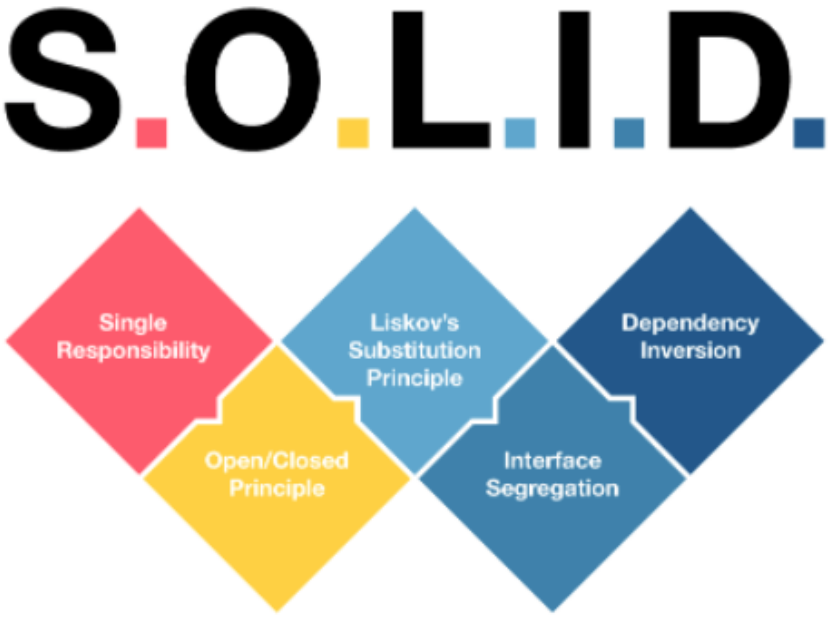

Coding kata bring it home - SOLID Principles edition.

In this kata, you will create a simple e-commerce shipping calculator application that adheres to the SOLID principles. The application should allow adding different shipping methods and calculate shipping costs based on the chosen method. You will focus on ensuring your design follows the Liskov Substitution Principle (LSP) and Dependency Inversion Principle (DIP) while also respecting the other SOLID principles.

Solid principles

Single Responsibility Principle (SRP): Each class should have only one reason to change.

Open/Closed Principle (OCP): Classes should be open for extension but closed for modification.

Liskov Substitution Principle (LSP): Subtypes must be substitutable for their base types without altering the correctness of the program.

Interface Segregation Principle (ISP): Clients should not be forced to depend on interfaces they do not use.

Dependency Inversion Principle (DIP): High-level modules should not depend on low-level modules; both should depend on abstractions. Abstractions should not depend on details; details should depend on abstractions.

Instructions

You have 90 minutes to complete the following tasks using Test-Driven Development (TDD). Write the tests first before implementing the functionality.

Start

This whole exercise can be done using the TDD method. Creating the test first and then implementing the functionality. Decide what method to use; what is important is that the code is 100% covered by tests so we can ship it to the client.

Clone the repo for this kata and create your own branch. **https://github.com/CrazyFrog-tech/SolidBringItHome.git**

1- Liskov substitution principle

Create StandardShippingMethod and ExpressShippingMethod classes that calculate the cost of the shiping

- StandardShippingMethod calculates the cost of the shipping by multiplying the weight by the distance by the standard cost, which is 0.5
- ExpressShippingMethod calculates the cost of the shipping by multiplying the weight by the distance by the express cost, which is 0.1
- Use Liskov substitution principle to create another class or interface which can calculate the cost for both methods

Commit your changes and continue to step 2...

2- Dependency Inversion principle

We don't want the app to depend directly on the core layer so we want to create an abstraction layer to do the shipping cost calculation.

Create a Shipping Calculator which can be used by the application and the core so that it satisfies the dependency inversion rule.

Commit your changes and continue to step 3...

3- Interface segregation

Calculate shipping duration which applies to express shipping only. return distance / 100 + 1 day for the express shipping?

Do your classes implement unnecessary methods which they don't use??

Commit your changes and continue to step 4...

4- Open Closed principle

Add a Membershipping method which calculates the cost of the shipping by multiplying the weight by the distance by the shipping member cost which is 0.2.

Add duration calculation which is distance / 100 + 2 days

Is your system open for extension closed for modification or not??

Commit your changes and continue to step 5...

5- Single responsibility

Is your system built with the single responsibility in mind. Do a refactoring round and see if everything is well named and fulfills the single responsibility principle.

Commit your changes..

Congratulations you have finished the solid principles training. You have now become a SOLID developer!!