

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

**КАТЕДРА ОБЧИСЛЮВАЛЬНОЇ ТЕХНІКИ**

## **Інженерія програмного забезпечення**

### **Лабораторна робота №5**

«Шаблони поведінки. Шаблони iterator, mediator, observer»

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

## Лабораторна робота №5

**Тема:** «Шаблони поведінки. Шаблони iterator, mediator, observer»

**Мета:** Вивчення шаблонів поведінки. Отримання базових навичок з застосування шаблонів Iterator, Mediator та Observer.

**Варіант:**  $3215 \% 13 = 4$

Визначити специфікації класів для елемента ігрового поля (комірки) та самого простору. Кожна комірка представляє стан частини поля (наприклад, порожнє місце, перешкода, ігровий персонаж тощо). Простір має контролювати та узгоджувати зміни станів комірок для забезпечення слабкої зв'язаності між ними. Реалізувати централізований механізм сумісної зміни стану елементів.

Код (TypeScript):

```
1  /**
2  * Інтерфейс для комірки ігрового поля.
3  */
4  interface Cell {
5  /**
6  * Повертає стан комірки.
7  */
8  getState(): string
9  /**
10 * Змінює стан комірки.
11 * @param state Новий стан комірки.
12 */
13 setState(state: string): void
14 }
15
16 /**
17 * Інтерфейс для простору.
18 */
19 interface Space {
20 /**
21 * Інтерфейс для простору.
22 */
23 addCell(cell: Cell): void
24 /**
25 * Видаляє комірку з простору.
26 * @param cell Комірка для видалення.
27 */
28 removeCell(cell: Cell): void
29 /**
30 * Змінює стан комірки в просторі.
31 * @param cell Комірка для зміни стану.
32 * @param state Новий стан комірки.
33 */
```

```
34 updateCellState(cell: Cell, state: string): void
35 /**
36  * Повіdomляє про зміни в просторі.
37  */
38 notifyChanges(): void
39 }
40
41 /**
42  * Клас комірки ігрового поля.
43  */
44 class GameCell implements Cell {
45 /**
46  * Стан комірки.
47  */
48 private state: string
49
50 /**
51  * Конструктор класу.
52  * @param initialState Початковий стан комірки.
53  */
54 constructor(initialState: string = 'empty') {
55     this.state = initialState
56 }
57
58 /**
59  * Повертає стан комірки.
60  */
61 getState(): string {
62     return this.state
63 }
64
65 /**
66  * Змінює стан комірки.
67  * @param state Новий стан комірки.
68  */
69 setState(state: string): void {
70     console.log(`Метод setState з параметром state=${state}`)
71     this.state = state
72 }
73 }
74
75 /**
76  * Клас простору.
77  */
78 class GameSpace implements Space {
79 /**
80  * Масив комірок.
81  */
82 private cells: Cell[]
83
84 /**
85  * Конструктор класу.
86  */
```

```

87 constructor() {
88   this.cells = []
89 }
90
91 /**
92  * Додає комірку до простору.
93  * @param cell Комірка для додавання.
94  */
95 addCell(cell: Cell): void {
96   console.log(`Метод addCell з параметром cell=${cell.getState()}`)
97   this.cells.push(cell)
98 }
99
100 /**
101  * Видаляє комірку з простору.
102  * @param cell Комірка для видалення.
103  */
104 removeCell(cell: Cell): void {
105   console.log(`Метод removeCell з параметром cell=${cell.getState()}`)
106   const index = this.cells.indexOf(cell)
107   index !== -1 && this.cells.splice(index, 1)
108 }
109
110 /**
111  * Змінює стан комірки в просторі.
112  * @param cell Комірка для зміни стану.
113  * @param state Новий стан комірки.
114  */
115 updateCellState(cell: Cell, state: string): void {
116   console.log(`Метод updateCellState з параметрами cell=${cell.getState()} state=${state}`)
117   cell.setState(state)
118 }
119
120 /**
121  * Повіdomляє про зміни в просторі.
122  */
123 notifyChanges(): void {
124   this.cells.forEach((cell, index) => {
125     console.log(`Стан комірки ${index}: ${cell.getState()}`)
126   })
127 }
128 }
129
130 const mediator = new GameSpace()
131 const cell1 = new GameCell()
132 const cell2 = new GameCell('obstacle')
133
134 mediator.addCell(cell1)
135 mediator.addCell(cell2)
136
137 mediator.updateCellState(cell1, 'player')
138 mediator.updateCellState(cell2, 'empty')

```

## Результат:

```
> yarn run start
```

Метод addCell з параметром cell=empty

Метод addCell з параметром cell=obstacle

Метод updateCellState з параметрами cell=empty state=player

Метод setState з параметром state=player

Метод updateCellState з параметрами cell=obstacle state=empty

Метод setState з параметром state=empty

Стан комірки 0: player

Стан комірки 1: empty

## Документація (за допомогою TypeDoc):

**Class GameCell**

Клас комірки ігрового поля.

**Implements**

- Cell

Defined in index.ts:44

▼ **Index**

**Constructors**

- constructor

**Methods**

- getState
- setState

▼ **Methods**

**getState**

---

`getState(): string`

---

Повертає стан комірки.

**Returns** *string*

Implementation of [Cell.getState](#)

Defined in index.ts:61

**setState**

---

`setState(state): void`

---

Змінює стан комірки.

**Parameters**

- state: *string*

Новий стан комірки.

**Returns** *void*

Implementation of [Cell.setState](#)

Defined in index.ts:69

**Висновок:**

Для виконання лабораторної роботи було прочитано та засвоєно необхідний теоретичний матеріал. Виконано 4 варіант, створений проєкт зі інтерфейсом та необхідними класами, для реалізації проєкту було використано шаблон “Mediator”. Також було створено коментарі у кожному файлі для TypeDoc. Мета лабораторної роботи виконана.