RECEIPT USER DOCUMENTATION

SUMMARY

TECHNICAL OVERVIEW

Language: ***Java***
Test Suite: ***jUnit***
Version: ***v1.1.0***

Core Assets: ***Input .txt files***

CHALLENGE OVERVIEW

Requesting Company: ***ThoughtWorks***
Applying Position: ***Associate Consultant Program - North America***
Challenge: ***#2 Sales Taxes*** *(See README for challenge requirements)*
Date: ***Jan 2017***
Estimated Completion Time: ***N/A***

INSTRUCTIONS FOR INSTALL & USE

Installation in IntelliJ

1. Import project into IntelliJ
2. Run *Receipt* class to activate the program.
3. Console will display calculated output.
4. Optional filenames can be provided by going to Run>File Configurations then typing: `"PATH/TO/Filename.txt"` in program arguments. It will take more than filename string at a time.

Testing Suite Available under Test directory

1. Run *ReceiptTest* class for challenge testing

TESTING OPTIONS

Project tests and basic challenge success tests have been provided.

They can be found in the `test` directory.

ASSUMPTIONS

Below are notes & thoughts based on challenge expectations.

File Input
A series of filenames can be imported at start of program.
If no filename is provided, 3 .txt files will be loaded.
The challenge samples will need to be provided as a .txt file, and executed, if no filename has been given.
There will need to be a line pattern or syntax guide involved with all input files. This will ensure proper handleing of data transcoding.
A file reader will need to be implemented at some point.

Transcoding
Each line item will contain 3 crucial pieces of information - Quantity. Description. Price.
The numerical string data will needed to be converted to an integer for calculations.
Regex could be used to pull apart the information.

Objects & Classes

Receipt - Controller class
Receipt Calculator - Class tasked to calculate sales tax and totals
Receipt Scanner - Builds Purchases and Items based on data from called upon file reader
Purchase - Class to hold multiple Items. Also could oversee totals?
Item - Class to deal with item information like description, quantity, what kind of tax bracket, and price.
Receipt Console - View class in charged of providing output to console.
~~*Receipt Validation* - Verifies data from file is valid~~

Other
Data output will be via the console.
Testing Suite will need to be built for challenge success.
Quantity multiplier was not discussed in challenge, but logic for it should be included.

USE CASES

As a Client/User...
...I want the ability to provide new purchase data via .txt filenames.
...I want to see the sales tax total.
...I want to see the updated item total.
...I want to see the purchase total.

...I want the item details arranged properly for output.
...I want the item purchase amount to be updated in the output.
...I want the output to be shown in the terminal’s console.
...I want sales tax to be charged at a base rate of 10%.
...I want an **EXEMPT** category of items with 0% sales tax.
...I want an **IMPORT** sales tax calculated at an additonal 5% on all goods.
...I want the sales tax calculation to round to the nearest .05.

As a Developer...
...I will use .git version control.
...I will provide 3 sample carts as a .txt files.
...I will allow new filenames to be provided with cart info - specified in these instructions.
...I will create models separating functionality for future improvements as well as to keep with OOD and RESTful practices.
...I will provide two sets of tests (method testing & challenge goals).

Notes, Thoughts, & Challenge Feedback

This was my first java project, so I had to quickly bring myself up to speed with Java syntax, IntelliJ, and jUnit. Reading, following tutorials, and tracking down errors were a big part of this challenge. Once I got

going in the coding aspect - it came together pretty fast.

The most difficult part of this challenge was wrapping my head around IntelliJ. Once I did a few sample tutorials regarding project set-ups, the building queues came very naturally. I didn’t do too much in the ways of test driven development on this project, which is no normal for me. I was trying to wrap my head around java in general, so in an effort to save some time I decided to do testing after. Lame - I know, I know.

There were three major critical thinking moments I had to do for this project. I had to figure out how to transcode each line item properly. I had to figure out how to determine the sales tax price. For that I think I came to a good solution for now, but it’s based on key words like ‘chocolate’. In a perfect world you’d have inventory models to access details. Lastly, I had to dig back into my code at the end because I realized I didn’t take in account quantity numbers - which if you purchase say “2 Batman Forever CDs” - then it would have only computed the price for one.

As for challenge feedback, I found the challenge pretty easy going. The wording of the challenge was a bit confusing at first, but as I started nailing down the parts I understood clearly - it became clear. I do wish that there was more context in what this would be used for(fictional or not). I find that helps the building process and stops those ‘what-if’ moments.

I used Team Treehouse, Lynda, Stack Overflow, and the java/junit wiki as my major educational resources. Since this is my first java project, I

would love feedback if you have the time or want! Thanks.