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 №6

ТРПЗ

Тема: Патерни проектування. HTTP-сервер

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

Київ 2025

Мета: Вивчити структуру шаблонів «Abstract Factory», «Factory Method», «Memento», «Observer», «Decorator» та навчитися застосовувати їх в реалізації програмної системи.

Завдання

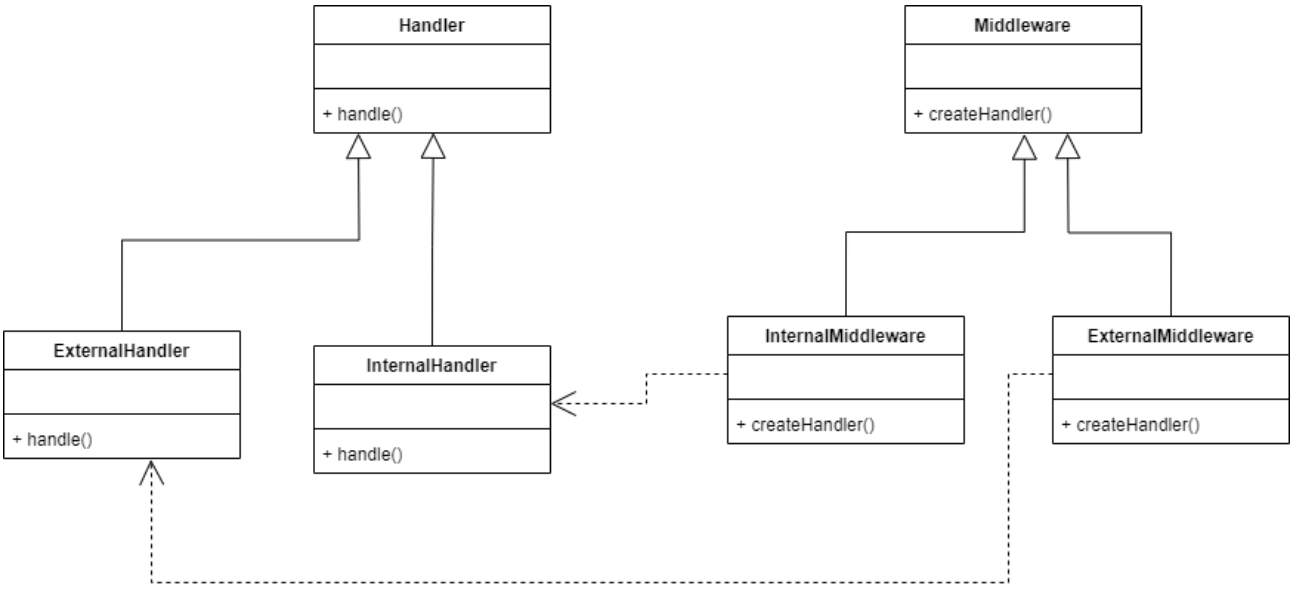
• Ознайомитись з короткими теоретичними відомостями.

• 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.




```
7 usages   2 implementations
public interface Handler {
    1 usage   2 implementations
    public HttpResponse handle(HttpRequest request);
}
```

```java
4 usages  2 inheritors
public abstract class Middleware {
    1 usage  2 implementations
    public abstract Handler createHandler();

}
```

```java
public class InternalHandler implements Handler {
    4 usages
    private static final Logger logger = Logger.getGlobal();

    1 usage
    FileHandler fileHandler = new FileHandler();

    3 usages
    Map<String, HttpRequestCallback> routes;
    1 usage
    public InternalHandler(Map<String, HttpRequestCallback> routes){
        this.routes=routes;
    }
    1 usage
    @Override
    public HttpResponse handle(HttpRequest request) {
        HttpResponse response;
        logger.info( msg: "Handling request for path: " + request.getPath());
        String path = request.getPath();
        if (routes.containsKey(path)) {
            HttpRequestCallback callback = routes.get(path);
            response = callback.execute(request);
            logger.info( msg: "Route found for path: " + path);
        } else {
            if(path.equals("/")) {
                path = "/index.html";
            }
            try{
            String fileContent = fileHandler.readFile( path: "src/main/resources" + path);
            logger.info( msg: "File found for path: " + path);
            response = HttpResponseDirector.Ok(fileContent, request.getHeaders());}
            catch (FileNotFoundException fileNotFoundException) {
                logger.warning( msg: "File not found for path: " + path);
                response = HttpResponseDirector.NotFound(request.getHeaders());
            }

        }
        return response;
    }
}
```

```java
1 usage
public class ExternalHandler implements Handler{
    3 usages
    private static final Logger logger = Logger.getGlobal();
    1 usage
    HttpResponseParser parser = new HttpResponseParser();
    1 usage
    @Override
    public HttpResponse handle(HttpRequest request) {
        ServerDTO server = HttpServer.requestAvailableServer();
        if(server == null){
            logger.warning( msg: "No server available to redirect request.");
            return null;
        }
        logger.info( msg: "Redirecting request ...");
        try(Socket socket = new Socket(server.getHost(), server.getPort());
            DataOutputStream out = new DataOutputStream(socket.getOutputStream())) {
            out.writeUTF(request.toString());
            out.flush();
            out.close();
            //Statistics.saveExternalStats();
            return parser.parse(socket.getInputStream());

        } catch (IOException e) {
            logger.severe( msg: "Error redirecting request: " + e.getMessage());
            return null;
        }
    }
}
```

```java
1 usage
public class InternalMiddleware extends Middleware{
    2 usages
    Map<String, HttpRequestCallback> routes;
    1 usage
    public InternalMiddleware(Map<String, HttpRequestCallback> routes){
        this.routes = routes;
    }
    1 usage
    @Override
    public Handler createHandler() { return new InternalHandler(routes); }
}

```

```java
1 usage
public class ExternalMiddleware extends Middleware{

    1 usage
    @Override
    public Handler createHandler() {
        return new ExternalHandler();
    }
}

```