Національний технічний університет України
«Київський політехнічний інститут імені Ігоря Сікорського»
Факультет інформатики та обчислювальної техніки
Кафедра інформаційних систем та технологій

Лабораторна робота № 2

з дисципліни «Технології розроблення
програмного забезпечення»
Тема: «Основи проектування»

Виконав:
студент групи ІА-32
Задорожний Костянтин
Леонідович

Перевірив:
Мягкий Михайло
Юрійович

Київ - 2025

Тема: «Основи проектування»

Мета: Обрати зручну систему побудови UML-діаграм та навчитися будувати діаграми варіантів використання для системи що проєктується, розробляти сценарії варіантів використання та будувати діаграми класів предметної області.

Завдання:

• Ознайомитись з короткими теоретичними відомостями.

• Проаналізувати тему та спроєктувати діаграму варіантів використання відповідно до обраної теми лабораторного циклу.

• Спроєктувати діаграму класів предметної області.

• Вибрати 3 варіанти використання та написати за ними сценарії використання.

• На основі спроєктованої діаграми класів предметної області розробити основні класи та структуру бази даних системи. Класи даних повинні реалізувати шаблон Repository для взаємодії з базою даних.

• Нарисувати діаграму класів для реалізованої частини системи. • Підготувати звіт щодо виконання лабораторної роботи. Поданий звіт повинен містити: діаграму варіантів використання відповідно, діаграму класів системи, вихідні коди класів системи, а також зображення структури бази даних.

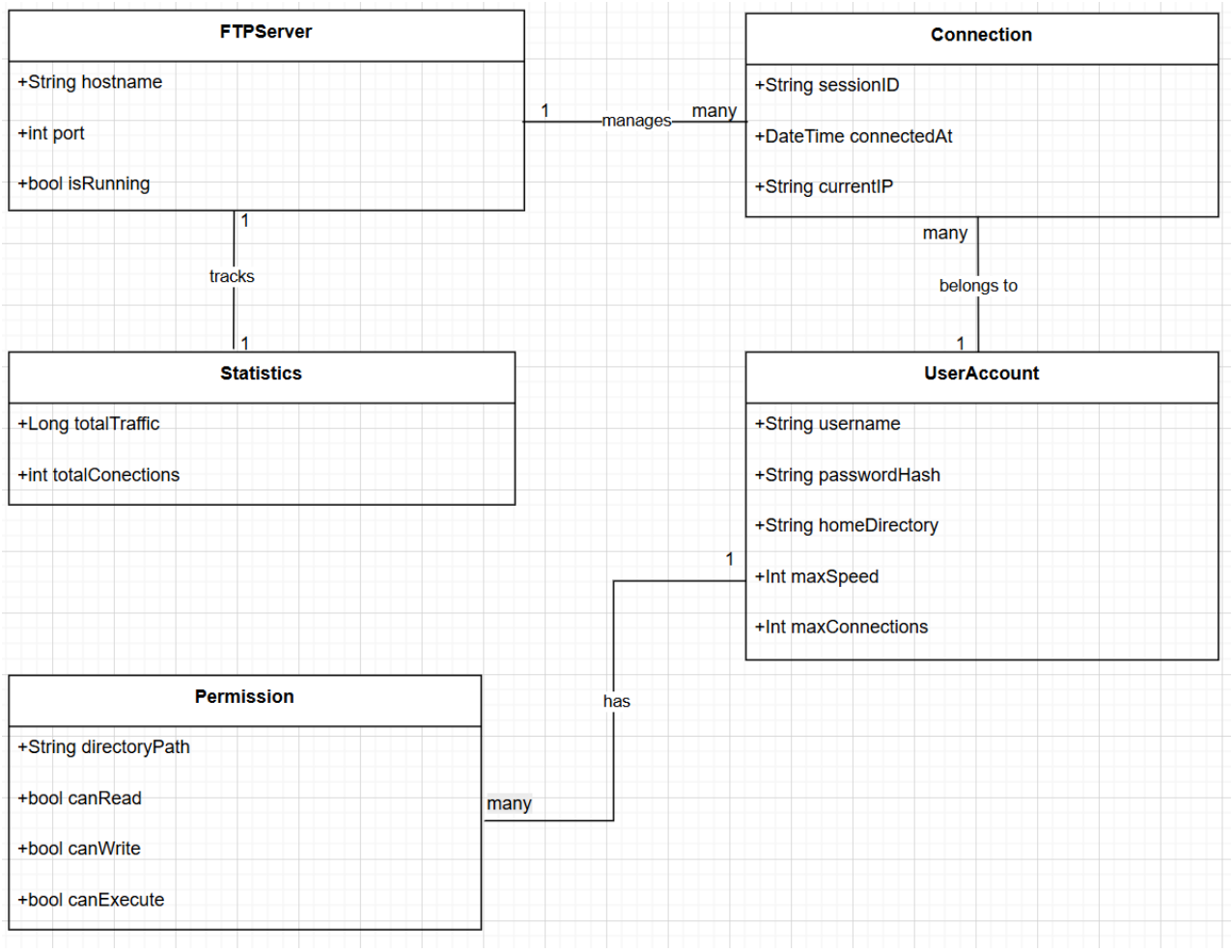
Предметна область:

22. FTP-server (state, builder, memento, template method, visitor, client-server) FTP-сервер повинен вміти коректно обробляти і відправляти відповіді п протоколу FTP, з можливістю створення користувачів (з паролями) і доступних їм папок, розподілу прав за стандартною схемою (rwe), ведення статистики з'єднань, обмеження максимальної кількості підключень і

максимальної швидкості поширення глобально і окремо для кожного облікового запису.

Хід роботи

Діаграма класів системи:



Ця діаграма відображає статичну структуру системи та зв'язки між сутностями без деталізації методів реалізації.

Основні сутності:

FtpServer: Головний клас, що керує з'єднаннями.

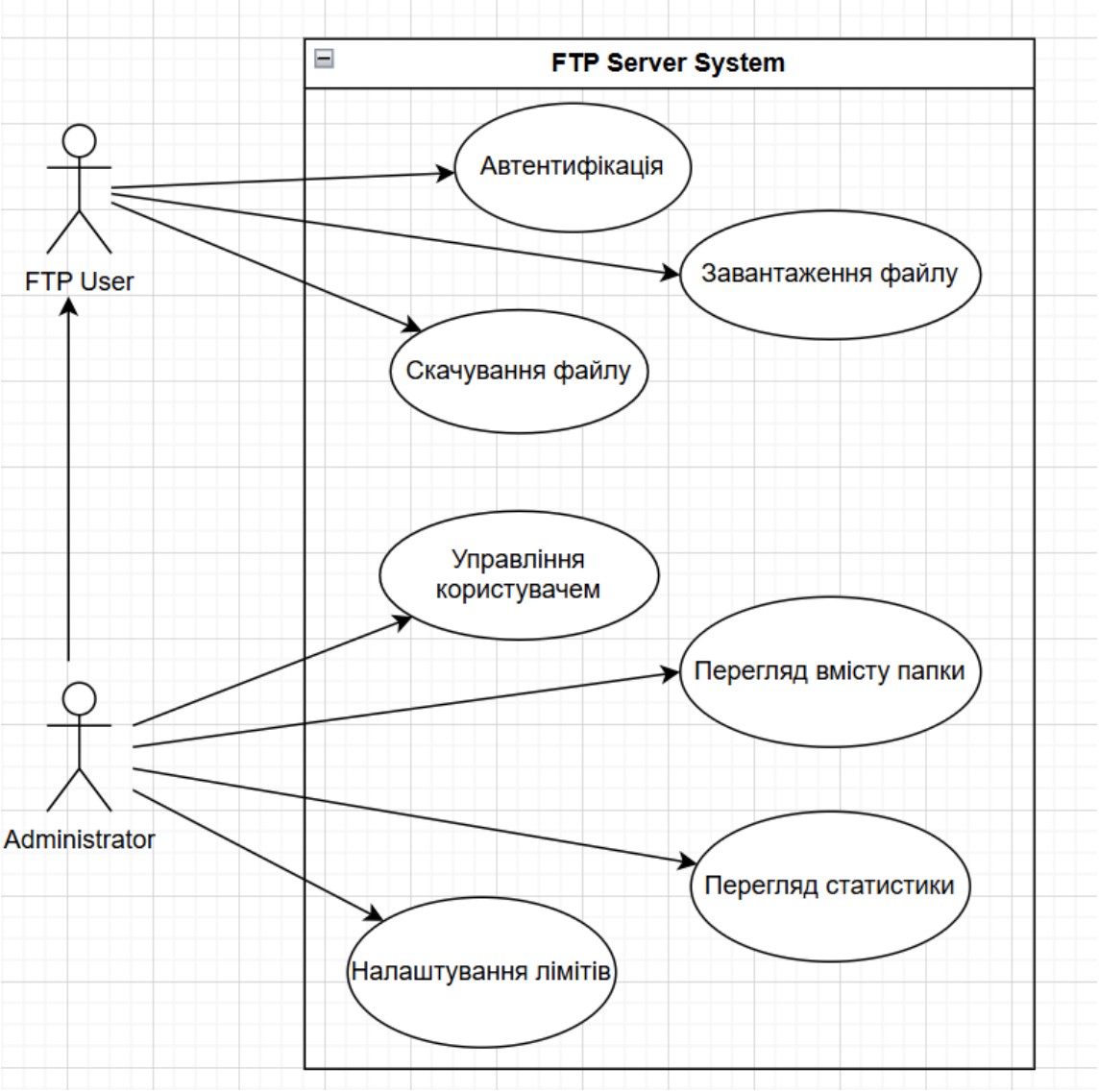
UserAccount: Обліковий запис користувача з налаштуваннями лімітів.

Permission: Права доступу (Read, Write, Execute).

Connection: Активна сесія користувача.

Statistics: Дані про активність.

Діаграма варіантів використання (Use Case Diagram):



Згідно з теоретичними відомостями, діаграма варіантів використання відображає вимоги до системи та взаємодію акторів з прецедентами.

Актори:

1. **FTP User:** Клієнт, що підключається до серверу для передачі файлів.
2. **Administrator:** Користувач з розширеними правами для управління сервером.

Основні варіанти використання:

- Автентифікація (вхід з паролем).
- Завантаження файлів (Upload) та Скачування файлів (Download).

- Управління файлами (List, Delete, Create Folder).
- Управління користувачами (Create User, Set Limits) — тільки Адміністратор.
- Перегляд статистики — тільки Адміністратор.

Сценарії Використання діаграми варіантів:

Сценарій 1: Автентифікація користувача

Актор: FTP User. Варіант використання: Автентифікація. Передумови: Сервер запущений, користувач має клієнтське ПЗ. Постумови: Створено сесію для користувача, доступ до файлової системи відкрито.
Основний потік подій:

1. Користувач ініціює з'єднання з сервером.
2. Сервер запитує ім'я користувача.
3. Користувач вводить ім'я (Username).
4. Сервер запитує пароль.
5. Користувач вводить пароль.
6. Сервер перевіряє наявність користувача в БД та коректність пароля.
7. Сервер перевіряє, чи не перевищено ліміт MaxConnections для цього користувача.
8. Сервер надсилає повідомлення про успішний вхід (код 230).

Винятки:

- Невірний логін/пароль: Сервер повертає помилку 530, з'єднання розривається.
- Перевищено ліміт з'єднань: Сервер повертає помилку 421, з'єднання розривається.

Сценарій 2: Завантаження файлу на сервер (Upload)

Актор: FTP User. Варіант використання: Завантаження файлу.

Передумови: Користувач автентифікований.

Основний потік подій:

1. Користувач надсилає команду STOR filename.ext.
2. Сервер перевіряє, чи має користувач право CanWrite для поточної директорії (частина вимог rwe).
3. Сервер відкриває канал передачі даних.
4. Сервер починає прийом даних, контролюючи швидкість згідно з MaxSpeed.
5. Після завершення передачі сервер зберігає файл на диск.
6. Сервер оновлює статистику (трафік).
7. Сервер надсилає підтвердження успішного завантаження (код 226).

Винятки:

- Відсутні права на запис: Сервер повертає помилку 550 ("Access denied").

Сценарій 3: Створення нового користувача (Адміністратором)

Актор: Administrator. Варіант використання: Створення користувача.

Передумови: Адміністратор увійшов у панель управління сервером.

Основний потік подій:

1. Адміністратор вибирає опцію "Додати користувача".
2. Система запитує параметри: логін, пароль, домашню папку.
3. Адміністратор вводить дані та встановлює ліміти (швидкість, к-сть з'єднань).
4. Адміністратор налаштовує права доступу (rwe) для домашньої папки.
5. Адміністратор підтверджує створення.
6. Система зберігає нового користувача в Базу Даних через репозиторій.

7. Система повідомляє про успішне створення.

Вихідні коди класів системи на мові Java:

FtpUser.java

```java
import java.util.ArrayList;
import java.util.List;

public class FtpUser {
    private Long id;
    private String username;
    private String passwordHash;
    private String homeDirectory;

    private int maxSpeedKbps;
    private int maxConnectionCount;

    private List<UserPermission> permissions = new ArrayList<>();

    public FtpUser(String username, String passwordHash, String homeDirectory, int maxSpeedKbps,
            int maxConnectionCount) {
        this.username = username;
        this.passwordHash = passwordHash;
        this.homeDirectory = homeDirectory;
        this.maxSpeedKbps = maxSpeedKbps;
        this.maxConnectionCount = maxConnectionCount;
    }

    public void addPermission(UserPermission permission) {
        this.permissions.add(permission);
    }
```

UserPermission.java

```java
public class UserPermission {
    private Long id;
    private Long userId;
    private String directoryPath;

    private boolean canRead;
    private boolean canWrite;
    private boolean canExecute;

    public UserPermission(String directoryPath, boolean canRead, boolean canWrite, boolean canExecute) {
        this.directoryPath = directoryPath;
        this.canRead = canRead;
        this.canWrite = canWrite;
        this.canExecute = canExecute;
    }
```

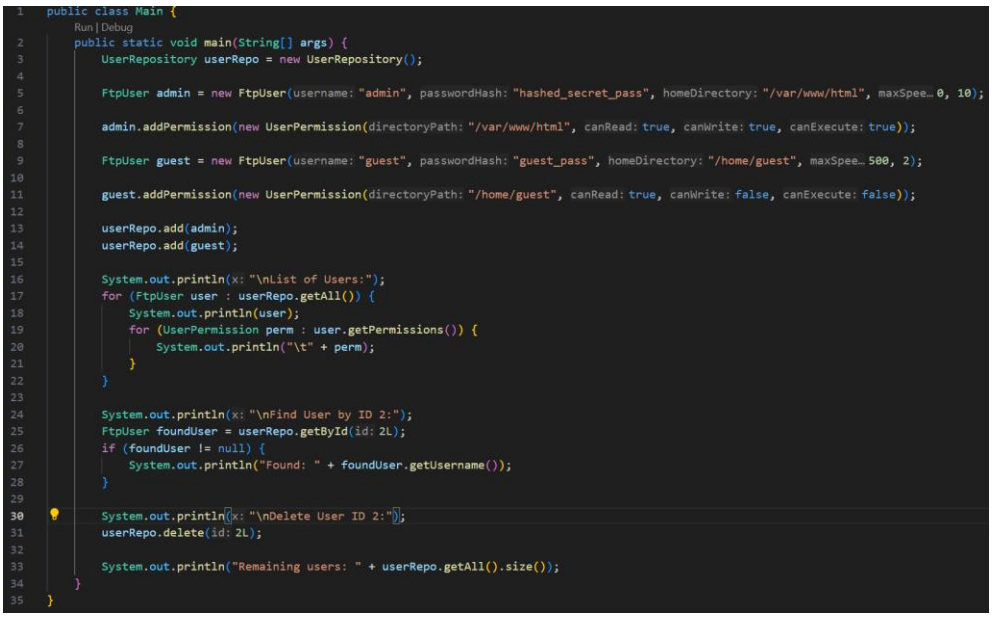
Repository.java

```java
import java.util.List;

public interface Repository<T> {
    List<T> getAll();

    T getById(Long id);

    void add(T entity);

    void update(T entity);

    void delete(Long id);
}
```

UserRepository.java

```java
import java.util.ArrayList;
import java.util.List;

public class UserRepository implements Repository<FtpUser> {
    private static List<FtpUser> users = new ArrayList<>();
    private static Long idCounter = 1L;

    @Override
    public List<FtpUser> getAll() {
        return new ArrayList<>(users);
    }

    @Override
    public FtpUser getById(Long id) {
        return users.stream()
                .filter(u -> u.getId().equals(id))
                .findFirst()
                .orElse(other: null);
    }

    public FtpUser getByUsername(String username) {
        return users.stream()
                .filter(u -> u.getUsername().equals(username))
                .findFirst()
                .orElse(other: null);
    }

    @Override
    public void add(FtpUser entity) {
        entity.setId(idCounter++);
        for (UserPermission p : entity.getPermissions()) {
            p.setUserId(entity.getId());
        }
        users.add(entity);
        System.out.println("User added: " + entity.getUsername());
    }
```

Main.java



```java
public class Main {
    Run | Debug
    public static void main(String[] args) {
        UserRepository userRepo = new UserRepository();

        FtpUser admin = new FtpUser(username: "admin", passwordHash: "hashed_secret_pass", homeDirectory: "/var/www/html", maxSpee… 0, 10);

        admin.addPermission(new UserPermission(directoryPath: "/var/www/html", canRead: true, canWrite: true, canExecute: true));

        FtpUser guest = new FtpUser(username: "guest", passwordHash: "guest_pass", homeDirectory: "/home/guest", maxSpee… 500, 2);

        guest.addPermission(new UserPermission(directoryPath: "/home/guest", canRead: true, canWrite: false, canExecute: false));

        userRepo.add(admin);
        userRepo.add(guest);

        System.out.println(x: "\nList of Users:");
        for (FtpUser user : userRepo.getAll()) {
            System.out.println(user);
            for (UserPermission perm : user.getPermissions()) {
                System.out.println("\t" + perm);
            }
        }

        System.out.println(x: "\nFind User by ID 2:");
        FtpUser foundUser = userRepo.getById(id: 2L);
        if (foundUser != null) {
            System.out.println("Found: " + foundUser.getUsername());
        }

        System.out.println(x: "\nDelete User ID 2:");
        userRepo.delete(id: 2L);

        System.out.println("Remaining users: " + userRepo.getAll().size());
    }
}
```

Висновки

Висновки: Я навчився будувати діаграми варіантів використання для системи що проєктується, розробив сценарії варіантів використання та побудував діаграми класів предметної області.