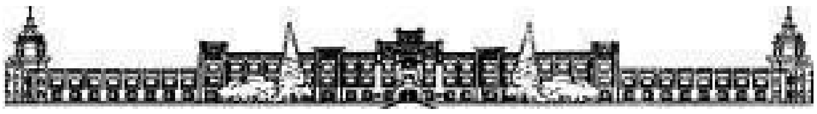


Міністерство освіти і науки України

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: «Теорія розробки програмного забезпечення»

Download Manager

P2P

Виконав:

студент групи

ІА-14 Юлдашев

А.Х.

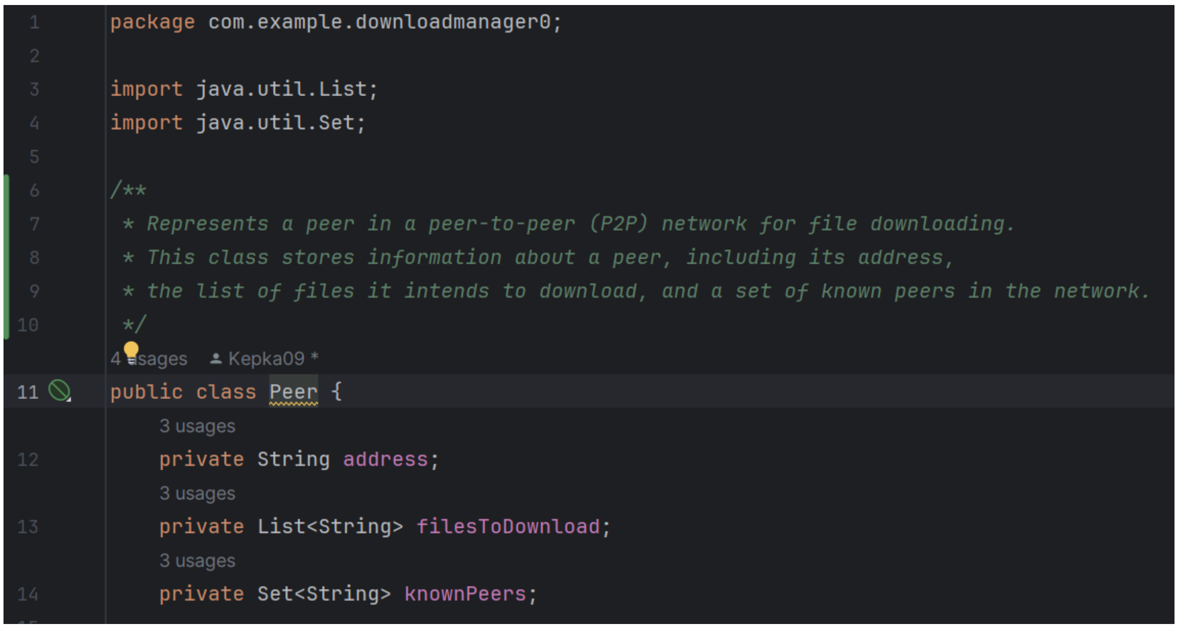
Київ 2023

Мета:

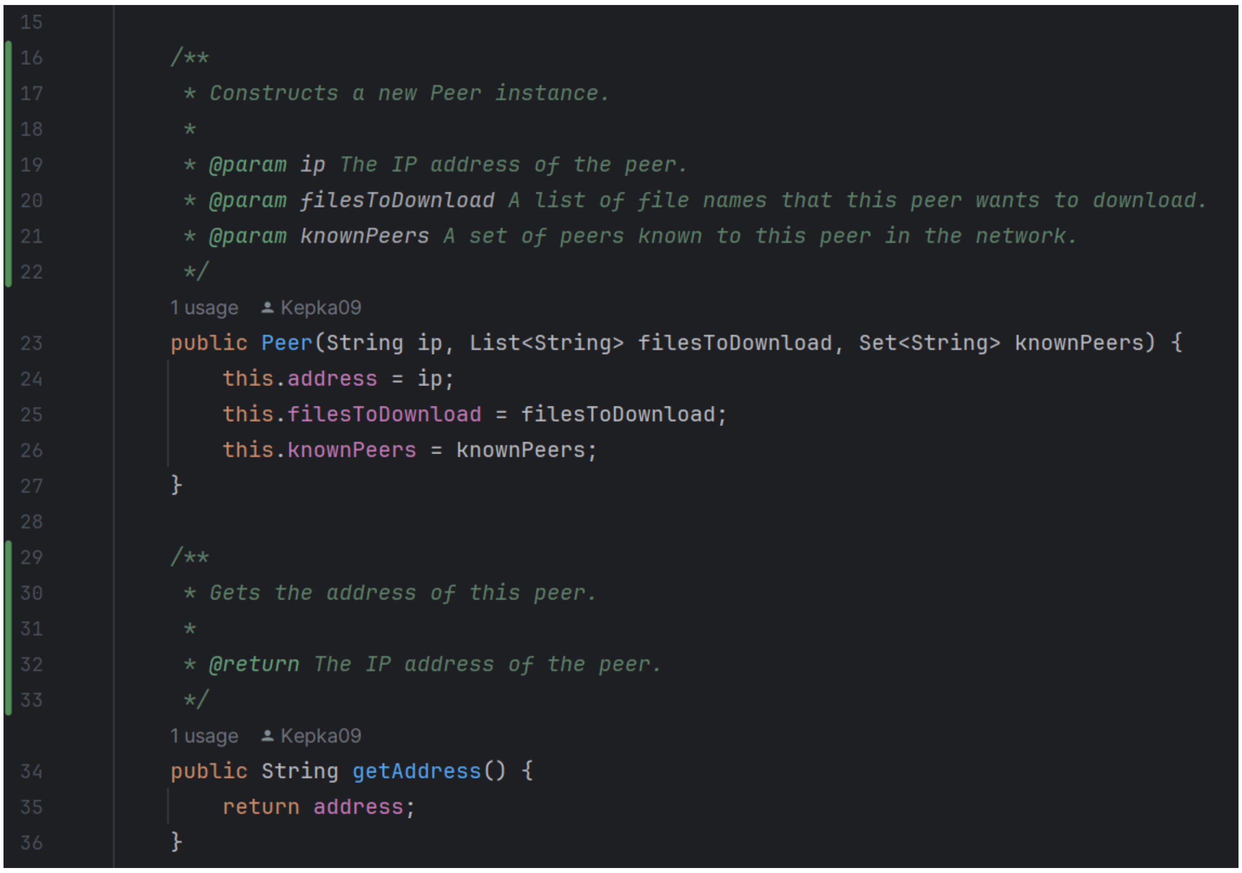
Реалізувати P2P в проекті на тему DownloadManager

Код з коментарями:

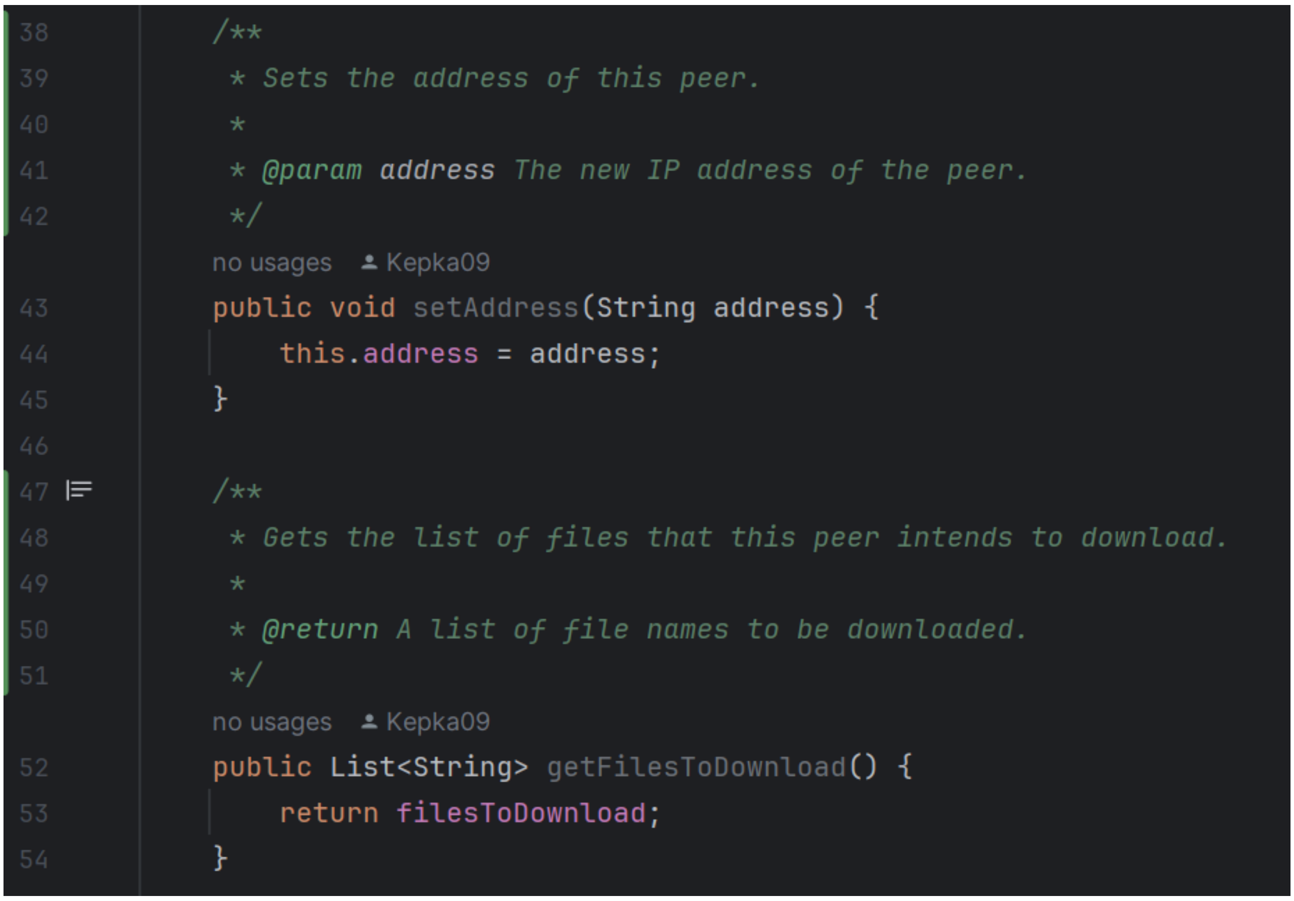
Peer



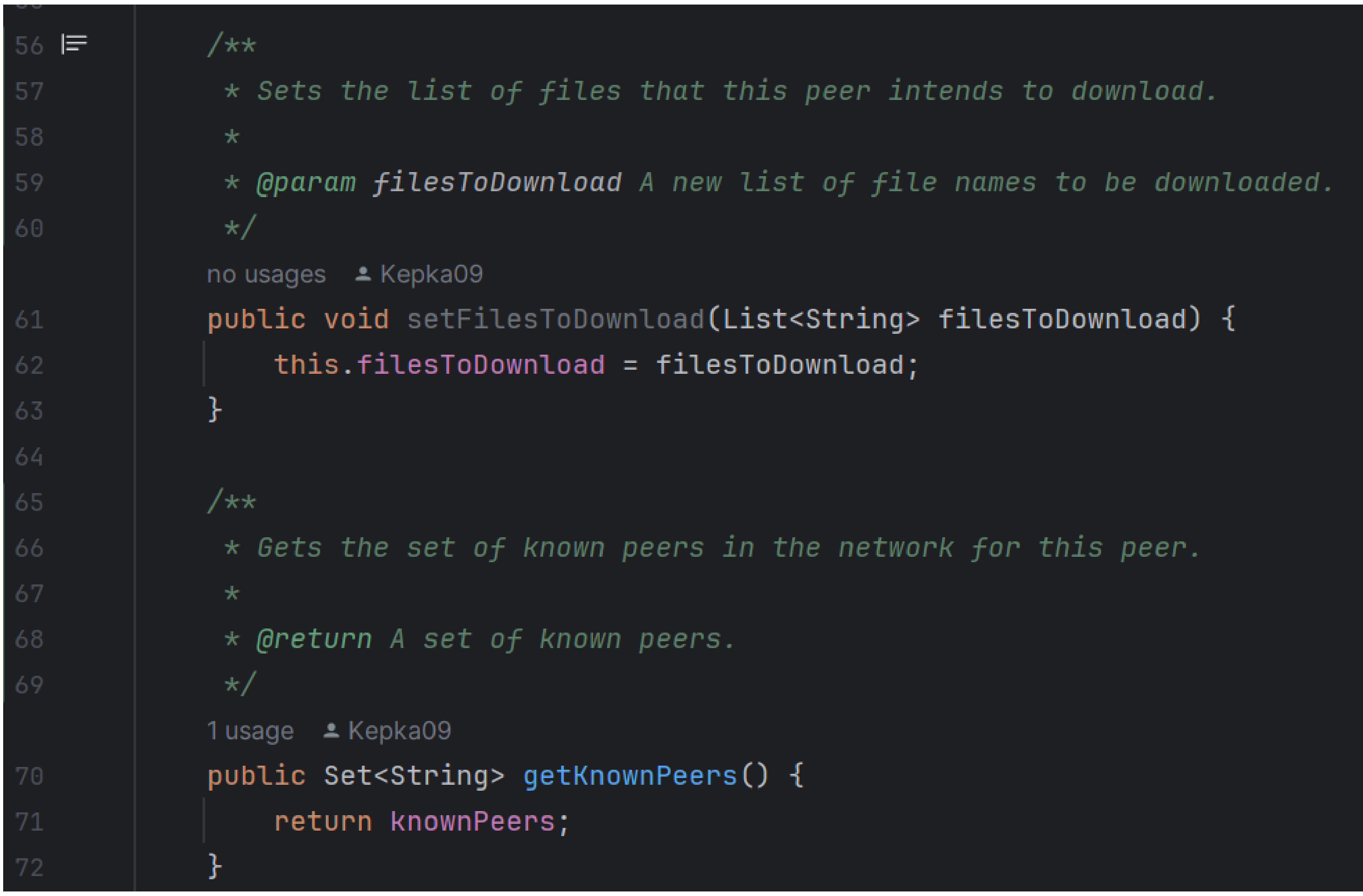
```java
package com.example.downloadmanager0;

import java.util.List;
import java.util.Set;

/**
 * Represents a peer in a peer-to-peer (P2P) network for file downloading.
 * This class stores information about a peer, including its address,
 * the list of files it intends to download, and a set of known peers in the network.
 */
4 usages   Kepka09 *
public class Peer {
    3 usages
    private String address;
    3 usages
    private List<String> filesToDownload;
    3 usages
    private Set<String> knownPeers;
```



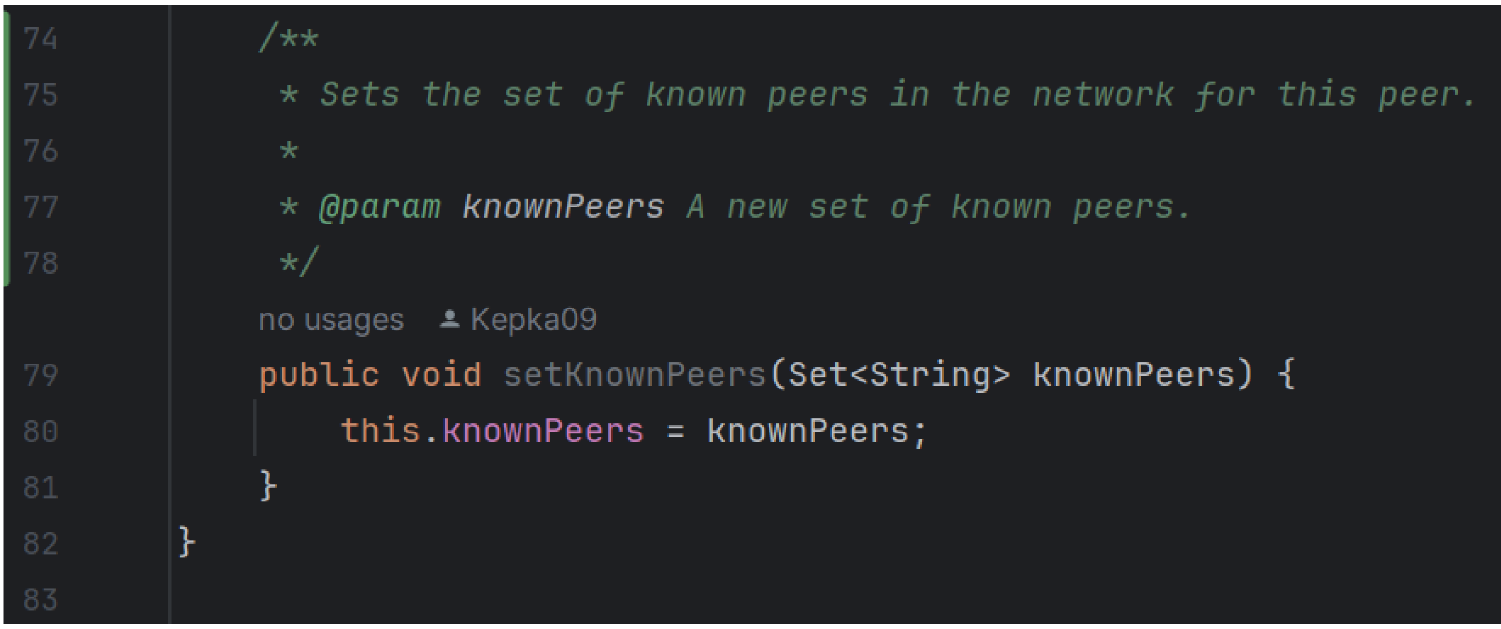
```java

    /**
     * Constructs a new Peer instance.
     *
     * @param ip The IP address of the peer.
     * @param filesToDownload A list of file names that this peer wants to download.
     * @param knownPeers A set of peers known to this peer in the network.
     */
    1 usage   Kepka09
    public Peer(String ip, List<String> filesToDownload, Set<String> knownPeers) {
        this.address = ip;
        this.filesToDownload = filesToDownload;
        this.knownPeers = knownPeers;
    }

    /**
     * Gets the address of this peer.
     *
     * @return The IP address of the peer.
     */
    1 usage   Kepka09
    public String getAddress() {
        return address;
    }
```



```
    /**
     * Sets the address of this peer.
     *
     * @param address The new IP address of the peer.
     */
    no usages  Kepka09
    public void setAddress(String address) {
        this.address = address;
    }

    /**
     * Gets the list of files that this peer intends to download.
     *
     * @return A list of file names to be downloaded.
     */
    no usages  Kepka09
    public List<String> getFilesToDownload() {
        return filesToDownload;
    }
```

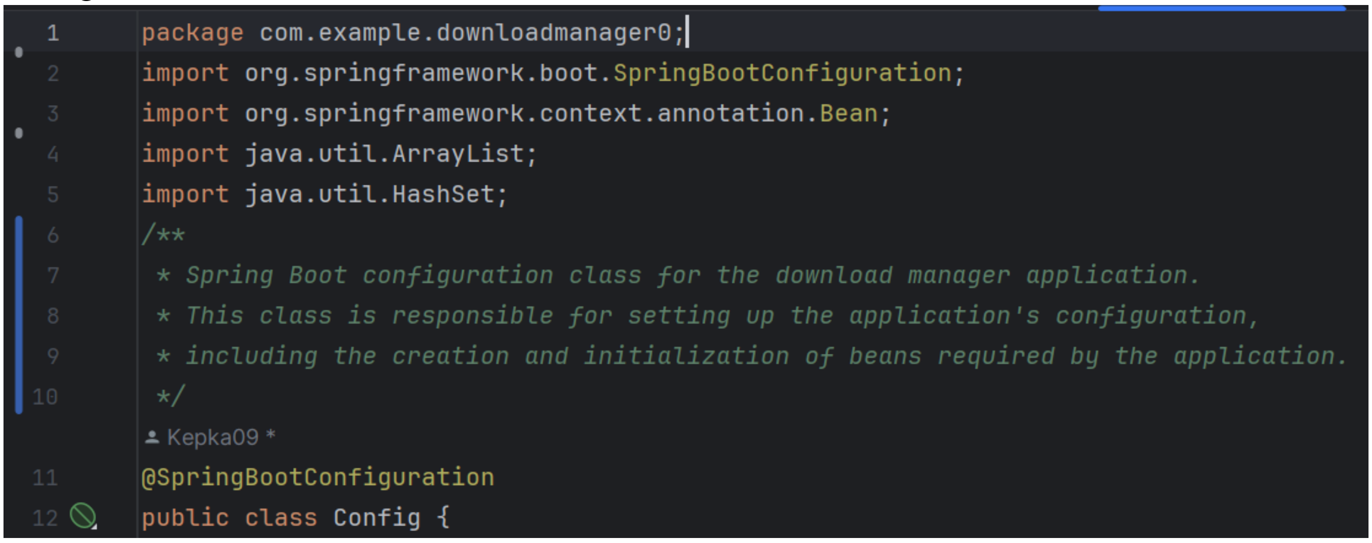


```
    /**
     * Sets the list of files that this peer intends to download.
     *
     * @param filesToDownload A new list of file names to be downloaded.
     */
    no usages  Kepka09
    public void setFilesToDownload(List<String> filesToDownload) {
        this.filesToDownload = filesToDownload;
    }

    /**
     * Gets the set of known peers in the network for this peer.
     *
     * @return A set of known peers.
     */
    1 usage  Kepka09
    public Set<String> getKnownPeers() {
        return knownPeers;
    }
```

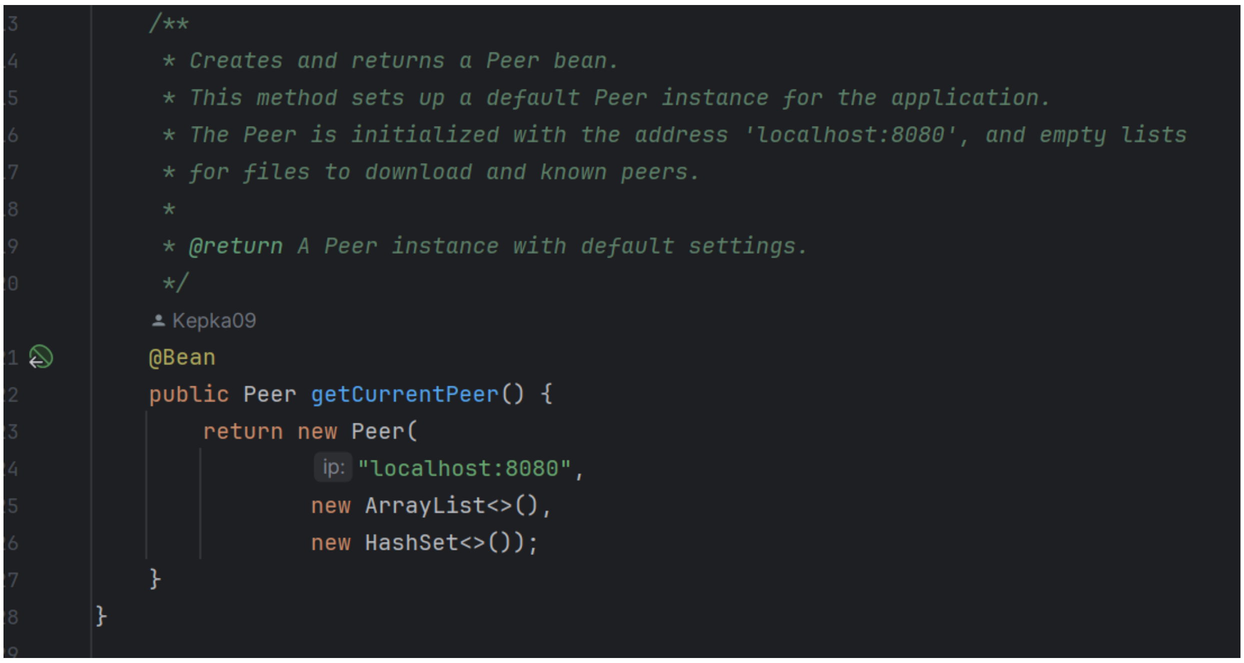


```
    /**
     * Sets the set of known peers in the network for this peer.
     *
     * @param knownPeers A new set of known peers.
     */
    no usages  Kepka09
    public void setKnownPeers(Set<String> knownPeers) {
        this.knownPeers = knownPeers;
    }
}
```

Config

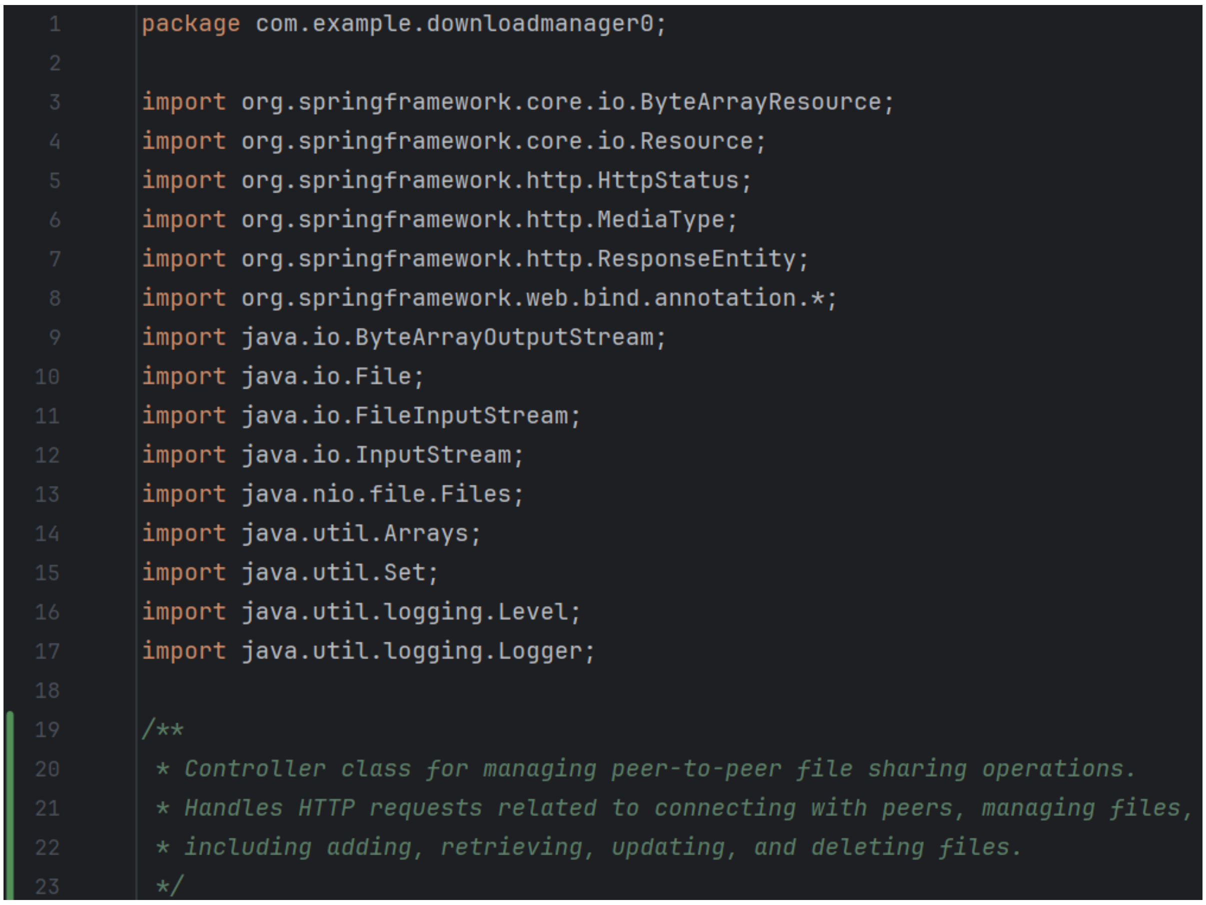


```
package com.example.downloadmanager0;
import org.springframework.boot.SpringBootConfiguration;
import org.springframework.context.annotation.Bean;
import java.util.ArrayList;
import java.util.HashSet;
/**
 * Spring Boot configuration class for the download manager application.
 * This class is responsible for setting up the application's configuration,
 * including the creation and initialization of beans required by the application.
 */
 Kepka09 *
@SpringBootConfiguration
public class Config {
```

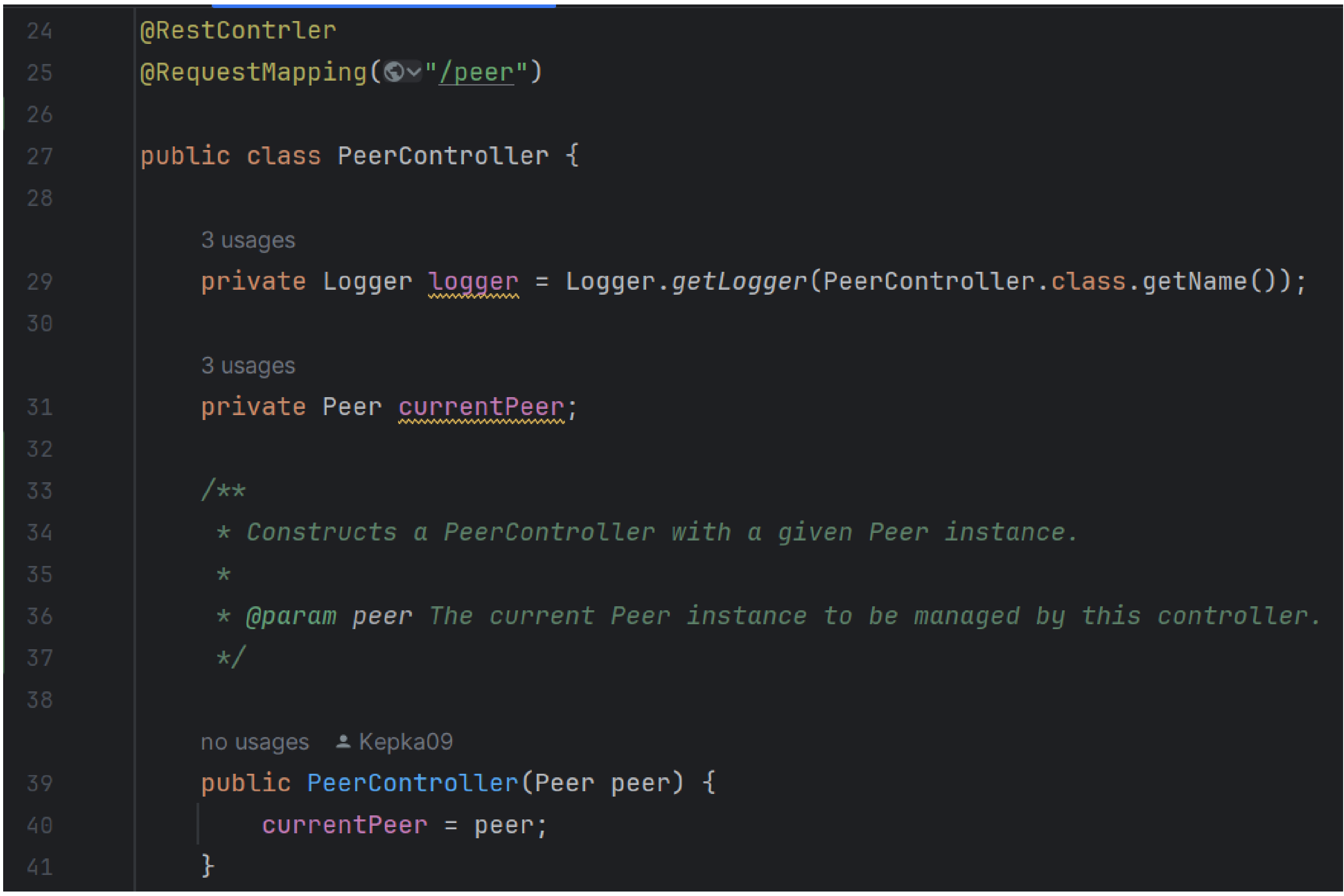


```
    /**
     * Creates and returns a Peer bean.
     * This method sets up a default Peer instance for the application.
     * The Peer is initialized with the address 'localhost:8080', and empty lists
     * for files to download and known peers.
     *
     * @return A Peer instance with default settings.
     */
     Kepka09
    @Bean
    public Peer getCurrentPeer() {
        return new Peer(
                ip: "localhost:8080",
                new ArrayList<>(),
                new HashSet<>());
    }
}
```

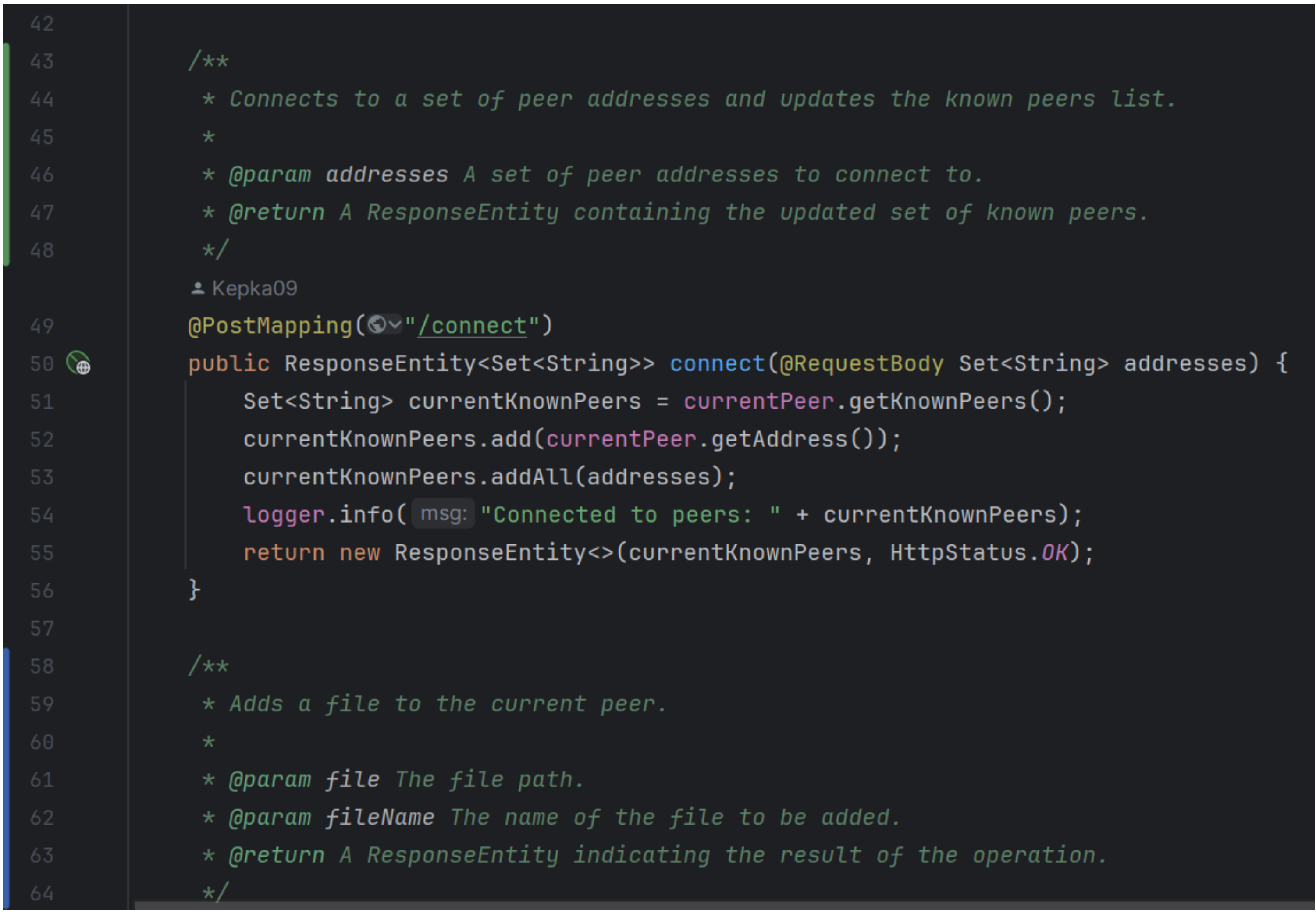
PeerController



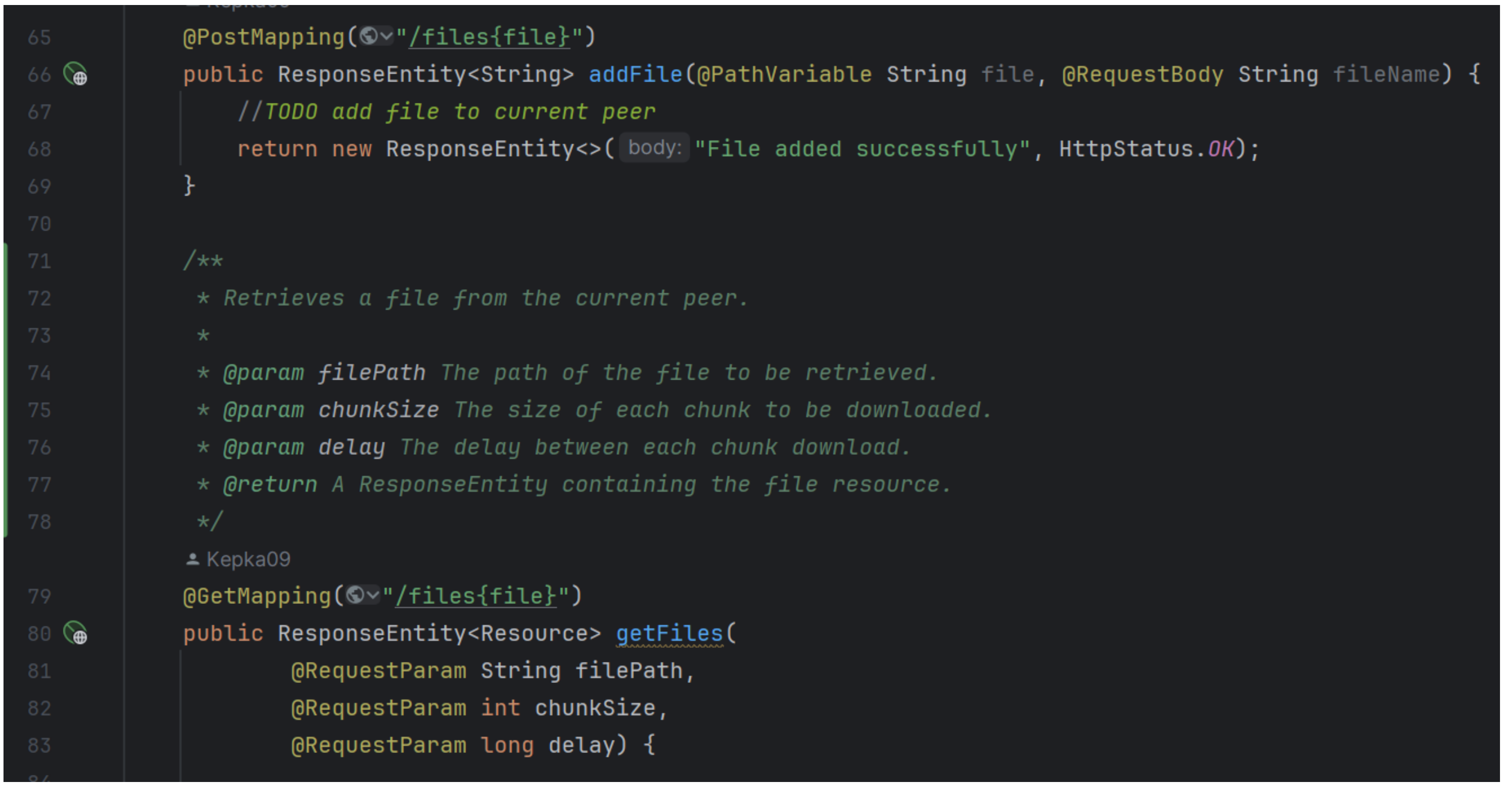
```java
package com.example.downloadmanager0;

import org.springframework.core.io.ByteArrayResource;
import org.springframework.core.io.Resource;
import org.springframework.http.HttpStatus;
import org.springframework.http.MediaType;
import org.springframework.http.ResponseEntity;
import org.springframework.web.bind.annotation.*;
import java.io.ByteArrayOutputStream;
import java.io.File;
import java.io.FileInputStream;
import java.io.InputStream;
import java.nio.file.Files;
import java.util.Arrays;
import java.util.Set;
import java.util.logging.Level;
import java.util.logging.Logger;

/**
 * Controller class for managing peer-to-peer file sharing operations.
 * Handles HTTP requests related to connecting with peers, managing files,
 * including adding, retrieving, updating, and deleting files.
 */
```



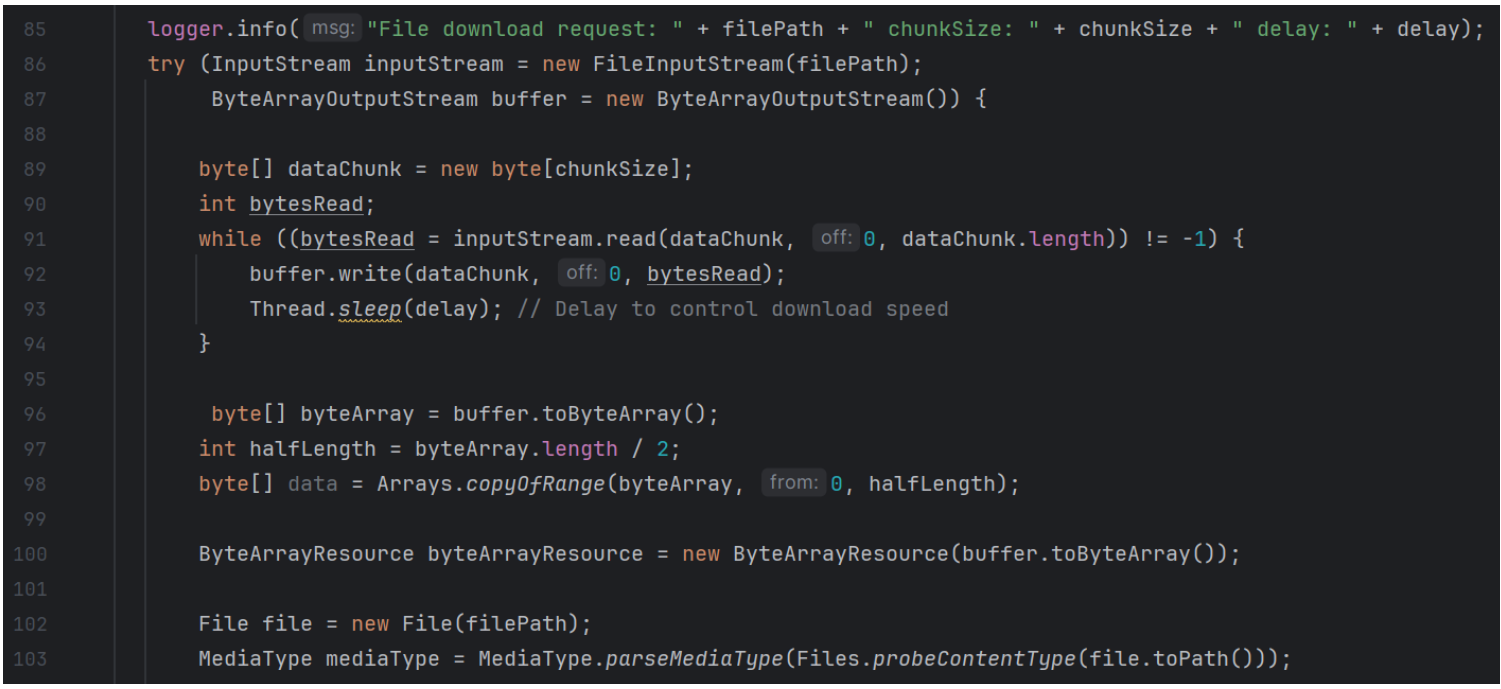
```java
@RestContrler
@RequestMapping("/peer")

public class PeerController {

    3 usages
    private Logger logger = Logger.getLogger(PeerController.class.getName());

    3 usages
    private Peer currentPeer;

    /**
     * Constructs a PeerController with a given Peer instance.
     *
     * @param peer The current Peer instance to be managed by this controller.
     */

    no usages   Kepka09
    public PeerController(Peer peer) {
        currentPeer = peer;
    }
```



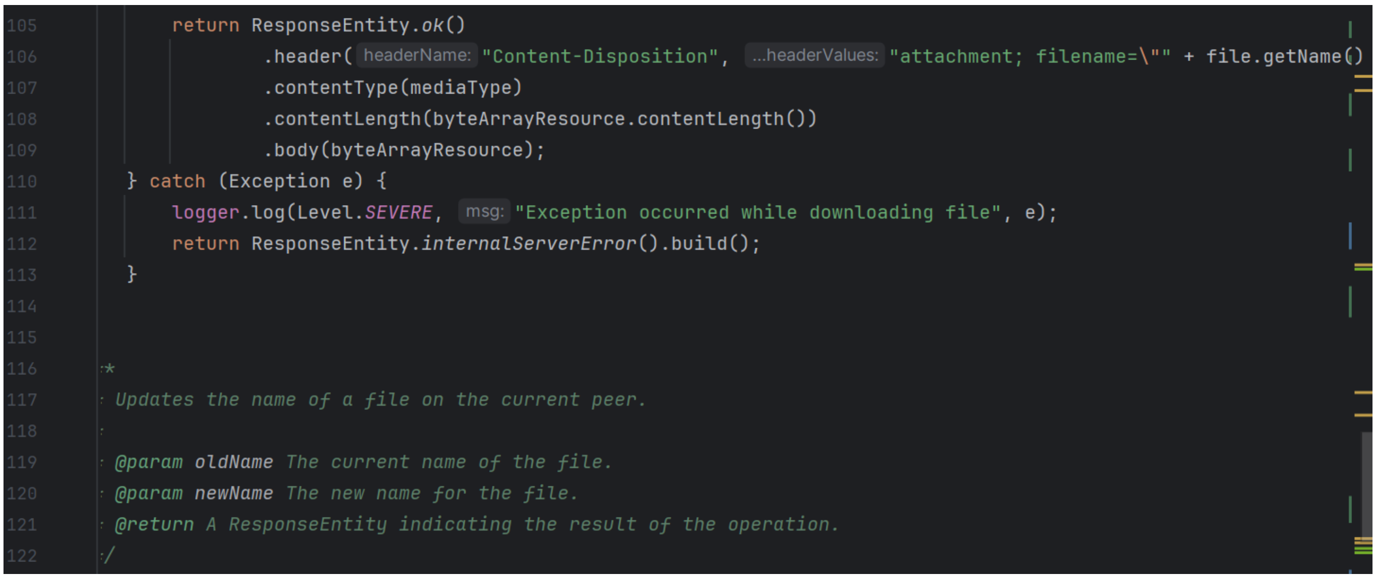
```java

    /**
     * Connects to a set of peer addresses and updates the known peers list.
     *
     * @param addresses A set of peer addresses to connect to.
     * @return A ResponseEntity containing the updated set of known peers.
     */
    Kepka09
    @PostMapping("/connect")
    public ResponseEntity<Set<String>> connect(@RequestBody Set<String> addresses) {
        Set<String> currentKnownPeers = currentPeer.getKnownPeers();
        currentKnownPeers.add(currentPeer.getAddress());
        currentKnownPeers.addAll(addresses);
        logger.info( msg: "Connected to peers: " + currentKnownPeers);
        return new ResponseEntity<>(currentKnownPeers, HttpStatus.OK);
    }

    /**
     * Adds a file to the current peer.
     *
     * @param file The file path.
     * @param fileName The name of the file to be added.
     * @return A ResponseEntity indicating the result of the operation.
     */
```



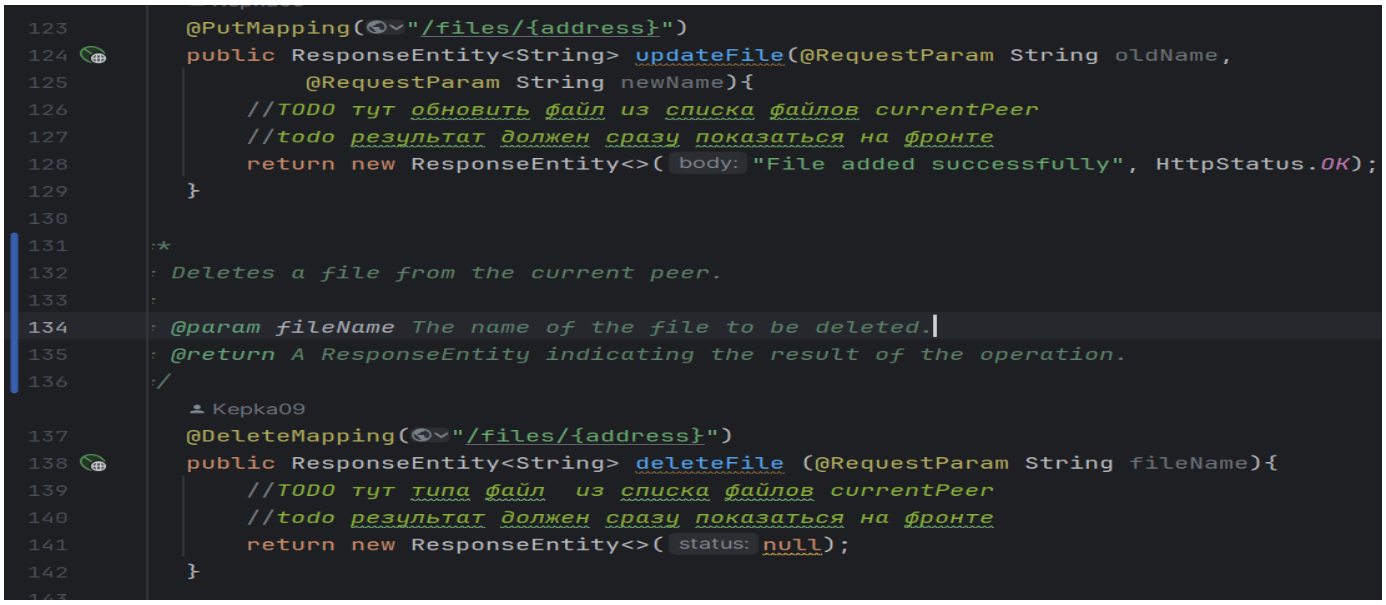
```java
    @PostMapping("/files{file}")
    public ResponseEntity<String> addFile(@PathVariable String file, @RequestBody String fileName) {
        //TODO add file to current peer
        return new ResponseEntity<>( body: "File added successfully", HttpStatus.OK);
    }

    /**
     * Retrieves a file from the current peer.
     *
     * @param filePath The path of the file to be retrieved.
     * @param chunkSize The size of each chunk to be downloaded.
     * @param delay The delay between each chunk download.
     * @return A ResponseEntity containing the file resource.
     */
    Kepka09
    @GetMapping("/files{file}")
    public ResponseEntity<Resource> getFiles(
            @RequestParam String filePath,
            @RequestParam int chunkSize,
            @RequestParam long delay) {
```



```java
        logger.info( msg: "File download request: " + filePath + " chunkSize: " + chunkSize + " delay: " + delay);
        try (InputStream inputStream = new FileInputStream(filePath);
             ByteArrayOutputStream buffer = new ByteArrayOutputStream()) {

            byte[] dataChunk = new byte[chunkSize];
            int bytesRead;
            while ((bytesRead = inputStream.read(dataChunk,  off: 0, dataChunk.length)) != -1) {
                buffer.write(dataChunk,  off: 0, bytesRead);
                Thread.sleep(delay); // Delay to control download speed
            }

             byte[] byteArray = buffer.toByteArray();
            int halfLength = byteArray.length / 2;
            byte[] data = Arrays.copyOfRange(byteArray,  from: 0, halfLength);

            ByteArrayResource byteArrayResource = new ByteArrayResource(buffer.toByteArray());

            File file = new File(filePath);
            MediaType mediaType = MediaType.parseMediaType(Files.probeContentType(file.toPath()));
```

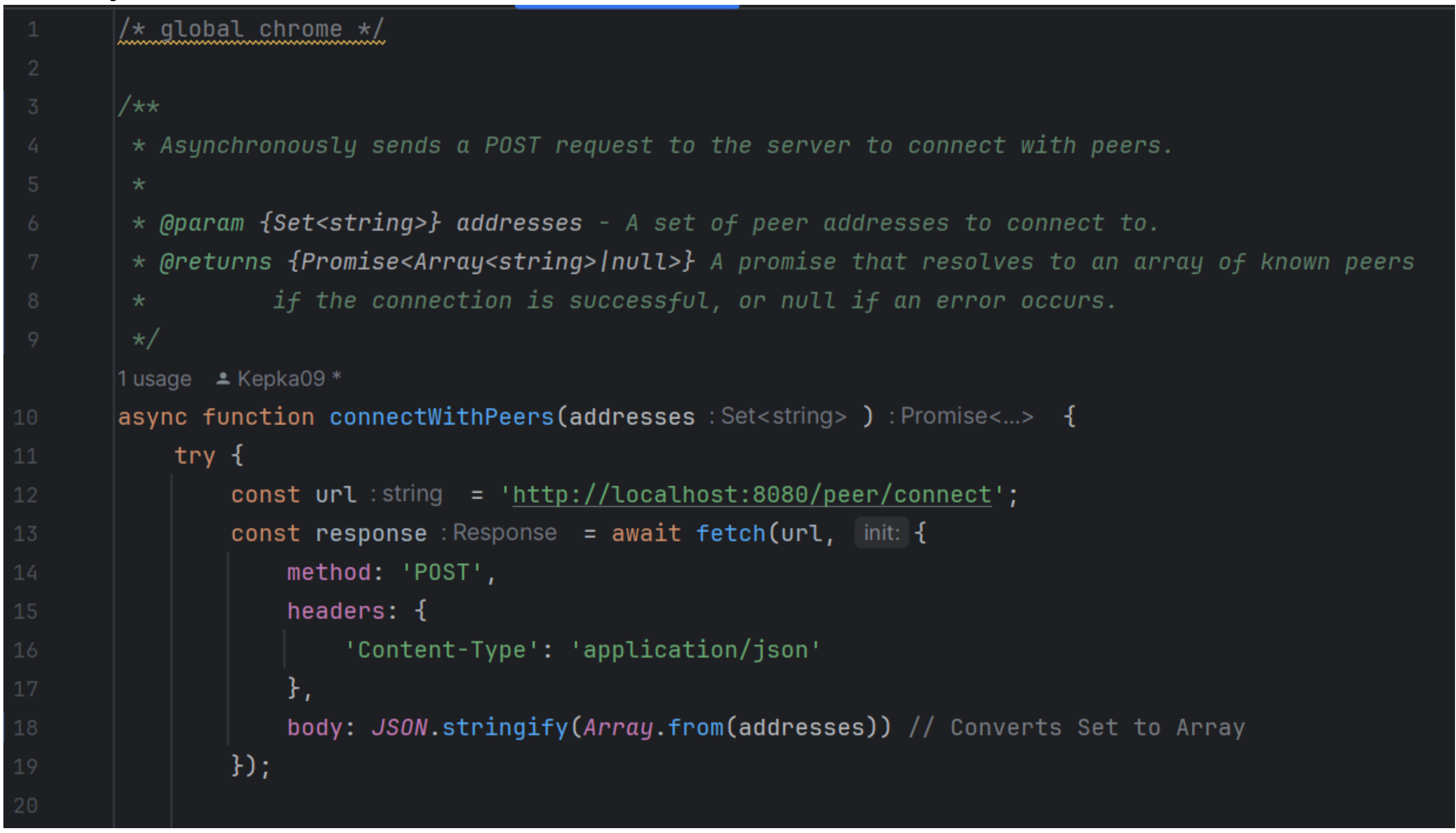


```java
            return ResponseEntity.ok()
                    .header( headerName: "Content-Disposition",  ...headerValues: "attachment; filename=\"" + file.getName()
                    .contentType(mediaType)
                    .contentLength(byteArrayResource.contentLength())
                    .body(byteArrayResource);
        } catch (Exception e) {
            logger.log(Level.SEVERE,  msg: "Exception occurred while downloading file", e);
            return ResponseEntity.internalServerError().build();
        }


    *
    Updates the name of a file on the current peer.

    @param oldName The current name of the file.
    @param newName The new name for the file.
    @return A ResponseEntity indicating the result of the operation.
    /
```

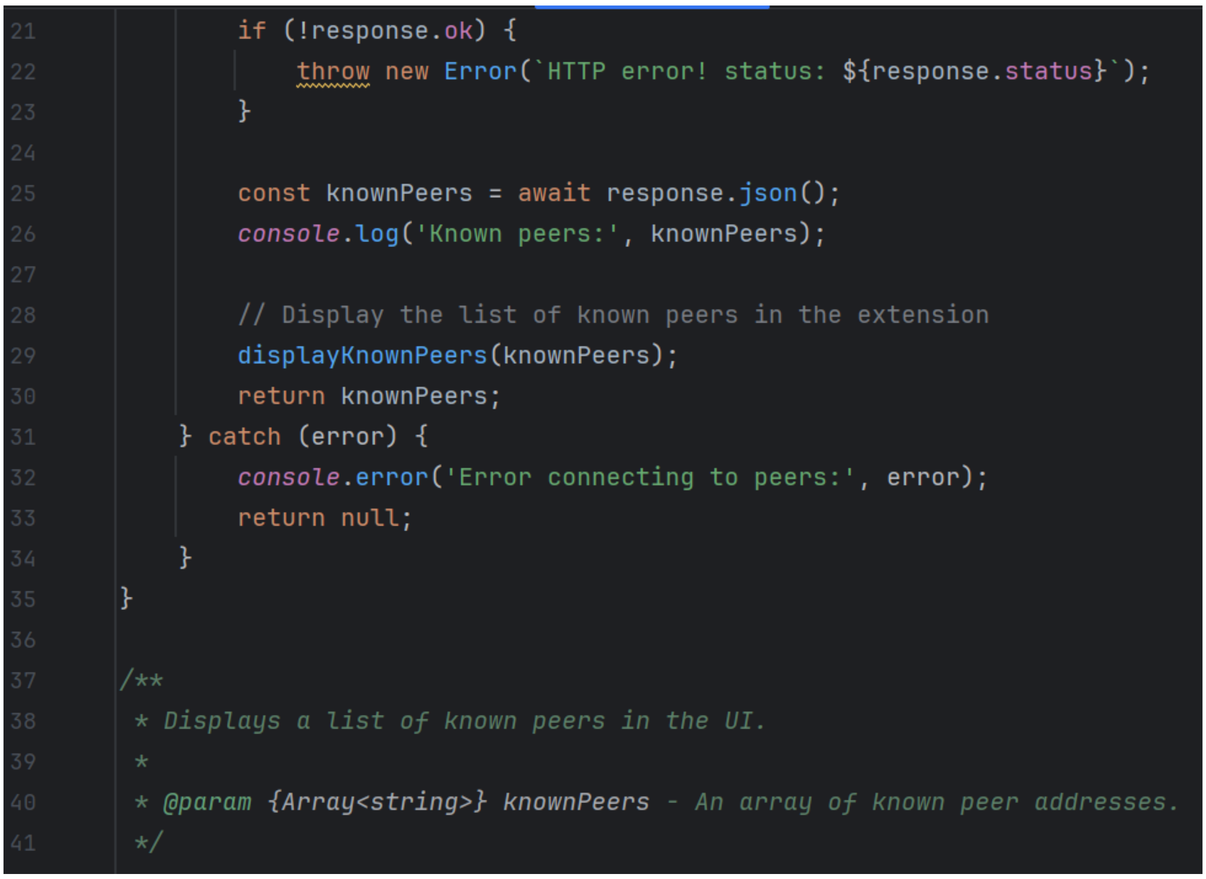


```java
    @PutMapping("/files/{address}")
    public ResponseEntity<String> updateFile(@RequestParam String oldName,
                                             @RequestParam String newName){
        //TODO тут обновить файл из списка файлов currentPeer
        //todo результат должен сразу показаться на фронте
        return new ResponseEntity<>( body: "File added successfully", HttpStatus.OK);
    }

    *
    Deletes a file from the current peer.

    @param fileName The name of the file to be deleted.
    @return A ResponseEntity indicating the result of the operation.
    /
    Kepka09
    @DeleteMapping("/files/{address}")
    public ResponseEntity<String> deleteFile (@RequestParam String fileName){
        //TODO тут типа файл  из списка файлов currentPeer
        //todo результат должен сразу показаться на фронте
        return new ResponseEntity<>( status: null);
    }
```

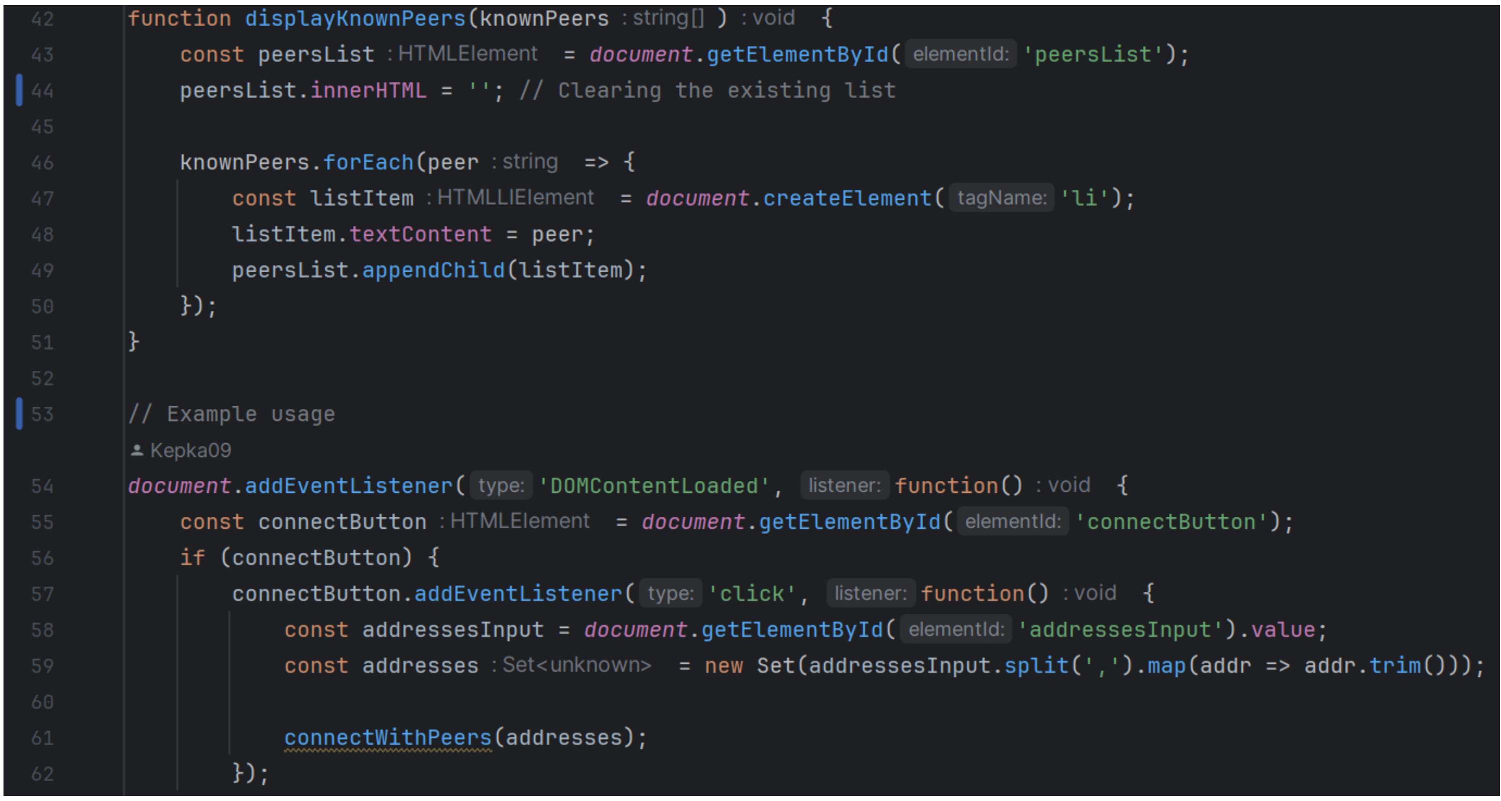
connect.js



```javascript
/* global chrome */

/**
 * Asynchronously sends a POST request to the server to connect with peers.
 *
 * @param {Set<string>} addresses - A set of peer addresses to connect to.
 * @returns {Promise<Array<string>|null>} A promise that resolves to an array of known peers
 *          if the connection is successful, or null if an error occurs.
 */
async function connectWithPeers(addresses) {
    try {
        const url = 'http://localhost:8080/peer/connect';
        const response = await fetch(url, {
            method: 'POST',
            headers: {
                'Content-Type': 'application/json'
            },
            body: JSON.stringify(Array.from(addresses)) // Converts Set to Array
        });

```

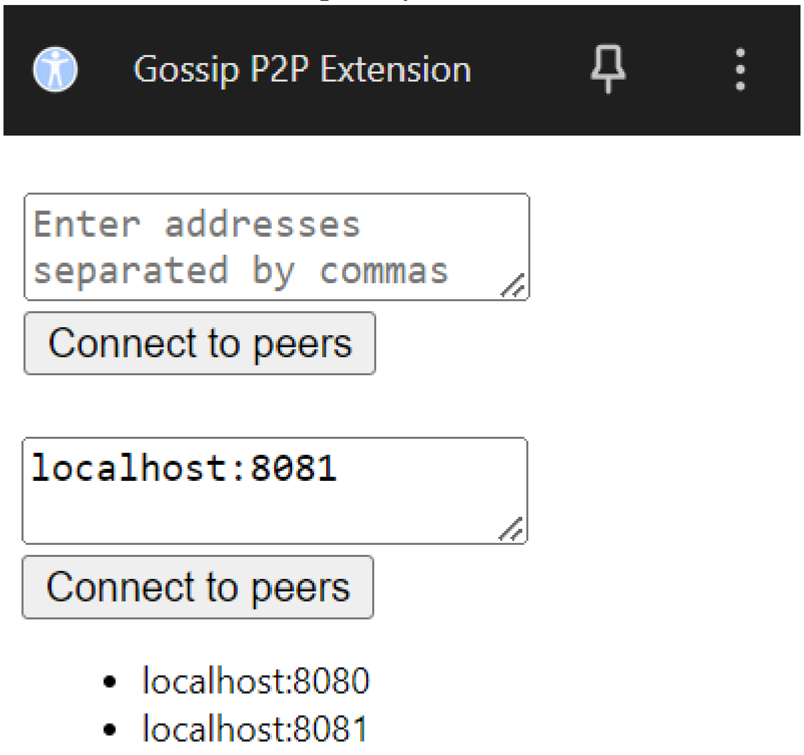


```javascript
        if (!response.ok) {
            throw new Error(`HTTP error! status: ${response.status}`);
        }

        const knownPeers = await response.json();
        console.log('Known peers:', knownPeers);

        // Display the list of known peers in the extension
        displayKnownPeers(knownPeers);
        return knownPeers;
    } catch (error) {
        console.error('Error connecting to peers:', error);
        return null;
    }
}

/**
 * Displays a list of known peers in the UI.
 *
 * @param {Array<string>} knownPeers - An array of known peer addresses.
 */
```



```
function displayKnownPeers(knownPeers :string[] ) :void  {
    const peersList :HTMLElement  = document.getElementById( elementId: 'peersList');
    peersList.innerHTML = ''; // Clearing the existing list

    knownPeers.forEach(peer :string  => {
        const listItem :HTMLLIElement  = document.createElement( tagName: 'li');
        listItem.textContent = peer;
        peersList.appendChild(listItem);
    });
}

// Example usage
Kepka09
document.addEventListener( type: 'DOMContentLoaded',  listener: function() :void  {
    const connectButton :HTMLElement  = document.getElementById( elementId: 'connectButton');
    if (connectButton) {
        connectButton.addEventListener( type: 'click',  listener: function() :void  {
            const addressesInput = document.getElementById( elementId: 'addressesInput').value;
            const addresses :Set<unknown>  = new Set(addressesInput.split(',').map(addr => addr.trim()));

            connectWithPeers(addresses);
        });
```

Реалізація мого P2P проекту



Gossip P2P Extension

```
Enter addresses
separated by commas
```

Connect to peers

```
localhost:8081
```

Connect to peers

- localhost:8080
- localhost:8081

Висновок: В даній лабораторній роботі була написана частина програми P2P“Download Manager”.