Assignment #3

Due 11:59 p.m., Monday, 5 November

General Instructions

All homework and programming exercises must be prepared in accordance with the instructions given in the Syllabus. Each assignment must be submitted to your instructor by its stated deadline.

Citations: In accordance with expected scholarly and academic standards, if you reference outside textbooks, reference books, articles, websites, etc., or discuss an assignment with individuals inside or outside the class, you must document these by including appropriate citations or comments at prominent places in your submission such as in the header of the primary source file.

Identification: Put your name, course name, and assignment number as comments in each file you submit.

Goal

The goal for this assignment is to practice applying the modular design and Python 3 implementation techniques we have studied this semester. These include information hiding, abstract interfaces, contracts, type hints, program families, modular programming, object-oriented programming, testing, etc.

Assignment Description

1. This is an individual assignment (unless otherwise approved by the instructor).
2. For the Wally World Marketplace (WWM) Point-of-Sale (POS) Project given below:
   • Develop a set of Python 3 modules to solve the problem.
   • Use an approach based on information-hiding and abstract interfaces to make the system robust with respect to likely changes. Try to keep the coupling between modules low.
   
   You may use a mix of modular and object-oriented programming techniques as appropriate. Follow good practices.
   • Document the design of each module by identifying the “secret” of the module and clearly defining the syntax and semantics of its interface.
   • For at least one of the modules, develop two significantly different implementations that follow the same specification. (That is, the two implementations have the same interface and functionality but make different choices on how to realize the secret.)

3. If you wish to develop a different project, you may do so after describing the system adequately and getting permission from your instructor.

4. Design appropriate tests for your program and test the program thoroughly.

5. Document your program appropriately.

6. Submit the source code and documentation for your program and test driver, any needed instructions on how to run the program, and appropriate test output to Blackboard. Be sure to identify yourself in the materials turned in.

Wally World Marketplace Point-Of-Sale Project

Motivation

This project was motivated by CSci 450, Assignment #2 and Kata01: Supermarket Pricing, a code kata proposed by Dave Thomas.

Description

Wally World Marketplace (WWM) is a “big box” store selling groceries, dry goods, hardware, electronics, various service, etc. both in “brick and mortar” stores and online. In this project, we develop part of a Point-of-Sale (POS) system for WWM.

Here is a partial list of characteristics of and requirements for the system. We likely will need to discuss these issues further.
• Every distinct kind of item has an **IdTag** (e.g. barcode, RFID, QR, NFC, etc.) with a unique **code**. The values of the **code** are from a totally ordered set.

Initially we can assume that a **code** is representable as a Python 3 **int**, but this might evolve in the future.

• Every item has an associated **IdTag**. The **IdTag**’s associated **code** is either unique for the item or for the kind of item.

For example, a signed, first-edition copy of John Grisham’s *The Firm* might have a unique code, but loose Gala apples might have the same **code**.

• For our purposes, a shopping cart (physical or virtual) consists of a “sequence” of **codes** for items. The actual source of the sequence may vary (barcode scanner outputs, virtual shopping cart data structure, etc.) depending upon the circumstances.

Given that the source may vary, we may want to hide the “sequence” behind an abstract interface in Python 3. This might be a custom interface or a standard interface such as `collection.abc.Iterable`.

Question: Do we need additional information in addition to the code? Maybe some kind of measure?

• An inventory “database” associates each **code** with a “record” that includes the name of the item, its price (see next bullet), the quantity in the current inventory, and perhaps other information. The “database” might be an in-memory data structure (e.g. association list, Python **dict**), a relational database on disk, etc.

Given that the “database” location and structure may vary, we may want to hide it behind an abstract interface in Python 3. This might be a custom interface or a standard interface such as `collection.abc.Mapping`.

• The price may be straightforward, giving a price for the item. However, it might be more complicated such as “three for a dollar”, “buy two get one free”, “a dollar a pound”, “ten dollars and hour” (for a service), etc.)

See Dave Thomas, CodeKata, Kata01: Supermarket Pricing for more discussion of these issues.

Initially assume that the prices are in US currency, but this might evolve in the future.

• WWM is responsible for collecting any application taxes relative to the sale.

Initially, assume a single flat sales tax rate, but this might evolve in the future (e.g. taxes from various jurisdictions or for different purposes, different rates associated with different categories of items)
• Once a sale is complete, the system must reduce the inventory in the database for the items sold.

• WWM does not currently have a customer loyalty program, but it might choose to add on such a system in the future.

• The system must print a receipt for the purchase. This is an item that might be different depending upon the equipment at the checkout counter or for an online purchase.

A receipt might like the following for several simple items:

Wally World Marketplace

Vanilla yogurt cups (4)........1.88
Ground turkey (1 lb).........3.16
Toasted oat cereal..........2.99
Ground turkey (1 lb).........3.16
Black tea bags (100).........3.07
Athletic socks (6)...........8.25
Claw hammer.................7.88
32-in. television............139.49
Zero sugar cola (12).........3.34

Subtotal...................176.26
Tax..........................12.34
Total.......................188.60

Processing

In general, the WWMPOS system is expected to carry out the following steps. Of course, this algorithm itself might evolve.

1. Input the next item from the shopping cart.
   (You may need other data, such as weight, volume, or time period.)
2. Look up the item in the database. Gets its price.
3. If more items, go back to step 1.
4. Calculate the subtotal for the shopping cart.
5. Calculate the applicable sales and other taxes.
6. Calculate the total.
7. Format and print a receipt.
8. Change the inventory database to reflect the sale of the items in the shopping cart.