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

Технології розроблення програмного забезпечення

«Різні види взаємодії додатків: CLIENT-SERVER, PEER-TO-PEER, SERVICE-ORIENTED ARCHITECTURE»

Виконала
студентка групи ІА-14:

Логінова І.С.

Перевірив:

Мягкий М.Ю.

Київ 2023

Тема: Різні види взаємодії додатків: CLIENT-SERVER, PEER-TO-PEER, SERVICE-ORIENTED ARCHITECTURE.

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

Паттерн Peer-To-Peer

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.

```python
def is_port_free(port):
    with socket.socket(socket.AF_INET, socket.SOCK_STREAM) as s:
        return s.connect_ex((SI, port)) != 0


3 usages
def find_free_port():
    for port in server_ports:
        if is_port_free(port):
            return port
    return None
```

```python
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()
```

```python
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()
```

```python
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()
```

```python
def handle_client_peer_wrapper(client_socket):
    try:
        db = DataBase(db_name)
        db.create_tables()
        handle_peer(client_socket, db)
    except (ConnectionAbortedError, ConnectionResetError) as exp:
        logging.error(f"Connection reset: {exp}")
    finally:
        client_socket.close()
```

```python
def handle_peer(client_socket, db):
    client_socket.sendall(b"Choose a VCS type (git, mercurial, svn), write 'sar' to show active repositories, or 'exit'
    while True:
        command = client_socket.recv(1024).decode('utf-8').strip()
        if command.lower() in ["git", "svn", "mercurial"]:
            vcs_type = command.lower()
            client_socket.sendall(b"Enter path to " + vcs_type.encode('utf-8') + b" repository: ")
            repo_path = client_socket.recv(1024).decode('utf-8').strip()
            adapter = VCSFA().create_vcs(client_socket, vcs_type, db)
            facade = VCSF(adapter)
            process_vcs_commands(client_socket, facade, vcs_type, repo_path, db)
        elif command.lower() == "sar":
            show_active_repositories(db, client_socket)
        elif command.lower() == "exit":
            client_socket.sendall(b"Exiting...\n")
            break
        else:
            client_socket.sendall(b"Invalid command. Please enter 'git', 'svn', 'mercurial', or 'exit'.\n")
```

```python
def process_vcs_commands(client_socket, facade, vcs_type, repo_path, db):
    client_socket.sendall(f"[{vcs_type} [{repo_path}]]".encode('utf-8'))
    command_iterator = CommandIterator(client_socket)
    command_executor = Executor()

    while True:
        command = client_socket.recv(1024).decode('utf-8').strip()
        if command.lower() == "back":
            handle_peer(client_socket, db)
        else:
            command_iterator.add_command(command)
            for cmd in command_iterator:
                command_executor.visit(client_socket, vcs_type, cmd, facade, repo_path)
```

Висновок: при виконанні цієї лабораторної роботи я реалізувала частину функціоналу робочої програми у вигляді класів та їхньої взаємодії для досягнення конкретних функціональних можливостей із застосуванням паттерну Peer-To-Peer при реалізації програми.