indicates the box number for multiple box
shipments */
SHIP WEIGHT: <double>
ITEM: <Code>, <Qty>
ITEM: <Code>, <Qty>
. . .
ITEM: <Code>, <Qty>
BOX END
 . . .
[ /* Optionally include more box records if required */ ]
 . . .
PICK SHIP END
```

Implementation Details

Your program should be written in Java as a console application. It can be compiled using Ant, Maven, or Eclipse, but it must compile.

Please provide the relevant source files, and build instructions.

Your solution will be evaluated with respect to many qualities: code style, implementation, object oriented design, etc.

The main program entry point should accept as command-line arguments: the name of the inventory file, and the name of the order file. For example:

```
java MyProgram <inventory-file> <order-file>
```

After parsing the file, and processing the order, the pick/ship list should be generated. You can either print to the screen, or output to a file. This is up to you.

You will be provided a sample inventory file and some order files.

For extra credit: describe your thought process around your implementation of this problem; why did you choose the algorithm you implemented. What problems did you encounter? How did you fix them?