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 №8

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

Варіант 23

Виконала студентка групи ІА–13:

Сиваченко Дар'я Євгенівна

Перевірив :

Мягкий Михайло Юрійович

Київ 2023

Тема: Project Management software

Хід роботи

В даній лабораторній роботі було реалізовано простий приклад клієнт-серверної архітектури. Клієнт може створювати Task та відправляти його на сервер для збереження та обробки.Було використано мережеве з’єднання на основі сокетів.Основні складові:

Створення серверного сокету: створюється об'єкт ServerSocket, який слухає з'єднання на певному порті (12345). У разі, якщо виникає помилка при створенні серверного сокету, обробка помилки відбувається у блоку catch, де виводиться інформація про помилку за допомогою e.printStackTrace()

Прийняття та обробка з'єднання клієнта:цикл сервера, який безкінечно очікує нові з'єднання в циклі while(true). Після прийняття з'єднання в методі accept(), створюються ObjectInputStream та ObjectOutputStream для читання та запису об'єктів через це з'єднання. У разі виникнення помилок вводу/виводу або відсутності класу під час десеріалізації (ClassNotFoundException), обробка помилок також відбувається в блоку catch, де виводиться відповідна інформація про помилку.

Обробка отриманого завдання:Ця частина коду отримує об'єкт Task від клієнта через потік ObjectInputStream. Після отримання об'єкту Task перевіряється, чи не є він null. Якщо об'єкт не є null, виводиться інформація про отримане завдання, така як його ID, назва та опис. У випадку, якщо отриманий об'єкт Task є null, виводиться повідомлення про помилку, пов'язану з отриманням порожнього завдання.

Відправлення відповіді клієнту:Після обробки отриманого завдання від клієнта, сервер відправляє відповідь клієнту через ObjectOutputStream. У цьому випадку відправляється рядок, який підтверджує отримання та обробку завдання на сервері.

Встановлення з'єднання з сервером: встановлюється з'єднання з сервером за допомогою Socket, вказуючи його IP-адресу та порт. Використовуються об'єкти ObjectOutputStream та ObjectInputStream для взаємодії з сервером через потоки вводу та виводу об'єктів. Обробка помилок, які можуть виникнути під час цього з'єднання, відбувається у блоку catch, де виводиться інформація про помилку за допомогою e.printStackTrace()

Відправлення об'єкта Task на сервер:Створюється новий об'єкт Task з визначеним ID, назвою та описом. Після цього об'єкт відправляється на сервер через ObjectOutputStream за допомогою методу writeObject().

Отримання відповіді від сервера:Клієнт очікує відповідь від сервера через ObjectInputStream та метод readObject(). Отриманий об'єкт перетворюється на рядок типу String, і виводиться інформація про відповідь сервера у консоль.

Висновок:

В даній лабораторній роботі було розглянуто простий приклад клієнт-серверної архітектури на основі Java з використанням мережевого з'єднання через сокети. На сервері створюється серверний сокет, який слухає вхідні з'єднання. Після отримання з'єднання, сервер приймає об'єкт типу `Task`, виводить його дані у консоль та надсилає підтвердження клієнту про успішне отримання.Клієнт встановлює з'єднання з сервером, створює об'єкт `Task`, відправляє його на сервер та отримує відповідь від сервера.Основні аспекти реалізації включають використання сокетів для з'єднання, передачу об'єктів через потоки вводу та виводу, а також обробку винятків.Ця лабораторна робота є простим прикладом клієнт-серверної архітектури, яка може бути розширена та вдосконалена для різноманітних цілей, таких як передача складніших даних, реалізація безпеки та автентифікації користувачів у реальних програмних рішеннях.

Відповіді на контрольні питання

1. **Що таке SOA?**
   Сервіс-орієнтована архітектура (SOA, англ. service-oriented architecture) —- модульний підхід до розробки програмного забезпечення, заснований на використанні розподілених, слабо пов’язаних (англ. Loose coupling) замінних компонентів, оснащених стандартизованими інтерфейсами для взаємодії за стандартизованими протоколами.
2. **Якими принципами керується SOA?**
   Інтерфейси компонентів в сервіс-орієнтованій архітектурі інкапсулюють деталі реалізації (операційну систему, платформу, мову програмування) від інших компонентів, таким чином забезпечуючи комбінування і багаторазове використання компонентів для побудови складних розподілених програмних комплексів, забезпечуючи незалежність від використовуваних платформ та інструментів розробки, сприяючи масштабованості і керованості створюваних систем.

3. **У чому полягають переваги та недоліки клієнт-серверної моделі?**

Переваги клієнт-серверної моделі:

- **Простота та гнучкість**: Дозволяє легко масштабувати системи, розширювати їх функціонал та модифікувати за потребою.
- **Централізація даних та ресурсів**: Сервери зберігають централізовану інформацію, що сприяє кращому контролю, забезпеченню доступу та збереженню даних.
- **Контроль доступу**: Можливість встановлення прав доступу та автентифікації на сервері забезпечує більш високий рівень безпеки.
- **Оновлення на стороні сервера**: Можливість впровадження змін на сервері без необхідності модифікації клієнтського ПЗ.

Недоліки клієнт-серверної моделі:

- **Залежність від сервера:** Якщо сервер перестане працювати, це може призвести до відмови у обслуговуванні всіх клієнтів.
- **Залежність від мережі**: Клієнтські застосунки вимагають постійного підключення до мережі для взаємодії з сервером.
- **Обмеження масштабування**: При великому навантаженні на сервер може виникнути проблема з його працездатністю та відповідальністю за обробку всіх запитів.
- **Необхідність підтримки різних версій**: При оновленні серверного ПЗ може виникнути необхідність підтримки старих версій клієнтського ПЗ.
- **Проблеми з безпекою**: Оскільки сервер централізований, він може стати об'єктом атак або перехоплення даних.

4.У чому полягають переваги та недоліки однорангової моделі взаємодії?

Переваги однорангової моделі:

- **Рівність вузлів**: Кожен вузол може функціонувати як клієнт та сервер, розподіляючи навантаження та забезпечуючи більшу швидкість взаємодії.
- **Розширюваність мережі:** Додавання нових вузлів не створює однієї центральної точки, що відповідає за обробку всіх запитів.
- **Менша вразливість**: Відсутність центрального вузла зменшує вплив відмови одного вузла на роботу всієї системи.

- **Масштабні можливості:** Мережі P2P можуть бути легко розширені шляхом додавання нових вузлів без значних змін у конфігурації.

Недоліки однорангової моделі:

- **Проблеми безпеки**: Зазвичай важче забезпечити високий рівень безпеки в однорангових мережах через розподілену природу системи.
- **Складність керування**: Управління ресурсами та обміном даними в однорангових мережах може бути складнішим через розподілену структуру.
- **Пропускна здатність та швидкість**:Швидкість та доступність даних може залежати від швидкості та доступності інших вузлів в мережі.
- **Витрати на мережу:** Однорангові мережі можуть вимагати більших ресурсів для підтримки спільного обміну даними.
- **Потреба у вдосконалених алгоритмах**: Організація обміну даними в одноранговій мережі вимагає спеціалізованих алгоритмів та протоколів.