CPSC 2720 - Assignment 3

Overview

In this assignment you will:

- Implement an inventory and shipping system using four design patterns.
- Use various software engineering tools to help create quality software (i.e. version control, document generation, code coverage, static and style analysis, memory leak checking, continuous integration).

Problem Description

The Dunder Mifflin Paper Co. is revising their inventory and shipping system. They have asked your team to create a new software system with the following requirements:

Inventory System

- Add a warehouse to the system.
- Remove a warehouse to the system.
    - Any paper products in the removed warehouse is moved to any other warehouse.
- Add a paper product to the inventory of a warehouse.
- Change the quantity of a paper product stored in a warehouse.
- Find a paper product in the inventory. The paper product may be in multiple warehouses.
- List the total of paper products in the inventory from across all warehouses.

Ordering System

- Create a paper product order.
- View current orders.
- Edit/Update a paper product in an order.
- Submit an order.
- Cancel an order.
- View an order.
- Get the status of an order [`In Cart`, `Pending`, `Complete`]
- Add an order to a waitlist (if there was not enough paper products in stock)

Shipping

Paper products are shipped by different methods depending on the weight of the order:

ORDER WEIGHT	SHIPPING METHOD
<= 100 KG	Truck
101 – 1000 KG	Air
1001 – 10000 KG	Train
> 10000 KG	Ship

Design

1. Create a UML static structure (i.e. class) diagram using `dia` that shows the design of your software.
2. Export your diagram to `PNG` format.
3. Put the image of your design in the `design` folder at the root of your repository.

You are to use the following design patterns in your software system:

Design Pattern	*Functionality*
Factory Method	Creating the shipping method based on the weight of the order.
State	Representing the different states that an Order can have, and applying different behaviour depending on the mode.
Observer	Maintaining a waitlist of incomplete orders that will be attempted when the warehouse receives more inventory.
Strategy	Representing the shipping method used when the order has been completed.

Implementation

Implement your design. It is suggested that you proceed in an incremental manner as follows:

1. Create the fundamental classes needed for an inventory/ordering software system.
2. Implement the shipping types using the Strategy Pattern.
3. Implement a Factory Method to create the correct shipping object.
4. Implement the State pattern for indicating the status of an order.
5. Implement the observer pattern to handle backorders.

Notes

- A `Makefile` is provided which:
  - Builds and runs a testing executable (make tests).
  - Checks for memory leaks (make memcheck)
  - Runs static analysis (make static)
  - Runs style checking (make style)
  - Runs code coverage (make coverage)
  - Runs all of the checks (`make all`)

- A continuous integration configuration file (`.gitlab-ci.yml`) is provided for you. It is not expected that you will need to change this file.
- A `CPPLINT.cfg` that contains configuration for the `cpplint` style checker.
- A `Doxyfile` that contains configuration for the `doxygen` documentation generator.
- An example application (`inventory`) is provided to give you some ideas for your implementation of the application.

Grading

You will be graded based on your demonstrated understanding of the use of version control, good software engineering design practices, and design patterns. Examples of items the grader will be looking for include (but are not limited to):

- Design and implementation shows use of requested design patterns.
- Version control history shows an iterative progression in completing the assignment. It is expected that you will have at least 10 new commits in your repository (roughly one commit for each design pattern and accompanying test fixture).
- Version control repository contains no files that are generated by tools (e.g. object files, binary files, documentation files)
- Implementation shows an understanding of software engineering design principles, specifically for the four design patterns.
- Source code is appropriately documented using the principles discussed in class (e.g. all classes and methods) so that doxygen can extract the documentation.
- Line coverage of 90% or better, function coverage is not considered.
- Memory leak checking, static analysis and style analysis show no errors.
- The build on GitLab passes as of the assignment deadline.

Submission

There is no need to submit anything, as GitLab tracks links to forks of the assignment repository. However, you **must** ensure the following instructions are followed:

- Make sure that the permissions are correctly set for your repository on GitLab so the grader has access. **You will receive an automatic 0 (zero) for the assignment if the grader cannot access your repository.** Refer to "Assignment Set up" linked from the Individual Assignment section of your class Moodle page.
- Creating **an independent copy of the repository (i.e. not a fork) will result in an automatic 0 (zero)** as the marker will not be able to find it.