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
З дисципліни «Технології розроблення програмного забезпечення»
Тема: **«РІЗНІ ВИДИ ВЗАЄМОДІЇ ДОДАТКІВ: CLIENT-SERVER, PEER-TO-PEER, SERVICE-ORIENTED ARCHITECTURE»**
Download manager

Виконала:
Студентка групи ІА-22
Степанюк-Боримська А. І.

Перевірив:
Мягкий М. Ю.

Київ-2024

Зміст

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

Мета:
Вивчити основні моделі взаємодії між додатками, такі як «CLIENT-SERVER», «PEER-TO-PEER», «SERVICE-ORIENTED ARCHITECTURE», розібратися в їхніх принципах функціонування, особливостях застосування та вибору в залежності від вимог проєкту, а також здобути навички ефективного використання цих архітектурних підходів для створення масштабованих і надійних програмних рішень.

Хід роботи
1. Реалізувати функціонал для роботи в розподіленому оточенні (логіку роботи)

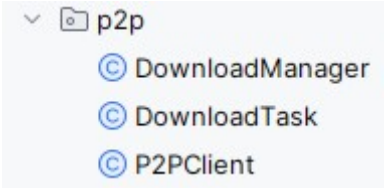

Рис. 1 — Структура проекту

Під час виконання лабораторної роботи було розроблено 3 класи, які реалізують функціонал для роботи в розподіленому оточенні (рис. 1). Нижче наведено детальний опис кожного з цих класів:

1. **DownloadManager**
   Призначення:
   Клас управляє завантаженням файлів та підтримує функціонал для роботи з одноранговими з'єднаннями в P2P-мережі. Він є центральним компонентом, який взаємодіє з іншими класами для виконання основних завдань.
   Основні функції:
   - Додавання завантаження: Метод addDownload створює нове завдання для завантаження файлу за вказаною URL-адресою і зберігає його у внутрішній колекції завдань.
   - Керування завантаженням: Підтримуються методи pauseDownload і resumeDownload, які дозволяють призупиняти та відновлювати завантаження за унікальним ідентифікатором завдання.

- Обмін файлами: За допомогою методу sendFile можна передавати файл іншому вузлу в мережі, використовуючи P2P-з'єднання.
- Статистика: Метод showStatistics виводить інформацію про стан усіх активних завантажень.

2. **DownloadTask**

   Призначення:

   Клас представляє окреме завдання завантаження файлу. Він відповідає за обробку завантаження у багатопоточному режимі з можливістю призупинення та відновлення.

   Основні функції:

   - Ідентифікація завдання: Кожне завантаження має унікальний ідентифікатор (id), який генерується при створенні.
   - Завантаження файлу: У методі run реалізована логіка отримання даних з URL і запису їх у локальний файл.
   - Керування процесом: Завдання може бути призупинене (pause) або відновлене (resume) без втрати прогресу.
   - Статистика завантаження: Метод getStatistics надає інформацію про URL, ідентифікатор завдання, а також обсяг завантажених даних.

3. **P2PClient**

   Призначення:

   Клас реалізує функціонал для організації однорангового (P2P) обміну файлами між вузлами в мережі. Включає серверну та клієнтську логіку для прийому і передачі файлів.

   Основні функції:

   - Запуск P2P-сервера: Метод startServer створює серверний сокет, який слухає вхідні з'єднання на заданому порту. Кожен підключений клієнт обробляється в окремому потоці.
   - Передача файлу: Метод sendFile дозволяє передавати файли іншому вузлу, вказуючи адресу та порт отримувача. Передача відбувається через сокетне з'єднання.
   - Прийом файлу: Метод handleClient обробляє запити від клієнтів на прийом файлу. Прийнятий файл зберігається в локальній файловій системі.

2. Реалізувати взаємодію розподілених частин

```java
public class DownloadManagerApp {
    public static void main(String[] args) throws IOException {
        System.out.println("Enter P2P server port:");
        BufferedReader reader = new BufferedReader(new InputStreamReader(System.in));
        int port = Integer.parseInt(reader.readLine());

        DownloadManager manager = new DownloadManager(port);
        System.out.println("Download Manager with P2P Server Started.");
        System.out.println("Commands: ADD [URL] [FILE], PAUSE [ID], RESUME [ID], SHOW, CONNECT [HOST] [PORT], EXIT");

        while (true) {
            String input = reader.readLine();
            String[] parts = input.split( regex: " ", limit: 4);

            switch (parts[0].toUpperCase()) {
                case "ADD":
                    manager.addDownload(parts[1], parts[2]);
                    break;
                case "PAUSE":
                    manager.pauseDownload(parts[1]);
                    break;
                case "RESUME":
                    manager.resumeDownload(parts[1]);
                    break;
                case "SHOW":
                    manager.showStatistics();
                    break;
                case "SEND":
                    manager.sendFile(parts[1], parts[2], Integer.parseInt(parts[3]));
                    break;
                case "EXIT":
                    System.exit( status: 0);
                    break;
                default:
                    System.out.println("Invalid command.");
            }
        }
    }
}
```

Рис. 2 — Точка входу в програму

Цей клас реалізує основний інтерфейс взаємодії користувача з функціоналом завантаження файлів та однорангового обміну (P2P). Він запускає P2P-сервер та дозволяє взаємодіяти із системою через командний рядок. Використовуючи два або більше екземпляри цієї програми, можна організувати роботу розподіленого середовища для обміну файлами між вузлами.

Перевірка роботи

```
Enter P2P server port:
226
Download Manager with P2P Server Started.
Commands: ADD [URL] [FILE], PAUSE [ID], RESUME [ID], SHOW, SEND [FILE] [HOST] [PORT], EXIT
P2P Server started on port: 226
add https://cdn.pixabay.com/photo/2016/10/03/17/11/cosmos-flower-1712177_640.jpg flower.jpg
show
ID: d876dbba-bd71-4ceb-8f02-414127353433, URL: https://cdn.pixabay.com/photo/2016/10/03/17/11/cosmos-flower-1712177_640.jpg, Downloaded: 37552/37552 bytes
send flower.jpg 127.0.0.1 229
Sending file: flower.jpg to 127.0.0.1:229
File sent successfully.
```

Рис. 3 — Перший вузол

```
Enter P2P server port:
229
Download Manager with P2P Server Started.
Commands: ADD [URL] [FILE], PAUSE [ID], RESUME [ID], SHOW, SEND [FILE] [HOST] [PORT], EXIT
P2P Server started on port: 229
Receiving file: flower.jpg from /127.0.0.1:58016
File received successfully. Saved as received_flower.jpg
```

Рис. 4 — Другий вузол

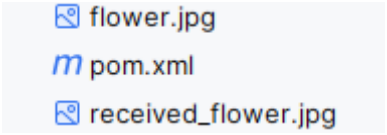

Рис. 5 — Отримані файли

У цьому прикладі було успішно продемонстровано взаємодію двох вузлів P2P-системи в межах одного проєкту. Спочатку на першому вузлі, який був запущений на порту 226, користувач здійснив завантаження з інтернет-ресурсу файлу зображення flower.jpg за допомогою команди add. Після успішного завершення завантаження, було відправлено цей файл на другий вузол, який працював на порту 229, за допомогою команди send.

Другий вузол, який отримав файл на порту 229, обробив запит і успішно прийняв файл, зберігши його під іменем received_flower.jpg. Виведені повідомлення на екрані кожного вузла підтверджують, що передача і прийом файлів відбулися без помилок.

В результаті виконання цього сценарію, обидва файли — оригінальний flower.jpg, що був завантажений з інтернет-ресурсу, та отриманий файл received_flower.jpg, — опинились в одному проєкті, оскільки обидва вузли працювали в межах однієї файлової системи. Це підтверджує правильну роботу P2P-системи, що дозволяє передавати файли між різними вузлами і зберігати їх у локальних папках кожного з вузлів.

Висновки:

У результаті реалізації цієї програми, ми успішно налаштували взаємодію розподілених частин за допомогою P2P (peer-to-peer) технології. Завдяки цьому досягнуто ефективний обмін файлами між двома вузлами, без необхідності центрального сервера. Такий підхід дозволяє знизити навантаження на сервери, підвищити масштабованість системи та забезпечити більш високу надійність, оскільки кожен вузол може діяти як сервер та клієнт одночасно. P2P забезпечує швидку передачу даних, гнучкість у масштабуванні та зниження витрат на інфраструктуру.