shopping-cart-test-java-main/README.md

Instructions for the assignment

If cloning the repo
1. Clone this repository on your machine.
2. Use your IDE of choice to complete the assignment.
3. When you are done with the solution and have pushed it to the repo, you can notify us that it was completed.

If working with an archive
1. Use your IDE of choice to complete the assignment.
2. Clean the repo of generated files and send back an archive with the solution.

Assignment Details

We will be using a simple shopping cart, similar to what you would see on any e-commerce website, as the domain for this problem.

We respect your time and understand that asking you to give up an hour or so of your personal time for the interview process is a big ask. However, we want to make the process as simple and stress-free as possible, by allowing you to complete the first stage of the process in the comfort of your own home. We will also be using your submission in further stages of the process should you be successful.

What we are looking for

Test Coverage: The solution should be developed ?test-first?, should have good unit tests, and common paths should be covered. Your tests should also be self-contained and not rely on external systems to be available to run.

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.

An understanding of functional programming: The solution should demonstrate the use of functional programming concepts, in particular, the use of immutable data structures, referential transparency, and the use of monads in dealing with effects.

The Problem Statement

Create a shopping cart package (e.g. com.siriusxm.example.cart) that facilitates 2 basic capabilities:

1. Add multiple products to the cart. A product has a name and price. You should be able

to specify how many of each product is being added to the cart and provide a means to observe the resulting state.
2. Calculate the totals
   i. Cart subtotal (sum of price for all items)
   ii. Tax payable, charged at 12.5% on the subtotal
   iii. Total payable (subtotal + tax)

Pricing data for each product should be retrieved via an HTTP call. You can find example pricing data for a set of sample products at the URI?s below. You should assume that the product name (lowercase) matches the file name. Use whatever libraries you like to get and parse the JSON.

Valid Product Information URLs

-
https://raw.githubusercontent.com/mattjanks16/shopping-cart-test-data/main/cheerios.json
-
https://raw.githubusercontent.com/mattjanks16/shopping-cart-test-data/main/cornflakes.json
-
https://raw.githubusercontent.com/mattjanks16/shopping-cart-test-data/main/frosties.json
-
https://raw.githubusercontent.com/mattjanks16/shopping-cart-test-data/main/shreddies.json
-
https://raw.githubusercontent.com/mattjanks16/shopping-cart-test-data/main/weetabix.json

Sample based on the data

The below is a sample with the correct values you can use to confirm your calculations

`Add 2 × cornflakes @ 2.52 each`

`Add 1 × weetabix @ 9.98 each`

`Subtotal = 15.02`

`Tax = 1.88`

`Total = 16.90`

FAQ

What other functionality is required?

No other capabilities are required. This is not a trick question and there is no single correct answer. We prefer simple, well tested solutions over clever solutions. The complexity of your solution should reflect that of the problem. Use whatever external libraries or packages you wish.

Should I include an app?

Please do not write a web, desktop, command line or any other kind of app. Your code needs only to be driven by tests and can extend IOApp.

Must I use Gradle + Springboot project?

We often use Gradle + Springboot in our Java projects, so we've provided all the boilerplate and dependencies in this template project. However, feel free to use a functional effect system that you are the most comfortable with.

What about ambiguity?

If there is any ambiguity please add this in a section added to the bottom of the README and make a choice yourself to resolve the ambiguity.

How should rounding be handled?

Prices should be rounded up where required.

Is there a time limit for completing the exercise?

There is no time limit for how long the exercise should take for you to complete, but we find most people complete the exercise in around 90 mins.

Internal Reference

Please do not remove this section, it helps us a lot to have this information available.

Template: java-gradle

shopping-cart-test-java-main/build.gradle

```
plugins {
    id 'java'
    id 'org.springframework.boot' version '3.0.2'
    id 'io.spring.dependency-management' version '1.1.0'
}

group = 'com.example'
version = '0.1.0-SNAPSHOT'

java {
    toolchain {
        languageVersion = JavaLanguageVersion.of(21)
    }
}

repositories {
    mavenCentral()
}

dependencies {
    implementation 'org.springframework.boot:spring-boot-starter'
    testImplementation 'org.springframework.boot:spring-boot-starter-test'
}

test {
    useJUnitPlatform()
}
```

shopping-cart-test-java-main/settings.gradle

```
rootProject.name = 'shopping-cart-test-java'
```

shopping-cart-test-java-main/src/main/java/com/shoppingCart/Application.java

```
package com.shoppingCart;

public class Application {
    public static void main(String[] args) {
        System.out.println("Good luck on the take home!");
    }
}
```

shopping-cart-test-java-main/src/main/resources/application.properties