Multi-Paradigm Programming
Shop Assignment (70%)

Dominic Carr

October 21, 2021

This assignment is building on the shop program which I developed in the video series. You are tasked to add some additional functionality:

Functionality

- The shop CSV should hold the initial cash value for the shop.
- Read in customer orders from a CSV file.
  - That file should include all the products they wish to buy and in what quantity.
  - It should also include their name and their budget.
- The shop must be able to process the orders of the customer.
  - Update the cash in the shop based on money received.
    - It is important that the state of the shop be consistent.
    - You should create customer test files (CSVs) which cannot be completed by the shop e.g. customer wants 400 loaves of bread but the shop only has 20, or the customer wants 2 cans of coke but can only afford 1.
    - If these files don't exist **penalties will be applied**.
  - Know whether or not the shop can fill an order.
    - Thrown an appropriate error.
- Operate in a live mode, where the user can enter a product by name, specify a quantity, and pay for it. The user should be able to buy many products in this way.

Notes

- The above described functionality should be completed in Python and C. This is to be done in a procedural programming style.
- The second part of the assessment involves replicating the functionality of the shop in an Object-Oriented manner. This can be done in either Python or Java.
- You must complete a short report, around 3-5 pages, which compares the **solutions achieved** using the procedural approach and the object oriented approach.
  - Excessively long reports will be penalised.
  - The report should be about comparing your implementations.
- The live mode, and the input files, should have the exact same behaviour in ALL implementations.
  - For example I should be able to use the Python implementation in the same way as the C one i.e. same CSV files, and the same process when doing an order in live mode.
  - The "user experience" of each implementation should be **identical**.

Marking Scheme

- Procedural Python Program (20%)
  - Good Procedural Programming (**NOT OOP**) (10%)
  - Level of functionality (5%)
  - Documentation and code quality (5%)
- Procedural C Program (30%)
  - Good Procedural Programming (10%)
  - Level of functionality (10%)
  - Documentation and code quality (10%)
- OOP Java or Python Program (30%)
  - Good Object Oriented Programming (10%)
  - Level of functionality (10%)
  - Documentation and code quality (10%)
- Report (20%)
  - Document Quality (5%)
  - Analysis of similarities (7.5%)
  - Analysis of differences (7.5%