COS 214 Practical Assignment 4

UNIVERSITEIT VAN PRETORIA
UNIVERSITY OF PRETORIA
YUNIBESITHI YA PRETORIA

- Date Issued: **26 September 2023**
- Date Due: **10 October 2023** at **11:00am**
- Submission Procedure: **Upload to ClickUP**
- Submission Format: **archive (zip or tar.gz)**

1 Introduction

1.1 Objectives

In this practical you will:

- Implement and use the Chain and adapter design patterns

1.2 Outcomes

When you have completed this practical you should:

- Understand the use of the chain and adapter design patterns
- Be able to demonstrate proper use of doxygen and git/github

2 Constraints

1. You must complete this assignment individually or in groups of two.
2. You may ask the Teaching Assistants for help but they will not be allowed to give you the solutions.
3. You must demo the system in the Labs at the conclusion of the prac and both partners need to be present if working in pairs

3 Submission Instructions

You are required to upload all your source files (that is `.h` and `.cpp`), your Makefiles, and a single PDF document and any data files you may have created, in a single archive to ClickUP before the deadline. A demonstration of the implementation will be conducted in the labs, It is required to create a main that allows for the proper demonstration of the patterns.

4 Mark Allocation

Task	**Marks**
Chain of responsibility	10
Adapter	10
TOTAL	20

5 Assignment Instructions

Task 1: Chain of responsibility ..... (10 marks)

Whenever our project become larger its natural to split the application into a number of components, these components may have differing terminology based on the layer of abstraction the distinction is made at. These could be differing services, modules, classes etc.

One key feature to these systems is that while the majority of the business logic might be contained within one module and not requiring the use of other sub-components we do end up in situations where multiple modules or sub-components have to be used to achieve the end result.

Consider the example of a 'claims' process for personal expenses for an organisation. Typically if employees are asked to plan an event there will be a process to reimburse them for minor expenses that they can voluntarily undergo during the planning, usually for small or inexpensive purchases.

An example of this might be that HR needs to set a budget, then the employee makes their purchases, after which they submit their claims back to HR, who will then approve the expense, which payroll will then be responsible to reimburse. During this process, from a design space, we include a variety of different systems ie. an ESS(employee self service) portal may be present for online management, MSS(Manager self service) for HR to view claims, Payroll to add the approved amount to the employees income.

Its here where, almost naturally, the chain of responsibility pattern emerges where we define a chain of handlers that can process an incoming request without the need to violate the single responsibility principle.

For this pattern we will be implementing an authentication chain, as seen in frameworks such as spring boot, that loosely follows the FIDO2 authentication pattern. FIDO2 is a passwordless login flow that uses the concept of PKI to ensure a users identity:

- a user will request a nonce value from our authentication server.
- the server will respond with a newly generated nonce and persist this to the DB
- the user will sign the nonce with their private key
- the server will then check the signature and verify that it matches the user
- the server will generate a token that it will respond with to the user
- the user will include the token in all their following requests to identify themselves

1.1 Implement the following using the chain of responsibility pattern: (8)

- First handlers check if we requested a nonce, if yes generate and persist, if no pass to next handles
- Next handler checks if we are attempting to validate a sign-in nonce, if yes verify and respond to user either a failure or a fresh token, if no pass to next handler
- Next we check if the token is valid, if yes pass to next handler, if no return an error to the user
- Lastly we can assume the user has been authenticated and is authorized, so we process the request and return the result to the user.

You have the freedom to select how you generate nonce values and tokens. How you verify the signature (if you want you can implement an actual system, or you can work out another simple way to verify a particular user signed the request). As well as how a request will be processed, be that printing to the console what the request was or doing anything else.

1.2 Using doxygen automatically generate documentation for your code (2)

Task 2: Adapter ..... (10 marks)

Adapter is typically used whenever we have code that we cannot alter, it may be legacy code that tends to be highly coupled and difficult to alter without severe ramifications or it may be some 3rd party library that we have incorporated. Here we would typically write an adapter that wraps the offending class(Third party/legacy etc) and provides the interface we expect to use for us.

For this task let us consider an application that has an existing 3rd party ORM(Object Relational Mapping) configured, This ORM holds a reference to our database connection and performs SQL queries on our behalf

and returns an object that is configured from the data received from the database. This ORM only supports typical SQL grammar such as (SELECT * FROM table). We have found a new database that works incredibly well but unfortunately has a different grammar in which we specify the FROM declaration first ie (FROM table SELECT *).

2.1 Develop an application to simulate the above scenario, using an adapter to translate the incompatible grammars and pass the request and its response back to the ORM. (Please do not attempt to use an actual database as then you will need to select between incompatible grammars such as with MySQL and SQLite where some features are not universally available) (5)

2.2 Use git to manage the development of this application and ensure you demonstrate a proper merge. preferably use the gitflow workflow(To be discussed as part of the git lecture and readily available online) (5)