МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
Національний Технічний Університет України
«Київський Політехнічний Інститут»
Факультет інформатики та обчислювальної техніки
Кафедра обчислювальної техніки

Лабораторна робота №5
з дисципліни «Інженерія програмного забезпечення»
на тему: «Шаблони поведінки. Шаблони Iterator, Mediator, Observer»

Виконав:
студент 2-го курсу ФІОТ
групи ІВ-71
Мазан Я.В.
Номер залікової книжки: 7109
Варіант: 8

Перевірив:
доцент кафедри ОТ
Антонюк А.І.

Київ – 2018

Завдання

Варіант – 8
Визначити специфікації класів для подання реляційної таблиці. Реалізувати механізм тригерів – виконання додаткових дій при зміні елемента.

Код програми

Main.java:

```
package com.lab111.labwork5;
/**
 * Program of labwork testing
 * @author - Yan Mazan, variant 8
 */
public class Main {
    public static void main(String[] args) {
        int[][] vals = {{3,3,3},
                        {6,16,6},
                        {-100,10,1},
                        {2,2,2}};
        String[] keys = {"Alliances","Labworks","Bleach characters","Countries"};
        Trigger1 subscriber = new Trigger1("Trigger");
        RelationTable table = new RelationTable(keys,vals);
        table.subscribeTrigger("Bleach characters",subscriber);
        table.set("Bleach characters", 1,2);
        table.set("Bleach characters", 0,2);
        System.out.println("Some table value: " + table.get("Bleach characters",1));
        Trigger2 subscriber2 = new Trigger2("Trigger2");
        table.subscribeTrigger("Countries", subscriber2);
        table.set("Countries",1,1000000);
        table.unsubscribeTrigger("Countries", subscriber2);
        table.set("Countries",2,1);
        System.out.println("Trigger2 wasn't activated because it unsubscribed from table
updates");
    }
}
```

RelationTable.java:

```
package com.lab111.labwork5;
import java.util.HashMap;
/**
 * our relation table
 */
public class RelationTable {
    private HashMap<String,Field> fields;
    public RelationTable(String[] keys, int[][] vals) {
        fields = new HashMap<>();
        for (int i = 0; i < keys.length; i++) {
            fields.put(keys[i],new Field(keys[i],vals[i]));
        }
    }
    public void set(String key, int index, int new_val) {
        fields.get(key).set(index,new_val);
    }
    public int get(String key, int index) {
        return fields.get(key).get(index);
    }
    public void subscribeTrigger(String key, Subscriber s) {
```

```
            fields.get(key).subscribeTrigger(s);
        }
        public void unsubscribeTrigger(String key, Subscriber s) {
            fields.get(key).unsubscribeTrigger(s);
        }
}
```

Field.java:

```
package com.lab111.labwork5;
import java.util.HashSet;
/**
 * Field of relation table
 * is main publisher of events, on which triggers subscribe
 */
public class Field {
    public String key;
    private int[] elements;
    private HashSet<Subscriber> subscribers;
    public Field(String k, int elems[]) {
        subscribers = new HashSet<>();
        elements = elems;
        key = k;
    }
    public void set(int index, int new_val) {
        elements[index] = new_val;
        notifyTriggers();
    }
    public int get(int index) {
        return elements[index];
    }
    public void subscribeTrigger(Subscriber s) {
        subscribers.add(s);
    }
    public void unsubscribeTrigger(Subscriber s) {
        subscribers.remove(s);
    }
    private void notifyTriggers() {
        subscribers.forEach((elem)->{
            elem.update(this);
        });
    }
}
```

Subscriber.java:

```
package com.lab111.labwork5;
/**
 * Basic interface that all subscribers to class' Field events must implement
 */
public interface Subscriber {
    void update(Field publsher);
}
```

Trigger1.java:

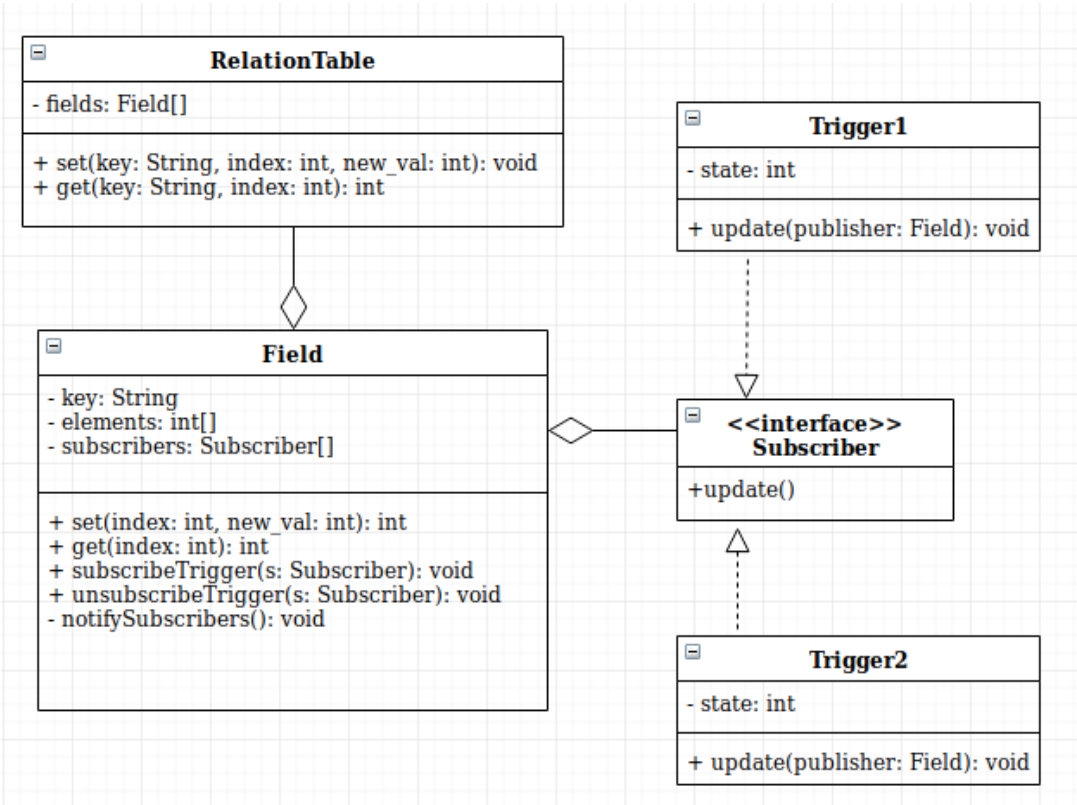
```
package com.lab111.labwork5;
/**
 * The first trigger
 */
public class Trigger1 implements Subscriber {
    private int state;
    private String name;
```

```
    public Trigger1(String name) {
        state = 0;
        this.name = name;
    }
    public void update(Field publisher) {
        state++;
        System.out.println(name + " activation by field " + publisher.key + ". State
updated to " + state);
    }
}
```

Trigger2.java:

```
package com.lab111.labwork5;
/**
 * The second trigger
 */
public class Trigger2 implements Subscriber {
    private int state;
    private String name;
    public Trigger2(String name) {
        state = 1000;
        this.name = name;
    }
    public void update(Field publisher) {
        state = (publisher.hashCode()+state)/2;
        System.out.println(name + " activation by field " + publisher.key + ". State
updated to " + state);
    }
}
```

Діаграма класів



Тестування програми

```
/usr/lib/jvm/java-1.11.0-openjdk-amd64/bin/java -javaagent:/home/yan/ic
Trigger activation by field Bleach characters. State updated to 1
Trigger activation by field Bleach characters. State updated to 2
Some table value: 2
Trigger2 activation by field Countries. State updated to 402291195
Trigger2 wasn't activated because it unsubscribed from table updates

Process finished with exit code 0
```

Висновок

Було реалізовано реляційну таблицю, в якій при зміні елемента виконуються тригери – в моєму випадку оголошується факт їх активації та змінюється стан тригера. Для реалізації механізму тригерів було використано шаблон Observer. Результати успішної роботи тестової програми, що наведені вище, підтверджують правильність обраних рішень.