

МIНIСТЕРСТВО ОСВIТИ І НАУКИ УКРАЇНИ

НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ

“КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ”

Факультет прикладної математики

Кафедра програмного забезпечення комп’ютерних систем

**Лабораторна робота № 1**

з дисципліни “ Об’єктно-орієнтоване програмування ”

тема “Реалізація структурних шаблонів проектування.”

|  |  |  |
| --- | --- | --- |
| Виконала  студентка II курсу  групи КП-72  Городченко Анна Володимирівна (*прізвище, ім’я, по батькові*) |  | Перевірив  “\_\_\_\_” “\_\_\_\_\_\_\_\_\_\_\_\_” 2019  р.  викладач  Заболотня Тетяна Миколаївна  (*прізвище, ім’я, по батькові*) |

Київ 2019

**Мета роботи**

Ознаойомлення з основними характеристиками шаблонів «Декоратор», «Компонувальник» та «Міст», запам’ятовування поширених ситуацій, коли використання цих шаблонів є доцільним, набуття вмінь та навичок реалізації шаблонів під час створення програмного коду.

**Постановка завдання**

1. Програмний текстовий чат версії 1.0 підтримує виведення

тексту повідомлення у віконці діалогу з іншим користувачем у звичайному

вигляді (шрифт 14 кегль, Times New Roman, чорний). На основі вже

існуючого движка чату реалізувати виведення тексту заданим кольором,

типом шрифта та розміром. Також на початку повідомлення додати час

його надходження.

2. За допомогою шаблона проектування реалізувати механізм

збирання коштів на ремонт школи. Кожен класний керівник збирає кошти

з батьків свого класу. Кожний завуч збирає гроші з керівників старших,

середніх та молодших класів. Директор збирає гроші із завучів.

**Коротке обгрунтування вибору шаблону проектування**

***Завдання 1 (Декоратор):***

Структурний шаблон «Декоратор» використовувала, для того щоб:

– обов’язки та поведінка об’єктів додавались динамічно;

– конкретні реалізації повинні бути відокремлені від обов'язків і поведінки;

– підкреслити те, що розширення об’єктів шляхом створення підкласів недоцільне або неможливе.

– при наявності великої кількості малих об’єктів, які мають схожу реалізацію.

– для реалізації можливостей, які потрібні не всім об’єктам та не завжди і можуть бути видалені при необхідності.

***Завдання 2 (Компонувальник):***

Структурний шаблон «Компонувальник» використовувала, для того коли:

– у наявності є оригінальна структура, що складається з об’єктів та композицій об’єктів;

– необхідно забезпечити ігнорування клієнтом істотних відмінностей між окремим об’єктом та складеним об’єктом;

– необхідно реалізувати уніфіковану обробку всіх об’єктів.

Але також можна розглядати і такі варіанти:

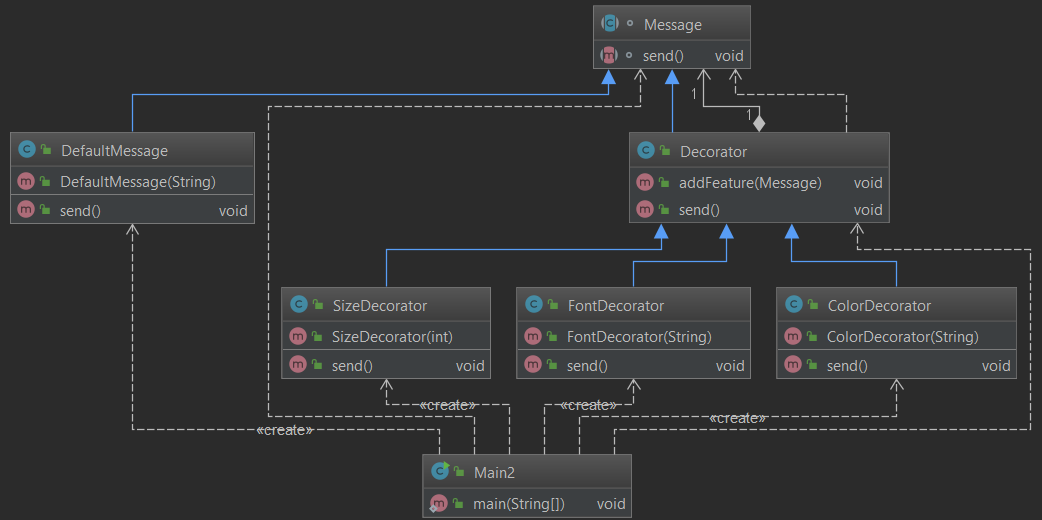
– шаблон «Декоратор», який містить операції типу Додати (Add), Видалити (Remove) та Знайти (Find)

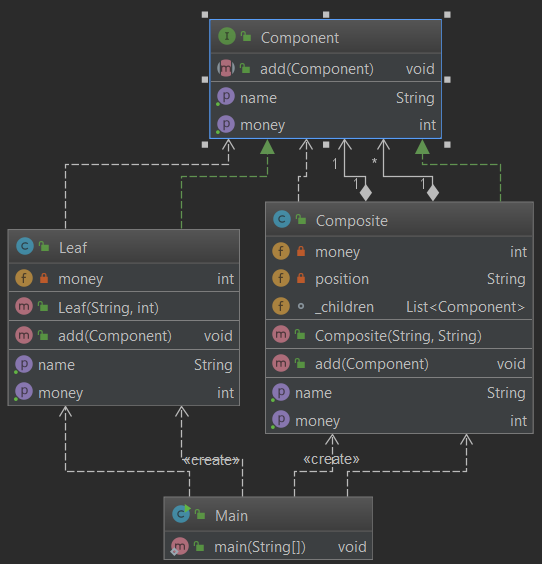
– шаблон «Пристосуванець» для розділення компонентів за умови, що характеристика «місцезнаходження» може бути проігнорована і всі операції будуть починатися з кореневої вершини композиції об’єктів

– шаблон «Відвідувач» для локалізації операцій, які на даний момент розподілені між класами Composite та Component

**Діаграма класів**

(Завдання 1)

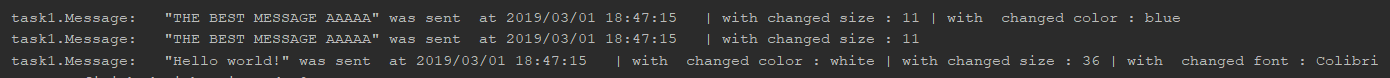


(Завдання 2)

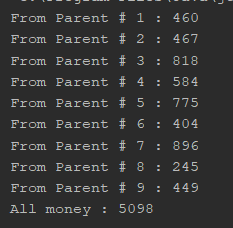
|  |
| --- |
| Component.java (task2) |
| package task2;  public interface Component {  void add(Component component);  int getMoney();  String getName(); } |
| Composite.java(task2) |
| package task2;  import java.util.ArrayList; import java.util.List;  public class Composite implements Component {   private String name;  private int money;  private String position;   List<Component> \_children = new ArrayList<Component>();   public Composite(String position, String name) {  this.name = name;  this.position = position;  }   @Override  public void add(Component component) {  \_children.add(component);  }   @Override  public int getMoney() {  this.money = 0;  for (Component component : \_children) {  this.money += component.getMoney();  }  return this.money;  }   @Override  public String getName() {  return this.name;  } } |

**Приклади результатів**

Завдання 1:



Завдання 2:



**Висновки**

Виконавши дану лабораторну роботу, я ознайомилась з основними характеристиками шаблонів «Міст», «Декоратор», «Компонувальник», запам’ятовування поширених ситуацій, коли використання цих шаблонів є доцільним, набуття вмінь та навичок реалізації шаблонів під час створення програмного коду.