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 №4

ШАБЛОНИ «SINGLETON», «ITERATOR», «PROXY», «STATE», «STRATEGY»

Варіант 4

Виконав
студент групи ІА – 13:
Запотоцький І.А

Перевірив:
М’який М.Ю

Завдання:

1. Ознайомитися з короткими теоретичними відомостями.
2. Реалізувати частину функціоналу робочої програми у вигляді класів та їхньої взаємодії для досягнення конкретних функціональних можливостей.
3. Застосування одного з розглянутих шаблонів при реалізації програми.

Варіант:

4)
Графічний редактор (proxy, prototype, decorator, bridge, flyweight, SOA) Графічний редактор повинен вміти створювати / редагувати растрові (або векторні на розсуд студента) зображення в 2-3 основних популярних форматах (bmp, png, jpg), мати панель інструментів для створення графічних примітивів, вибору кольорів, нанесення тексту, додавання найпростіших візуальних ефектів (ч/б растр, інфрачервоний растр, 2-3 на вибір учня), роботи з шарами.

Хід роботи

1.Шаблон “Proxy”

Шаблон Proxy (проксі) є структурним шаблоном проектування, який дозволяє створити замісника або обгортку для іншого об'єкта. Цей шаблон використовується для контролю доступу до об'єкта, для надання додаткової логіки при викликах методів об'єкта або для забезпечення віддаленого доступу до об'єкта.

Основні учасники шаблону Proxy:

Subject (Суб'єкт): Інтерфейс, який визначає загальні методи для реального об'єкта та його проксі.

RealSubject (Реальний суб'єкт): Реальний об'єкт, для якого створюється проксі. Він реалізує інтерфейс Subject.

Proxy (Проксі): Клас, який діє як замісник або обгортка для реального об'єкта. Цей клас має той самий інтерфейс, що і RealSubject, тобто реалізує Subject.

Проксі може мати різні види, такі як:

Віддалений проксі (Remote Proxy): Надає локальному викликаючому доступ до об'єкта, який знаходиться в іншому просторі.

Віртуальний проксі (Virtual Proxy): Керує створенням ресурсоєм, що є дорогим у вигляді реального об'єкта. Наприклад, завантаження важких зображень лише при необхідності.

Захистний проксі (Protection Proxy): Контролює доступ до об'єкта, визначаючи, чи користувач має необхідні права.

Локальний проксі (Cache Proxy): Зберігає копію результатів викликів об'єкта для уникнення повторних обчислень.

```java
import ...

2 usages
@Service
public class UserServiceImpl implements UserService{

    11 usages
    private final UserRepository userRepository;

    1 usage
    @Autowired
    public UserServiceImpl(UserRepository repo) { this.userRepository = repo; }

    2 usages
    @Override
    public ResponseEntity<List<UserDTO>> findAllUsers() {
        return new ResponseEntity<>(this.userRepository.findAll().stream().map(this::userDtoMapper).collect(Collectors.toList())
                ,HttpStatus.OK);
    }

    2 usages
    @Override
    public ResponseEntity<?> deleteUser(Long id) {
        if(this.userRepository.findById(id).isPresent()) {
            this.userRepository.deleteById(id);
            return new ResponseEntity<>(HttpStatus.OK);
        }else {
            return new ResponseEntity<>( body: "User with this id does not exist",HttpStatus.BAD_REQUEST);
        }
    }

    2 usages
    @Override
    public ResponseEntity<?> getUser(Long id) {
        if(this.userRepository.findById(id).isPresent()){
            User user = this.userRepository.findById(id).get();
            return new ResponseEntity<UserDTO>(UserDTO.builder().
                    id(user.getId()).
                    login(user.getLogin()).
                    build(), HttpStatus.OK);
        }else {
            return new ResponseEntity<>( body: "User with this id does not exist",HttpStatus.BAD_REQUEST);
        }

    }

    1 usage
    private UserDTO userDtoMapper(User user){
        return UserDTO.builder().
                id(user.getId()).
                login(user.getLogin()).
                build();
    }
```

```java
@Service
public class ProxyStatusCheckerUserService implements UserService{


    private static final Map<Long, User> users = new HashMap<>();

    2 usages
    private UserRepository userRepository;
    4 usages
    private final BannedUserRepository bannedUsersRepository;

    public ProxyStatusCheckerUserService(UserRepository userRepository, BannedUserRepository bannedUsersRepository) {
        this.userRepository = userRepository;
        this.bannedUsersRepository = bannedUsersRepository;
    }

    5 usages
    UserServiceImpl service = new UserServiceImpl(userRepository);

    2 usages
    @Override
    public ResponseEntity<List<UserDTO>> findAllUsers() { return service.findAllUsers(); }

    2 usages
    @Override
    public ResponseEntity<?> deleteUser(Long id) {
        if(bannedUsersRepository.findByUser_id(id).isPresent()){
            return new ResponseEntity<>( body: "User with this id is banned from being edited.", HttpStatusCode.valueOf( code: 406));
        }
        return service.deleteUser(id);

    }
```

```java
2 usages
@Override
public ResponseEntity<?> updateUser(User user, Long id) {
    if(bannedUsersRepository.findByUser_id(id).isPresent()){
        return new ResponseEntity<>( body: "User with this id is banned from being edited.", HttpStatusCode.valueOf( code: 406));
    }
    return service.updateUser(user,id);
}

2 usages
@Override
public ResponseEntity<?> createUser(User user) {
    return service.createUser(user);
}

2 usages
@Override
public ResponseEntity<?> getUser(Long id) {
    if(bannedUsersRepository.findByUser_id(id).isPresent()){
        return new ResponseEntity<>( body: "User with this id is banned from being edited or retrieved.", HttpStatusCode.valueOf( code: 406));
    }
    return service.getUser(id);
}
```

В цьому випадку Proxy використовується як проміжний клас, який перевіряє, чи юзер, з яким ми починаємо працювати не є забаненим, а тільки після віддаємо його у основний клас UserService.