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

ТРПЗ

Тема: Патерни проектування. HTTP-сервер

Виконав:
студент групи ІА-31
Машин Д. І.

Київ 2025

Мета: Вивчити структуру шаблонів «Composite», «Flyweight» (Пристосуванець), «Interpreter», «Visitor» та навчитися застосовувати їх в реалізації програмної системи.

Завдання

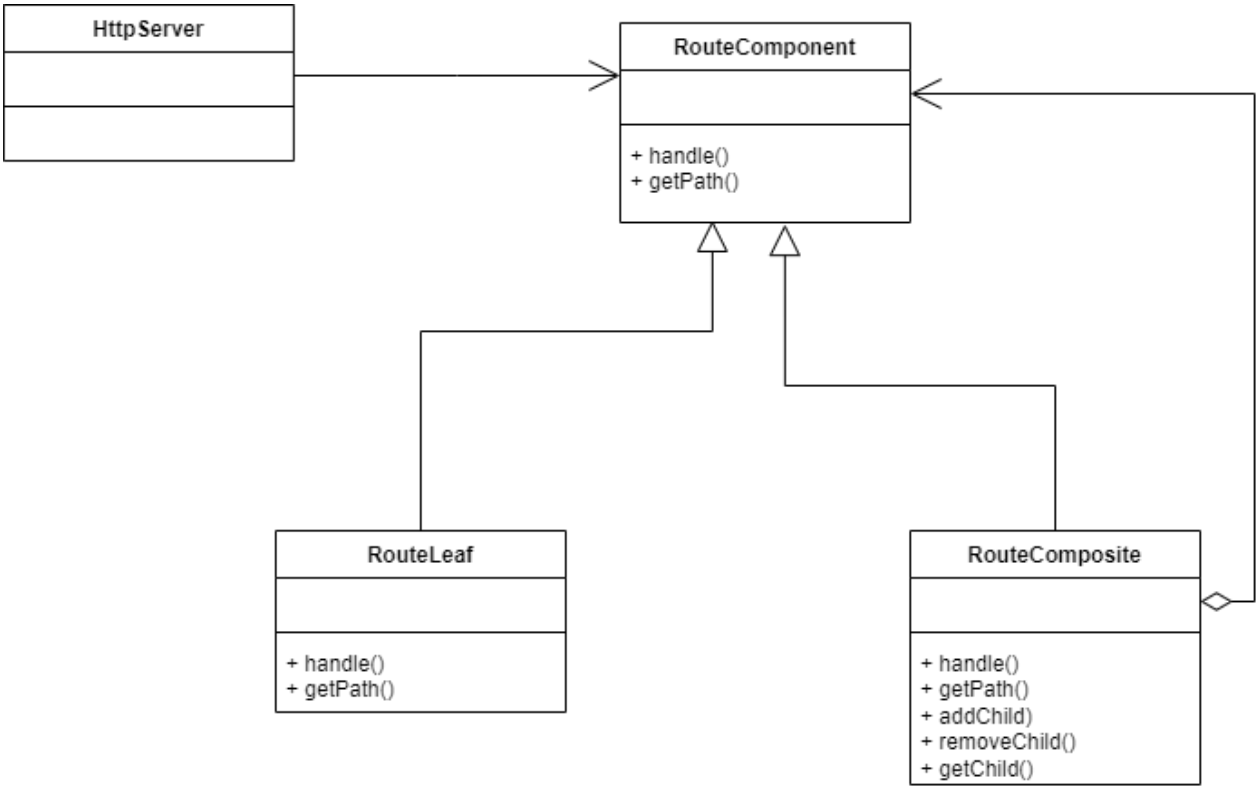
• Ознайомитись з короткими теоретичними відомостями.

• Реалізувати частину функціоналу робочої програми у вигляді класів та їхньої взаємодії для досягнення конкретних функціональних можливостей.

• Реалізувати один з розглянутих шаблонів за обраною темою.

• Реалізувати не менше 3-х класів відповідно до обраної теми.

• Підготувати звіт щодо виконання лабораторної роботи. Поданий звіт повинен містити: діаграму класів, яка представляє використання шаблону в реалізації системи, навести фрагменти коду по реалізації цього шаблону.



```java
public interface RouteComponent {
    2 usages  2 implementations
    public HttpResponse handle(HttpRequest request);
    4 usages  2 implementations
    public String getPath();
}
```

```java
public class RouteComposite implements RouteComponent{
    5 usages
    List<RouteComponent> componentList = new ArrayList<>();
    4 usages
    String path;
    1 usage
    public RouteComposite(String path){
        this.path=path;
    }
    2 usages
    @Override
    public HttpResponse handle(HttpRequest request) {
        String requestPath = request.getPath();

        if (!requestPath.startsWith(path)) {
            return null;
        }


        String remainingPath = requestPath.substring(path.length());
        for (RouteComponent component : componentList) {
            if (remainingPath.startsWith(component.getPath())) {
                return component.handle(new HttpRequest(request.getMethod(),remainingPath, request.getHeaders(), request.getBody()));
            }
        }

        return null;
    }
```

```java
4 usages
@Override
public String getPath() { return path; }
```

```java
1 usage
public void addChild(RouteComponent component){
    if(componentList.stream().anyMatch(c->c.getPath()==component.getPath())){
        throw new IllegalArgumentException("Path is already taken");
    }
    componentList.add(component);
}
no usages
public void removeChild(RouteComponent component) { componentList.remove(component); }
no usages
public RouteComponent getChild(String path){
    return componentList.stream().filter(c->c.getPath()==path).findFirst().orElse( other: null);

}
```

```java
public class RouteLeaf implements RouteComponent{
    3 usages
    String path;
    3 usages
    HttpRequestCallback callback;

    1 usage
    public RouteLeaf(String path, HttpRequestCallback callback) {
        this.path = path;
        this.callback = callback;
    }

    2 usages
    @Override
    public HttpResponse handle(HttpRequest request) {
        if(callback==null || !request.getPath().replaceAll( regex: "/", replacement: "").equals(path.replaceAll( regex: "/", replacement: "")))
            return null;
        }
        return  callback.execute(request);
    }

    4 usages
    @Override
    public String getPath() {
        return path;
    }
}
```