Міністерство освіти і науки України
Національний технічний університет України
“Київський політехнічний інститут імені Ігоря Сікорського”
Факультет інформатики та обчислювальної техніки
Кафедра інформаційних систем та технологій

Лабораторна робота №4

Технології розроблення програмного забезпечення

«*ШАБЛОНИ «SINGLETON», «ITERATOR», «PROXY», «STATE», «STRATEGY»*»

Варіант 29

Виконав:
студент групи ІА-13
Хілько І.А.

Перевірив:
Мягкий М. Ю.

Київ 2023

Тема: Шаблони Singleton, Iterator, Proxy, State, Strategy.

Варіант:
Система для колективних покупок. (**State**, Chain of responsibility, Abstract factory, Mediator, Composite, Client-server).

Система дозволяє створити список групи для колективної покупки, список що потрібно купити з орієнтовною вартістю кожної позиції та орієнтовною загальною вартістю, запланувати хто що буде купляти. Щоб користувач міг відмітити що він купив, за яку суму, з можливістю прикріпити чек. Система дозволяє користувачу вести списки бажаних для нього покупок, з можливістю позначати списки, які будуть доступні для друзів (як списки, що можна подарувати користувачеві). Система дозволяє добавляти інших користувачів в друзі.

Хід роботи:
1. Реалізувати не менше 3-х класів та шаблон State.

State – це поведінковий патерн проектування, що дає змогу об’єктам змінювати поведінку в залежності від їхнього стану.

Патерн State реалізований за допомогою Enum ServerState, який визначає можливі стани сервера (NOT_LOGGED_IN та LOGGED_IN). Поведінка сервера змінюється в залежності від його поточного стану.

```
class ServerState(Enum):
    NOT_LOGGED_IN = 1
    LOGGED_IN = 2
```

У класі Server є атрибут current_state, який представляє стан сервера. Поведінка сервера змінюється в залежності від цього стану, як видно в методі handle_client.

```
class Server:
    def __init__(self, host='127.0.0.1', port=5559):
        self.host = host
        self.port = port
        self.server = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
        self.server.bind((self.host, self.port))
        self.server.listen()
        self.current_state = ServerState.NOT_LOGGED_IN
        self.register_handler = RegisterHandler()
        self.login_handler = LoginHandler()
        self.login_register_handler = LoginRegisterCommandHandler()
        self.client_states = {}
        print(f'Server is listening on {self.host}:{self.port}')
```

```
def handle_client(self, client_socket):
        self.client_states[client_socket] = ServerState.NOT_LOGGED_IN
        current_user = None
        while True:
            data = client_socket.recv(1024)
            if not data:
                break
            decoded_data = data.decode('utf-8')
            print(f'Received from client: {decoded_data}')

            if self.client_states[client_socket] == ServerState.NOT_LOGGED_IN:
                if decoded_data.startswith(('REGISTER', 'LOGIN')):
                    command, *command_args = decoded_data.split()
                    current_user = self.login_register_handler.handle(
                        self, client_socket, current_user, command, command_args
                    )
                    if current_user and command == 'LOGIN':
                        self.client_states[client_socket] = ServerState.LOGGED_IN
                else:
                    client_socket.send('Please log in first.'.encode('utf-8'))
            elif self.client_states[client_socket] == ServerState.LOGGED_IN:
                self.process_user_command(client_socket, current_user,
decoded_data.split())

        del self.client_states[client_socket]
        client_socket.close()
```

У класі LoginHandler, при успішному вході користувача, стан сервера оновлюється на LOGGED_IN.

```
class LoginHandler(Handler):
    def handle(self, server, client_socket, current_user, command_args):
        if len(command_args) == 2:
            username, password = command_args
            try:
                with DatabaseManager(SQLiteDatabaseFactory()) as cursor:
                    cursor.execute('SELECT * FROM users WHERE username=? AND
password=?', (username, password))
                    user = cursor.fetchone()
                    if user:
                        client_socket.send('Login successful!'.encode('utf-8'))
                        server.current_state = ServerState.LOGGED_IN
                        return user
                    else:
                        client_socket.send('Invalid username or
password.'.encode('utf-8'))
            except Exception as e:
                print(f'Error during login: {e}')
                client_socket.send('Error during login. Please try
again.'.encode('utf-8'))
        else:
            client_socket.send('Invalid login format. Usage: LOGIN username
password'.encode('utf-8'))

        return None
```

Патерн State використовується для управління різними станами сервера та визначення поведінки, пов'язаної з кожним станом. Залежно від стану, сервер обробляє команди по-різному в методі handle_client.

Висновок: Отже, під час виконання лабораторної роботи, я реалізував частину функціоналу робочої програми у вигляді класів та їхньої взаємодії для досягнення конкретних функціональних можливостей та застосував шаблон “State” при реалізації програми.