



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
**Технологія розроблення програмного
забезпечення**

«COMPOSITE», «FLYWEIGHT», «INTERPRETER», «VISITOR»
«Застосування шаблону «COMPOSITE» в темі Download
Manager»
Варіант 26

Виконала
студентка групи ІА-13
Танасієнко Анастасія Вячеславівна

Київ 2023р.

Мета: дослідження шаблонів «COMPOSITE», «FLYWEIGHT», «INTERPRETER», «VISITOR» реалізація шаблону проектування «COMPOSITE»

Шаблон проектування «COMPOSITE»

Шаблон проектування "Composite" відноситься до структурних патернів і використовується для об'єднання об'єктів в структури деревоподібної ієрархії для представлення їх як одиничних об'єктів. Цей шаблон дозволяє клієнтам обробляти як одиничні об'єкти, так і композитні структури однаковою способом.

Основні учасники шаблону "Composite":

Component (Компонент): Визначає інтерфейс для всіх конкретних об'єктів та композитів.

Leaf (Листок): Представляє одиничний об'єкт, який не має підкомпонентів.

Composite (Композит): Представляє композит, який має підкомпоненти. Реалізує методи інтерфейсу Component для роботи з підкомпонентами.

Принцип роботи шаблону полягає в тому, що інтерфейс для індивідуальних об'єктів і композитів є ідентичним, що дозволяє їх використовувати уніфіковано.

Шаблон проектування «FLYWEIGHT»

Шаблон проектування "Flyweight" відноситься до структурних патернів і використовується для оптимізації роботи з об'єктами, які мають спільні частини, шляхом винесення загальної частини в зовнішню структуру. Цей шаблон дозволяє економити ресурси, особливо коли кількість об'єктів велика і багато з них може використовувати одні й ті ж ресурси.

Основні учасники шаблону "Flyweight":

Flyweight (Легковаговик): Визначає інтерфейс, через який конкретні легковаговики можуть отримати та використовувати зовнішні ресурси.

ConcreteFlyweight (Конкретний легковаговик): Реалізує інтерфейс легковаговика та зберігає внутрішній стан, який може бути спільно використаний.

UnsharedConcreteFlyweight (Неподілений конкретний легковаговик): Має власні внутрішній стан, який не може бути спільно використаний з іншими об'єктами.

FlyweightFactory (Фабрика легковаговиків): Відповідає за управління легковаговиками, забезпечуючи можливість вилучення та повернення легковаговиків.

Принцип роботи шаблону полягає в тому, що спільні ресурси витягуються в окремий об'єкт (легковаговик), і замість створення нового об'єкта для кожного випадку використання ресурсів, використовується вже існуючий об'єкт.

Шаблон проектування «INTERPRETER»

Шаблон проектування "Interpreter" відноситься до поведінкових патернів і використовується для визначення граматики для мови та інтерпретації речень цієї мови. Цей шаблон дозволяє створити інтерпретатор, який читає, розуміє та виконує вирази, визначені у мові.

Основні учасники шаблону "Interpreter":

AbstractExpression (Абстрактний вираз): Визначає інтерфейс для інтерпретаторів.

TerminalExpression (Термінальний вираз): Реалізує інтерфейс абстрактного виразу для термінальних символів граматики.

NonterminalExpression (Нетермінальний вираз): Реалізує інтерфейс абстрактного виразу для нетермінальних символів граматики, об'єднує термінальні вирази та інші нетермінальні вирази.

Context (Контекст): Містить інформацію, яку інтерпретатор використовує для виконання операцій.

Client (Клієнт): Створює об'єкт контексту та вирази для інтерпретації мови.

Принцип роботи шаблону полягає в тому, що ми визначаємо граматику мови та створюємо вирази для інтерпретації цієї граматики. Клієнт створює об'єкт контексту та вирази, інтерпретатор яких здатен виконувати вирази, представлені у мові.

Шаблон проектування «VISITOR»

Шаблон проектування "Visitor" відноситься до поведінкових патернів і використовується для визначення нової операції над об'єктами без зміни їхньої структури. Цей шаблон дозволяє визначати нові алгоритми, не змінюючи класи об'єктів, над якими ці алгоритми використовуються.

Основні учасники шаблону "Visitor":

Visitor (Відвідувач): Визначає інтерфейс з методами відвідувача для кожного класу елемента.

ConcreteVisitor (Конкретний відвідувач): Реалізує інтерфейс відвідувача та визначає конкретні алгоритми для кожного класу елемента.

Element (Елемент): Визначає інтерфейс для об'єктів, над якими можуть бути використані операції відвідувача.

ConcreteElement (Конкретний елемент): Реалізує інтерфейс елемента та визначає специфічну для класу операцію відвідувача.

ObjectStructure (Об'єктна структура): Зберігає колекцію об'єктів і надає інтерфейс для їхньої ітерації.

Принцип роботи шаблону полягає в тому, що нові алгоритми визначаються в класі відвідувача, і кожен клас елемента має метод `accept()`, який приймає відвідувача та викликає відповідний метод відвідувача.

Дослідження шаблону «Composite»

Інтерфейс "Browser" має методи "printBrowserName" (вивести назву браузера) і "printBrowserLocation" (вивести розташування браузера).



```
FirefoxBrowser.java x InternetExplorerBrowser.java x OperaBrowser.java x
1 public interface Browser {
2     void printBrowserName();
3     void printBrowserLocation();
4 }
```

Цей клас FirefoxBrowser реалізує інтерфейс Browser. Він має поля для ідентифікації, назви та розташування браузера. Конструктор приймає ці параметри при створенні об'єкта. Методи printBrowserName та printBrowserLocation виводять відповідно назву браузера та його розташування. Клас надає також методи доступу до полів (getId, getName, getLocation) та їх змінення (setId, setName, setLocation).

```
FirefoxBrowser.java x InternetExplorerBrowser.java x OperaBrowser.java x GoogleBrowser.java x BrowserIntegrator.java x Browser.java x C
1 public class FirefoxBrowser implements Browser{
2
3     private Integer id;
4     private String name;
5     private String location;
6
7     public FirefoxBrowser(int id, String name, String location) {
8         this.id = id;
9         this.name = name;
10        this.location = location;
11    }
12
13    @Override
14    public void printBrowserName() { System.out.println(getClass().getSimpleName()); }
15
16
17    public Integer getId() { return id; }
18
19
20    public void setId(Integer id) {
21        this.id = id;
22    }
23
24
25    public String getName() { return name; }
26
27
28    public void setName(String name) { this.name = name; }
29
30
31    public String getLocation() { return location; }
32
33
34    public void setLocation(String location) { this.location = location; }
35
36
37
38
39
40
41
```

```
33
34    public String getLocation() { return location; }
35
36
37
38    public void setLocation(String location) { this.location = location; }
39
40
41
42    @Override
43    public void printBrowserLocation() {
44        System.out.println("Firefox location: "+getLocation());
45    }
46
47
```

Цей клас `GoogleBrowser` реалізує інтерфейс `Browser` та має поля для ідентифікації, назви та розташування браузера. Конструктор приймає ці параметри при створенні об'єкта. Методи `printBrowserName` та `printBrowserLocation` виводять відповідно назву браузера та його розташування. Клас надає також методи доступу до полів (`getId`, `getName`, `getLocation`) та їх змінення (`setId`, `setName`, `setLocation`).

```
FirefoxBrowser.java x InternetExplorerBrowser.java x OperaBrowser.java x GoogleBrowser.java x BrowserIntegrator.java x Browser
1 public class GoogleBrowser implements Browser{
2     private Integer id;
3     private String name;
4     private String location;
5
6     public GoogleBrowser(int id, String name, String location) {
7         this.id = id;
8         this.name = name;
9         this.location = location;
10    }
11
12    @Override
13    public void printBrowserName() { System.out.println(getClass().getSimpleName()); }
14
15
16
17    public Integer getId() { return id; }
18
19
20
21    public void setId(Integer id) { this.id = id; }
22
23
24
25
26    public String getName() { return name; }
27
28
29
30    public void setName(String name) { this.name = name; }
31
32
33
34    public String getLocation() { return location; }
35
36
37
38    public void setLocation(String location) { this.location = location; }
39
40
41
42    @Override
43    public void printBrowserLocation() {
44        System.out.println("Google location: " + getLocation());
45    }
46
47 }
```

```
41
42    @Override
43    public void printBrowserLocation() {
44        System.out.println("Google location: " + getLocation());
45    }
46
47 }
```

Цей клас `InternetExplorerBrowser` реалізує інтерфейс `Browser` та має поля для ідентифікації, назви та розташування браузера. Конструктор приймає ці параметри при створенні об'єкта. Методи `printBrowserName` та `printBrowserLocation` виводять відповідно назву браузера та його розташування. Клас надає також методи доступу до полів (`getId`, `getName`, `getLocation`) та їх змінення (`setId`, `setName`, `setLocation`).

```
FirefoxBrowser.java x InternetExplorerBrowser.java x OperaBrowser.java x GoogleBrowser.java x BrowserIntegrator.java x Browser.java x Comp
1 public class InternetExplorerBrowser implements Browser{
2
3     private Integer id;
4     private String name;
5     private String location;
6
7     public InternetExplorerBrowser(int id, String name, String location) {
8         this.id = id;
9         this.name = name;
10        this.location = location;
11    }
12
13    @Override
14    public void printBrowserName() { System.out.println(getClass().getSimpleName()); }
15
16
17    public Integer getId() { return id; }
18
19
20    public void setId(Integer id) { this.id = id; }
21
22
23    public String getName() { return name; }
24
25
26    public void setName(String name) { this.name = name; }
27
28
29    public String getLocation() { return location; }
30
31
32    public void setLocation(String location) { this.location = location; }
33
34
35
36
37
38
39
40
41
42
43    @Override
44    public void printBrowserLocation() {
45        System.out.println("Internet explorer location: " + getLocation());
46    }
47
48
49
50
51
52
53
54
55
56
57
58
59
60
61
62
63
64
65
66
67
68
69
70
71
72
73
74
75
76
77
78
79
80
81
82
83
84
85
86
87
88
89
90
91
92
93
94
95
96
97
98
99
100
```

Цей клас OperaBrowser реалізує інтерфейс Browser та має поля для ідентифікації, назви та розташування браузера. Конструктор приймає ці параметри при створенні об'єкта. Методи printBrowserName та printBrowserLocation виводять відповідно назву браузера та його розташування. Клас надає також методи доступу до полів (getId, getName, getLocation) та їх змінення (setId, setName, setLocation).

```
FirefoxBrowser.java x InternetExplorerBrowser.java x OperaBrowser.java x GoogleBrowser.java x BrowserIntegrator.java x Br
1 public class OperaBrowser implements Browser{
2
3     private Integer id;
4     private String name;
5     private String location;
6
7     public OperaBrowser(int id, String name, String location) {
8         this.id = id;
9         this.name = name;
10        this.location = location;
11    }
12
13    @Override
14    public void printBrowserName() { System.out.println(getClass().getSimpleName()); }
15
16
17    public Integer getId() { return id; }
18
19
20    public void setId(Integer id) { this.id = id; }
21
22
23    public String getName() { return name; }
24
25
26    public void setName(String name) { this.name = name; }
27
28
29    public String getLocation() { return location; }
30
31
32    public void setLocation(String location) { this.location = location; }
33
34
35    @Override
36    public void printBrowserLocation() {
37        System.out.println("Opera location: " + getLocation());
38    }
39
40
41 }
```

```
41
42     @Override
43     public void printBrowserLocation() {
44         System.out.println("Opera location: " + getLocation());
45     }
46
47 }
```


Клас BrowserIntegrator реалізує інтерфейс Browser та представляє інтегратор браузерів. Він має поле для назви та список дочірніх браузерів (childBrowsers). Конструктор приймає назву та ініціалізує порожній список дочірніх браузерів. Методи printBrowserName та printBrowserLocation викликають відповідні методи для всіх дочірніх браузерів. Метод addBrowser додає новий браузер до списку дочірніх браузерів.

```
FirefoxBrowser.java x InternetExplorerBrowser.java x OperaBrowser.java x GoogleBrowser.java x BrowserIntegrator.java
1  import java.util.ArrayList;
2  import java.util.List;
3
4  public class BrowserIntegrator implements Browser {
5      private String name;
6
7      private List<Browser> childBrowsers;
8
9      public BrowserIntegrator(String name) {
10         this.name = name;
11         this.childBrowsers = new ArrayList<>();
12     }
13
14     public void printBrowserName() { childBrowsers.forEach(Browser::printBrowserName); }
15
16
17
18     public void printBrowserLocation() {
19         childBrowsers.forEach(Browser::printBrowserLocation);
20     }
21
22
23     public void addBrowser(Browser department) { childBrowsers.add(department); }
24
25
26 }
```

У цьому класі CompositeDemo створюються об'єкти різних браузерів (FirefoxBrowser, GoogleBrowser, InternetExplorerBrowser, OperaBrowser) і об'єкт BrowserIntegrator, який представляє інтегратор браузерів. До інтегратора додаються різні браузери за допомогою методу addBrowser. Потім викликаються методи printBrowserName та printBrowserLocation, які виводять назви та розташування всіх браузерів, включаючи дочірні браузери, що є частиною інтегратора.

```
1 public class CompositeDemo {
2     public static void main(String args[]) {
3         FirefoxBrowser firefoxBrowser= new FirefoxBrowser( id: 1, name: "Firefox Browser",
4             location: "C:\\Program Files\\Mozilla Firefox\\firefox.exe");
5         GoogleBrowser googleBrowser = new GoogleBrowser( id: 2, name: "Google Browser",
6             location: "C:\\Program Files\\Google\\Chrome\\Application\\chrome.exe");
7         InternetExplorerBrowser internetExplorerBrowser = new InternetExplorerBrowser( id: 3, name: "Internet Explorer Browser",
8             location: "C:\\Program Files\\Internet Explorer\\iexplore.exe");
9         OperaBrowser operaBrowser = new OperaBrowser( id: 4, name: "Opera Browser",
10            location: "C:\\Users\\shepe\\AppData\\Local\\Programs\\Opera\\launcher.exe");
11
12         BrowserIntegrator browserIntegrator = new BrowserIntegrator( name: "Browser Integrator");
13         browserIntegrator.addBrowser(firefoxBrowser);
14         browserIntegrator.addBrowser(googleBrowser);
15         browserIntegrator.addBrowser(internetExplorerBrowser);
16         browserIntegrator.addBrowser(operaBrowser);
17         browserIntegrator.printBrowserName();
18         browserIntegrator.printBrowserLocation();
19     }
20 }
```

Результат:

```
Run: CompositeDemo x
C:\Users\shepe\.jdk\corretto-17.0.3\bin\java.exe -javaagent:C:\Users\shepe\AppData\Local\JetBrains\Toolbox\apps\IDEA-U\
FirefoxBrowser
GoogleBrowser
InternetExplorerBrowser
OperaBrowser
Firefox location: C:\Program Files\Mozilla Firefox\firefox.exe"
Google location: C:\Program Files\Google\Chrome\Application\chrome.exe
Internet explorer location: C:\Program Files\Internet Explorer\iexplore.exe
Opera location: C:\Users\shepe\AppData\Local\Programs\Opera\launcher.exe

Process finished with exit code 0
```

Висновок:

Лабораторна робота спрямована на вивчення та реалізацію шаблону проектування "Composite". У процесі вивчення цього шаблону були розглянуті основні концепції, такі як компоненти, листки і композити, що дозволяють створювати структури деревоподібних ієрархій.

У реалізації було показано, як створити абстрактний клас Component, який описує загальний інтерфейс для всіх елементів структури. Класи Leaf представляють одиничні об'єкти, а Composite дозволяють об'єднувати об'єкти в більш складні структури.

Основний висновок полягає в тому, що шаблон "Composite" дозволяє створювати ієрархії об'єктів так, що клієнт може взаємодіяти з одиничними об'єктами і композитами однаковою способом. Це полегшує роботу зі структурами, де одні об'єкти можуть містити інші, а клієнту не потрібно відмежовувати їхні типи під час використання.