**Ministerul Educaţiei și Cercetării al Republicii Moldova Universitatea Tehnică a Moldovei**

**Facultatea Calculatoare, Informatică și Microelectronică**

**Laboratory work 2:**

Structural Design Patterns

# Elaborated:

st. gr. FAF-222 Șarov Andrei

# Verified:

asist. univ. Furdui Alexandru

# Chişinău - 2024

**Objectives:**

* Study and understand the Structural Design Patterns.
* As a continuation of the previous laboratory work, think about the functionalities that your system will need to provide to the user.
* Implement some additional functionalities using structural design patterns.

**Main Tasks:**

 1. By extending your project, implement atleast 3 structural design patterns in your project:

* The implemented design pattern should help to perform the tasks involved in your system.
* The object creation mechanisms/patterns can now be buried into the functionalities instead of using them into the client.
* There should only be one client for the whole system.

2. Keep your files grouped (into packages/directories) by their responsibilities (an example project structure):

* client
* domain
  + factories
  + builder
  + models
* utilities
* data(if applies)

3. Document your work in a separate markdown file according to the requirements presented below (the structure can be extended of course):

* Topic of the laboratory work
* Author
* Introduction/Theory/Motivation
* Implementation & Explanation (you can include code snippets as well)
  + Indicate the location of the code snippet
  + Emphasize the main idea and motivate the usage of the pattern
* Results/Screenshots/Conclusions

**Used Design Patterns:**

* **Adapter Pattern:** Used to adapt a legacy system (e.g., LegacyProduct) to work with the new system interface, providing compatibility without modifying the legacy code.
* **Decorator Pattern:** Allows dynamic modification of Product behavior by adding additional features like discounts and taxes without altering the original class.
* **Facade Pattern:** Simplifies interaction with complex systems by providing a single interface (ProductFacade) that aggregates multiple functionalities.

**Implementation:**

**Step 1: Project Setup**

Organized the project with directories by functionality, including client, domain, models, and utilities, each containing relevant classes.

**Step 2: Adapter Pattern Implementation**

* Created LegacyProduct with an old method for retrieving product data.
* Implemented ProductAdapter in domain/adapter.py, adapting LegacyProduct's old\_get\_item to the new system interface using the get\_product method.

**Step 3: Decorator Pattern Implementation**

* Created the DiscountDecorator and TaxDecorator in domain/decorator.py to dynamically modify Product's price.
* Each decorator applied a specific modification (discount or tax) to the original Product class.

**Step 4: Facade Pattern Implementation**

* Developed ProductFacade in domain/facade.py to simplify access to various system functions, including legacy compatibility and price calculations.
* The facade utilized the adapter for legacy compatibility and decorators for price adjustments, providing a unified interface to the client.

**Step 5: Logging**

* Set up a simple logging utility in utilities/logger.py to track output in client/main.py.

**Step 6: Client Code and Testing**

* Created the main client code in client/main.py to test and demonstrate the combined functionalities of all patterns.

**Conclusions:**

This project demonstrated the practical benefits of structural design patterns in software development. By implementing the Adapter, Decorator, and Facade patterns, we enhanced system functionality and maintained code organization without altering existing components. The Adapter pattern enabled integration with legacy systems, the Decorator pattern allowed flexible, dynamic feature additions, and the Facade pattern provided a simplified, unified interface for the client. These design patterns improved modularity, maintainability, and scalability, meeting all project objectives and highlighting the importance of structural patterns in building robust, extensible systems.