An Order Management System

DS3500: Advanced Programming with Data

Purpose

- Design a multi-component system in Python.
- Create generic and reusable data structures.
- Work with JSON data.
- Practice test-driven development with pytest.

Overview

Large companies rely on complex software to manage orders, inventory, shipping and receiving, and production planning. The software responsible for managing the flow of data throughout the enterprise is called an ERP (Enterprise Resource Planning) system. SAP, a company you may have heard of, is one of the largest ERP vendors in the world.

Arriving orders must be prioritized. It is not always just first-come, first-serve. Certain customers may be granted a higher priority simply because they are considered a more important (premier) customer. The order quantity may also be a factor. Companies often wish to prioritize larger, more profitable, orders to maximize sales revenue.

In this assignment, you will build a rudimentary order-management system that prioritizes incoming orders (in JSON format). To maintain the order priority you will implement a custom priority queue data structure. Like regular queues, orders can be added (enqueued) and removed (dequeued) but with a priority queue, the order sequence is maintained according to the relative priority of the order. You will employ test-driven methods to implement a comprehensive collection of unit tests aimed at validating the functionality of your priority queue.

In this scenario, orders are prioritized according to three attributes:

- **Priority (L = Low, M = Medium, H = High):** This represents the priority of the customer who placed the order. All High priority orders should be fulfilled before Medium priority orders and all Medium priority orders should be fulfilled before Low priority orders.

- **Order date:** Orders received earlier within the same customer priority level (Low/Medium/High) will be fulfilled earlier.

- **Quantity**: If two orders have the same customer priority (Low/Medium/High) and were received on the same date, then the larger order is given priority. The company wants to maximize sales revenue.

Deliverables

Submit a zip file containing a directory with the following files:

hw1_your_full_name

order_manager.py: Your main application problem which will read the JSON, transform the JSON to order objects, add orders to the priority queue, and report the fulfillment sequence. It is sufficient to just report the sequence of order identifiers. Invalid orders should raise exceptions that are appropriate caught and reported, but invalid orders are not added to the priority queue.

order.py: Your order object, defined with appropriate getters and setters.

priority_queue.py: This is where the orders are actually stored. It is sometimes called an orderbook. Here are the methods that your priority queue should support: __init__, enqueue (add an item to the queue), dequeue (remove the next item from the queue), peek (identify item at the head of the queue), size (the number of items in the queue), is_empty, to_list, clear (empty the queue), str (convert to a string representation)

tests: This folder contains unit tests for each type of object
order_manager_tests.py
order_tests.py
priority_queue_tests.py

README.md: General comments or instructions for the TAs

Some General Guidelines

- Received orders should be validated for correctness and ignored if invalid. Your code should employ appropriate exception-handling methods.
- Unit tests should be comprehensive to validate that the various objects behave correctly.
- Grading criteria include the correctness of your code, the completeness of your unit tests, style, documentation, and overall readability.
- The efficiency of your priority queue is not a critical factor.
- You can make use of any library you want but you must implement your own priority queue.
- Try to design your priority queue so that it could handle any type of object, not just Orders and it should flexibly support different prioritization schemes. One way to do this would be to register some sort of comparator function with the queue upon constructing the queue.