Shopping Cart

Please make sure you read this document in its entirety.

Overview

Please write code delivering the requirements of the steps that follow. The requirements don't mention a command line, web or any other type of application, and we are not looking for one. Do write code that you would be happy delivering to a paying client, keeping in mind the simplicity comments below.

You should not find this test to be particularly difficult. It is designed to be a straightforward coding exercise, and it should take you no more than 90 minutes.

The problem

As you work through the steps, you'll create code to allow a user to add products to a shopping cart, calculate the total price and then the sales tax for the items contained in the cart. As mentioned previously, we are not looking for a command line or a web application, so please just use your tests to drive the code (e.g. class libraries).

Please Note: All totals should be rounded up to 2 decimal places, i.e. 0.565 should result in 0.57 but 0.5649 should result in 0.56. This is baisically the same rounding you learned at school, but you can follow this link if you really want more details.

Step 1: Add products to the shopping cart.

Given:

- An empty shopping cart
- And a product, *Dove Soap* with a unit price of *39.99*

When:

- The user adds 5 *Dove Soaps* to the shopping cart

Then:

- The shopping cart should contain *5* Dove Soaps each with a unit price of *39.99*
- And the shopping cart's total price should equal *199.95*

Archive this as a *separate* .zip file named *step1.zip* then continue on to step two.

Step 2: Add additional products of the same type to the shopping cart.

Given:

- An empty shopping cart
- And a product, *Dove Soap* with a unit price of *39.99*

When:

- The user adds 5 *Dove Soaps* to the shopping cart
- And then adds another 3 *Dove Soaps* to the shopping cart

Then:

- The shopping cart should contain 8 Dove Soaps each with a unit price of *39.99*
- And the shopping cart's total price should equal *319.92*

Archive this as a *separate* .zip file named *step2.zip* then continue on to step three.

Step 3: Calculate the tax rate of the shopping cart with multiple items

Given:

- An empty shopping cart
- And a product, *Dove Soap* with a unit price of *39.99*
- And another product, *Axe Deo* with a unit price of *99.99*
- And a sales tax rate of 12.5%

When:

- The user adds 2 *Dove Soaps* to the shopping cart
- And then adds 2 *Axe Deos* to the shopping cart

Then:

- The shopping cart should contain 2 Dove Soaps each with a unit price of *39.99*
- And the shopping cart should contain 2 Axe Deos each with a unit price of *99.99*
- And the total sales tax amount for the shopping cart should equal *35.00*
- And the shopping cart's total price should equal *314.96*

Archive this as a *separate* zip file named *step3.zip* file and send all of the zip files as email attachments to your recruitment contact, following the guidance set out below.

Please include the version number `72cf4fe47f85c39779267d0ecee07655a354e623` in a README file with your submission so we know what version of the instructions you've been given.

Note: Please *don't* include the compiled code in the zip files. We won't use it, and it causes trouble with many email systems.

Please do not submit Dropbox or Google Drive links. They don't meet the requirements of our anonymous submission policy.

Note for Gradle Users: Normal best practice would be to include the gradle wrapper in your build script. However, this causes the project to include .jar and .bat files, which can cause trouble with a lot of email systems. You don't need to include a gradle wrapper with your project. The reviewer will have the latest version of gradle installed on their system if gradle is used.

Anonymous submission

We practice a blind review process for all submissions, so please don't include your name (or the name of your limited company) anywhere in your submitted source code, documentation, comments, or zip files. This makes our interview process as fair and systematic as possible.

The person who reviews your submission won't have access to your CV or know anything about you, including your name.

Obviously, do include your name in the email you send, so we know who submitted the code! This won't be shared with the reviewer.

What we are looking for:

Test Coverage: The solution should be developed "test-first", and should have excellent unit tests and test coverage. It will be hard to pass this exercise if we can't reasonably believe the solution was developed "test-first" following TDD.

Build file: Please provide an automated build file that compiles your code and runs the tests. For java submissions, a Gradle or Maven build file is ideal. Submissions without an automated build will not be accepted.

Simplicity: We value simplicity as an architectural virtue and a development practice. Solutions should reflect the difficulty of the assigned task, and should not be overly complex. Layers of abstraction, patterns, or architectural features that aren't called for should not be included.

Self-explanatory code: The solution you produce must speak for itself. Multiple paragraphs explaining the solution are a sign that it isn't straightforward enough to understand purely by reading code, and are not appropriate.

Deliverable

Include a readme: Please include a readme file in the root of the project which states any requirements to run the code, and commands needed to run the code. Assume the reviewer is unfamiliar with your choice of framework and/or build tools and maybe using a different IDE.

This version number: As mentioned above, please include the version number `72cf4fe47f85c39779267d0ecee07655a354e623` in a README file with your submission so we know what version of the instructions you've been given.