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 №9

РІЗНІ ВИДИ ВЗАЄМОДІЇ ДОДАТКІВ:
CLIENT-SERVER, PEER-TO-PEER,
SERVICE-ORIENTED ARCHITECTURE

Варіант 10

Виконала
студентка групи ІА – 13:
Луценко Юлія

Перевірив:
Мягкий Михайло

Завдання:

1. Ознайомитися з короткими теоретичними відомостями.

2. Реалізувати частину функціоналу робочої програми у вигляді класів і їх взаємодій для

досягнення конкретних функціональних можливостей.

3. Реалізувати взаємодію програми в одній з архітектур відповідно до обраної теми.

Варіант:

10. VCS all-in-one (iterator, adapter, factory method, facade, visitor, p2p)

Клієнт для всіх систем контролю версій повинен підтримувати основні команди і дії (commit, update, push, pull, fetch, list, log, patch, branch, merge, tag) для 3-х основних систем управління версіями (svn, git, mercurial), а також мати можливість вести реєстр репозиторіїв (і їх типів) і відображати дерева фіксації графічно

Хід роботи

Кожен екземпляр програми може виступати як сервер, обслуговуючи підключення клієнтів, і як клієнт, взаємодіючи з іншими серверами. Це дозволяє створювати децентралізовану мережу, де кожен вузол може взаємодіяти з іншими вузлами.

Опис роботи:

При запуску скрипта одночасно запускає:

start_server_peer:

- Запускає серверний потік.
- за допомогою функції find_free_port визначає вільний порт
- створює серверний сокет, який прив’язується до визначеного порту та чекає на підключення.
- для кожного підключення створюється окремий потік. викликає handle_client_peer_wraperr для обробки підключення клієнта

start_client_peer:

- запускає клієнтський потік
- визначає працюючий порт для підключення.
- створює клієнтський сокет, який приєднується до іншого peer'у

Кожен екземпляр може виступати як сервер для інших клієнтів як і клієнт для інших серверів.

```
def is_port_free(port):
    with socket.socket(socket.AF_INET, socket.SOCK_STREAM) as s:
        return s.connect_ex((SI, port)) != 0
2 usages  new *
def find_free_port():
    for port in server_ports:
        if is_port_free(port):
            return port
    return None
5 usages  new *
def start_server_peer():
    logging.basicConfig(filename=_file_log, level=logging.INFO)
    free_port = find_free_port()

    if free_port is None: return
    server_socket = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
    server_socket.bind((SI, free_port))
    server_socket.listen(1)
    logging.info(f"Server started on port {free_port}. Waiting for connections...")

    while True:
        client_socket, client_address = server_socket.accept()
        logging.info(f"Accepted connection from {client_address}")
        client_thread = threading.Thread(target=handle_client_peer_wrapper,args=(client_socket,))
        client_thread.start()
```

```
def handle_client_peer_wrapper(client_socket):
    try:
        handle_peer(client_socket, status=0)
    except (ConnectionAbortedError, ConnectionResetError) as exp:
        logging.error(f"Connection reset: {exp}")
    finally:
        client_socket.close()

5 usages  new *
def start_client_peer():
    port = find_free_port()
    if port is None: return
    client_socket = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
    client_socket.connect((SI, port))
    try:
        data = client_socket.recv(4096).decode('utf-8')
        print(data)
        while True:
            command = input('[-]')
            client_socket.sendall(command.encode('utf-8'))
            data = client_socket.recv(4096).decode('utf-8')
            print(data)
            if command.lower() == "exit":
                break
    finally:
        client_socket.close()
```

```
2 usages  new *
def handle_peer(client_socket, status):
    if status == 0:
        welcome_message = (
            b"\033[32mWelcome to VCS Console App!\n"
            b"This application allows you to interact with various Version Control Systems (G
            b"Choose a Version Control System (VCS):\n"
            b"1. Git\n"
            b"2. Mercurial\n"
            b"3. SVN\n"
            b"4. Exit\n")


    client_socket.sendall(welcome_message)

    while True:
        command = client_socket.recv(1024).decode(encoding).strip()
        if command in ["1", "2", "3"]:
            vcs_type = "Git" if command == "1" else "Mercurial" if command == "2" else "SVN"
            client_socket.sendall(b"\033[31m[*]Enter path to " + vcs_type.encode(encoding) +
            version_control = VersionControlFactory.create_version_control_system(connection,
            repo_path = convert_to_absolute_path(client_socket.recv(1024).decode(encoding).st
            vcs_facade = VCSFacade(connection)
            process_vcs_commands(client_socket, vcs_facade, version_control, repo_path,vcs_ty
        elif command.lower() == "exit":
            client_socket.sendall(b"[*] Exiting...\n")
            break
        else:
            client_socket.sendall(b"[*] Invalid command. Please enter 'git', 'svn', 'mercuria
```

```
if __name__ == "__main__":
    server_thread = threading.Thread(target=start_server_peer)
    server_thread.start()
    client_thread = threading.Thread(target=start_client_peer)
    client_thread.start()
    server_thread.join()
    client_thread.join()
```

Висновок: я реалізувала взаємодію програми в p2p архітектурі відповідно до обраної теми.