

**МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ**  
**НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ**  
**«КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ**  
**ІМЕНІ ІГОРЯ СІКОРСЬКОГО»**

**КАТЕДРА ОБЧИСЛЮВАЛЬНОЇ ТЕХНІКИ**

## **Інженерія програмного забезпечення**

### **Лабораторна робота №4**

«Структурні шаблони проектування. Шаблони  
Flyweight, adapter, bridge, facade»

Виконав:  
студент групи ІО-32  
Крадожон М.  
Перевірів(-ла):  
Васильєва М.

Київ – 2023

## Лабораторна робота №4

**Тема:** «Структурні шаблони проектування. Шаблони Flyweight, adapter, bridge, facade»

**Мета:** Вивчення структурних шаблонів. Отримання базових навичок з застосування шаблонів Flyweight, Adapter, Bridge, Facade.

**Варіант:** 3215 % 11 = 3

Визначити специфікації класів, які подають об'єкти-іконки для зображення елементів файлової системи при побудові графічного інтерфейсу користувача (GUI) – примітиви (файли) та їх композиції (директорії). Забезпечити ефективне використання пам'яті при роботі з великою кількістю графічних об'єктів. Реалізувати метод рисування графічного об'єкту.

Код (TypeScript):

```
1  /**
2  * Інтерфейс для координат.
3  * @interface
4  */
5  interface Coordinates {
6  /** Координата по осі X */
7  x: number
8  /** Координата по осі Y */
9  y: number
10 }
11
12 /**
13 * Інтерфейс для іконки.
14 * @interface
15 */
16 interface Icon {
17 /**
18  * Метод для малювання іконки.
19  * @param {Coordinates} coordinates - Координати, де потрібно намалювати іконку.
20  */
21 draw({ x, y }: Coordinates): void
22 }
23
24 /**
25 * Інтерфейс для Flyweight.
26 * @interface
27 */
28 interface Flyweight {
29 /**
30  * Метод для малювання Flyweight.
31  * @param {Coordinates} coordinates - Координати, де потрібно намалювати Flyweight.
32  */
```

```

33 draw({ x, y }: Coordinates): void
34 }
35
36 /**
37  * Реалізація іконки.
38  * @class
39  * @implements {Icon}
40  */
41 class IconImpl implements Icon {
42     private type: string
43
44     /**
45      * Створює нову іконку.
46      * @param {string} type - Тип іконки.
47      */
48     constructor(type: string) {
49         this.type = type
50     }
51
52     /**
53      * Малює іконку за заданими координатами.
54      * @param {Coordinates} coordinates - Координати для малювання.
55      */
56     draw({ x, y }: Coordinates): void {
57         console.log(`Метод draw з параметрами x=${x} y=${y} для іконки типу ${this.type}`)
58     }
59 }
60
61 /**
62  * Фабрика для створення іконок.
63  * @class
64  */
65 class IconFactory {
66     private icons: { [key: string]: Flyweight }
67
68     /**
69      * Створює нову фабрику іконок.
70      */
71     constructor() {
72         this.icons = {}
73     }
74
75     /**
76      * Отримує іконку за типом, створюючи її, якщо вона ще не існує.
77      * @param {string} type - Тип іконки.
78      * @returns {Flyweight} - Іконка.
79      */
80     getIcon(type: string): Flyweight {
81         if (!this.icons[type]) {
82             this.icons[type] = new IconImpl(type)
83         }
84         return this.icons[type]
85     }

```

```

86 }
87
88 /**
89  * Іконка файлу.
90  * @class
91  * @implements {Icon}
92  */
93 class FileIcon implements Icon {
94     private icon: Flyweight
95
96     /**
97      * Створює нову іконку файлу.
98      * @param {string} type - Тип іконки.
99      * @param {IconFactory} factory - Фабрика для отримання іконок.
100     */
101     constructor(type: string, factory: IconFactory) {
102         this.icon = factory.getIcon(type)
103     }
104
105     /**
106      * Малює іконку файлу за заданими координатами.
107      * @param {Coordinates} coordinates - Координати для малювання.
108     */
109     draw({ x, y }: Coordinates): void {
110         this.icon.draw({ x, y })
111     }
112 }
113
114 /**
115  * Іконка директорії.
116  * @class
117  * @implements {Icon}
118  */
119 class DirectoryIcon implements Icon {
120     private icons: Flyweight[]
121
122     /**
123      * Створює нову іконку директорії.
124      * @param {string[]} types - Типи іконок.
125      * @param {IconFactory} factory - Фабрика для отримання іконок.
126     */
127     constructor(types: string[], factory: IconFactory) {
128         this.icons = types.map((type) => factory.getIcon(type))
129     }
130
131     /**
132      * Малює іконки директорії за заданими координатами.
133      * @param {Coordinates} coordinates - Координати для малювання.
134     */
135     draw({ x, y }: Coordinates): void {
136         this.icons.forEach((icon) => icon.draw({ x, y }))
137     }
138 }

```

```

139
140 const factory = new IconFactory()
141
142 const fileIcon = new FileIcon('file', factory)
143 fileIcon.draw({ x: 10, y: 20 })
144
145 const directoryIcon = new DirectoryIcon(['file', 'directory'], factory)
146 directoryIcon.draw({ x: 30, y: 40 })
147

```

## Результат:

```
> yarn run start
```

Метод draw з параметрами x=10 y=20 для іконки типу file

Метод draw з параметрами x=30 y=40 для іконки типу file

Метод draw з параметрами x=30 y=40 для іконки типу directory

## Документація (за допомогою TypeDoc):

lab3 / IconFactory /

### Class IconFactory

Фабрика для створення іконок.

Defined in index.ts:65

#### ▼ Index

#### Constructors

Ⓒ constructor

#### Methods

Ⓜ getIcon

#### ▼ Methods

##### getIcon

getIcon(type): *Flyweight*

Отримує іконку за типом, створюючи її, якщо вона ще не існує.

#### Parameters

▪ type: *string*

Тип іконки.

Returns *Flyweight*

◦ Іконка.

Defined in index.ts:80

**Висновок:**

Для виконання лабораторної роботи було прочитано та засвоєно необхідний теоретичний матеріал. Виконано 3 варіант, створений проєкт зі інтерфейсом та необхідними класами, для реалізації проєкту було використано шаблон "Flyweight". Також було створено коментарі у кожному файлі для TypeDoc. Мета лабораторної роботи виконана.