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

Теорія Розробки Програмного Забезпечення

Шаблон «Flyweight»

Виконав

студент групи ІА-14:

Фіалківський І.О.

Перевірив:

Мягкий М.Ю.

Київ 2023

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

Виконання

Завданням на лобораторну було реалізувати шаблон ***flyweight***.

Шаблон проектування "Flyweight" входить в категорію структурних шаблонів і використовується для оптимізації роботи з великою кількістю схожих об'єктів. Основна ідея полягає в тому, щоб спільно використовувати однакові частини об'єктів, замість того щоб кожен об'єкт зберігав свої власні копії цих частин. Це дозволяє економити пам'ять і прискорює виконання програми.

У шаблоні "Flyweight" виділяють два типи об'єктів: внутрішній та зовнішній. Внутрішні об'єкти є незмінними та можуть бути використані багатьма зовнішніми об'єктами. Зовнішні об'єкти, у свою чергу, зберігають лише ті дані, які вони спільно використовують, вказуючи на відповідний внутрішній об'єкт. Це сприяє зменшенню обсягу пам'яті, що використовується, і поліпшує продуктивність програми.

Шаблон "Flyweight" особливо ефективний у випадках, коли велика кількість об'єктів має спільні атрибути, і може бути використаний для оптимізації різних аспектів програмного коду, таких як графічні інтерфейси, обробка текстової інформації чи мережеві операції.

Прикладом реалізації в моєму проекті може бути клас Task.

```
@Entity
@Table(name = "task")
public class Task {

    @Id
    @GeneratedValue(strategy = GenerationType.IDENTITY)
    @Column(name = "id")
    private int id;

    @NotBlank
    @Size(max = 100, message = "Less then 100")
    @Column(name = "title")
    private String title;

    @NotBlank
    @Size(max = 200, min = 10, message = "From 10 to 200 chars")
    @Column(name="description")
    private String description;

    @Enumerated(EnumType.ORDINAL)
    @Column(name = "status")
    private Status status;
```

```
    @NotNull
    @Temporal(TemporalType.TIMESTAMP)
    @Column(name = "deadline")
    private LocalDateTime deadline;

    @Column(name = "priority")
    private int priority;

    @ManyToOne
    @JoinColumn(name = "project_id", referencedColumnName = "id")
    private Project projectStorage;

    @OneToMany(mappedBy = "taskStorage", cascade = CascadeType.REMOVE)
    private List<File> fileList;

    public Task(int id) {
        this.id = id;
    }
}
```

Поле Status у класі Task є прикладом використання паттерну Flyweight через використання перерахування (Enum). Перерахування вже вбудовано в концепцію Flyweight, оскільки воно обмежує та стандартизує набір можливих значень, і кожен елемент перерахування відновлюється лише один раз.

Паттерн Flyweight використовується для оптимізації використання пам'яті шляхом витягування спільних частин об'єктів із багатьох екземплярів. У випадку перерахувань, як Status, ця спільність полягає в тому, що кожне унікальне значення перерахування відоме лише один раз і може бути використане в усіх екземплярах класу.

Це дозволяє економити пам'ять, оскільки для кожного екземпляра класу Task не потрібно зберігати окремий об'єкт Status. Замість цього, усі екземпляри можуть посилатися на один і той же об'єкт перерахування Status, що призводить до зменшення використання пам'яті і покращення продуктивності програми.

```
package com.example.entity.enums;

public enum Status {
    PENDING,  IN_PROCESS, FINISHED
}
```

Висновок: В даній лабораторній роботі я реалізував частину проекту використавши шаблон проектування “Flyweight ”