**НТУУ «КПІ» 2014**

**Інженерія ПЗ**

**Лабораторна робота №6:**

**Шаблони поведінки - 2.**

**Шаблони Strategy, Chain of Responsibility та Visitor.**

**Факультет: ФІОТ**

**Група: ІО-34**

**Заліковка №3405**

**Виконав:**

**Власов М. Д.**

**Викладач:**

**Антонюк А.І.**

**Мета**

Вивчення шаблонів поведінки. Отримання базових навичок з застосування шаблонів Strategy, Chain of Responsibility та Visitor.

**Завдання**

1. Повторити шаблони поведінки для проектування ПЗ. Знати загальну характеристику шаблонів поведінки та призначення кожного з них.

2. Детально вивчити шаблони поведінки для проектування ПЗ — Strategy, Chain of Responsibility та Visitor. Для кожного з них:

1. вивчити Шаблон, його призначення, альтернативні назви, мотивацію, випадки коли його застосування є доцільним та результати такого застосування;
2. знати особливості реалізації Шаблону, споріднені шаблони, відомі випадки його застосування в програмних додатках;
3. вільно володіти структурою Шаблону, призначенням його класів та відносинами між ними;
4. вміти розпізнавати Шаблон в UML діаграмі класів та будувати сирцеві коди Java-класів, що реалізують шаблон.

3. В підготованому проекті (ЛР1) створити програмний пакет com.lab111.labwork6. В пакеті розробити інтерфейси і класи, що реалізують завдання (згідно варіанту) з застосуванням одного чи декількох шаблонів (п.2). В розроблюваних класах повністю реалізувати методи, пов'язані з функціюванням Шаблону. Методи, що реалізують бізнес-логіку закрити заглушками з виводом на консоль інформації про викликаний метод та його аргументи. Приклад реалізації бізнес-методу:

void draw(int x, int y){

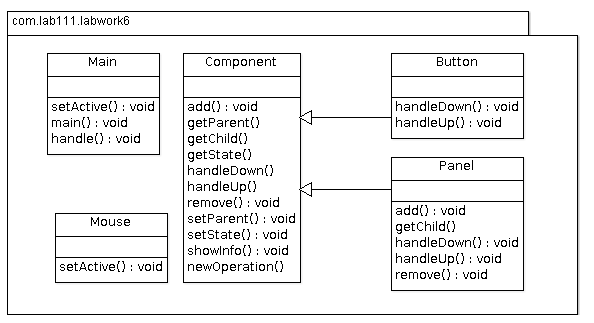
System.out.println(“Метод draw з параметрами x=”+x+” y=”+y);

}

4. За допомогою автоматизованих засобів виконати повне документування розроблених класів (також методів і полів), при цьому документація має в достатній мірі висвітлювати роль певного класу в загальній структурі Шаблону та особливості конкретної реалізації.

**Варіант 3405 mod 10**

5. Визначити специфікації класів, що реалізують елементи графічного інтерфейсу користувача — панелі (композит) та кнопки (компонент). Реалізувати децентралізований механізм обробки події переміщення курсору миші. Кількість компонентів інтерфейсу, які реагують на цю подію, може змінюватись динамічно.

**Код**

**package** com.lab111.labwork6;

/\*\*

\* Leaf of the composite structure

\*

\* **@author** Maxym Vlasov

\* **@version** 1.0

\*/

**public** **class** Button **extends** Component {

Button(String name, Component parent) {

**this**.name = name;

**this**.parent = parent;

**this**.size = 0;

}

Button(String name) {

**this**(name, **null**);

}

/\*\*

\* **@param** value

\*/

**public** **void** handleUp(**boolean** value) {

System.***out***.println("name: " + name + " setFocusedState: " + value);

setState(value);

parent.handleUp(value);

}

**public** **void** handleDown() {

**if** (state == **true**) {

showInfo();

}

}

}

**package** com.lab111.labwork6;

/\*\*

\* Introduces interface of components

\*

\* **@author** Maxym Vlasov

\* **@version** 1.0

\*/

**public** **abstract** **class** Component {

/\*\*

\* state of the component

\*/

**protected** **boolean** state;

/\*\*

\* name of the component

\*/

**protected** String name;

/\*\*

\* size of the massive of components

\*/

**protected** **int** size;

/\*\*

\* holds parent of component

\*/

**protected** Component parent;

/\*\*

\*

\* **@return** state

\*/

**public** **boolean** getState() {

**return** state;

}

/\*\*

\* sets state=true

\*

\* **@param** value

\*/

**public** **void** setState(**boolean** value) {

**this**.state = value;

}

/\*\*

\* adds component

\*

\* **@param** comp

\*/

**public** **void** add(Component comp) {

}

/\*\*

\* removes component

\*

\* **@param** n

\*/

**public** **void** remove(**int** n) {

}

/\*\*

\* returns child

\*

\* **@param** n

\* - number of element

\* **@return**

\*/

**public** Component getChild(**int** n) {

**return** **null**;

}

/\*\*

\* **@return** parent of the component

\*/

**public** Component getParent() {

**return** parent;

}

/\*\*

\* sets parent

\*

\* **@param** parent

\* - reference to parent

\*/

**public** **void** setParent(Component parent) {

**this**.parent = parent;

}

/\*\*

\* shows info about component

\*/

**public** **void** showInfo() {

System.***out***.println("Focused component: " + name);

}

/\*\*

\* going from bottom focused element to header

\*

\* **@param** value

\*/

**public** **abstract** **void** handleUp(**boolean** value);

/\*\*

\* going from header to bottom focused element

\*/

**public** **abstract** **void** handleDown();

}

**package** com.lab111.labwork6;

/\*\*

\* Test class

\*

\* **@author** Maxym Vlasov

\* **@version** 1.0

\*/

**public** **class** Main {

**public** **static** **void** main(String[] args) {

Mouse mouse = **new** Mouse();

Component window = **new** Panel("Window");

Component panel1 = **new** Panel("Panel1", window);

Component panel2 = **new** Panel("Panel2", window);

Component panel3 = **new** Panel("Panel3", window);

Component panel4 = **new** Panel("Panel4", panel3);

window.add(panel1);

window.add(panel2);

window.add(panel3);

panel3.add(panel4);

Component button1 = **new** Button("Button1", panel1);

Component button2 = **new** Button("Button2", panel1);

Component button3 = **new** Button("Button3", panel2);

Component button4 = **new** Button("Button4", panel2);

Component button5 = **new** Button("Button5", panel4);

panel1.add(button1);

panel1.add(button2);

panel2.add(button3);

panel2.add(button4);

panel4.add(button5);

*setActive*(panel3);

*handle*(window);

*setActive*(panel1);

**if** (panel3.getState() == **true**) {

System.***out***.println("Panel 3 is active");

}

*handle*(window);

}

/\*\*

\* Calls handle methods

\*

\* **@param** window

\*/

**public** **static** **void** handle(Component window) {

Mouse.*getActive*.handleUp(**true**);

System.***out***.println();

window.handleDown();

System.***out***.println();

}

/\*\*

\* Sets active component

\*

\* **@param** component

\*/

**public** **static** **void** setActive(Component component) {

Component active = Mouse.*getActive*;

**if** (active != **null**)

active.handleUp(**false**);

Mouse.*setActive*(component);

}

}

**package** com.lab111.labwork6;

/\*\*

\* Introduces mouse

\*

\* **@author** Maxym Vlasov

\* **@version** 1.0

\*/

**public** **class** Mouse {

**public** **static** Component *getActive*;

/\*\*

\* sets mouse cursor on component

\*

\* **@param** component

\*/

**public** **static** **void** setActive(Component component) {

component.setState(**true**);

*getActive* = component;

}

}

**package** com.lab111.labwork6;

/\*\*

\* Composite structure

\*

\* **@author** Maxym Vlasov

\* **@version** 1.0

\*/

**public** **class** Panel **extends** Component {

**private** Component[] elements;

**protected** **int** counter = 0;

Panel(String name, Component parent) {

elements = **new** Component[20];

size = 0;

**this**.name = name;

**this**.parent = parent;

}

Panel(String name) {

**this**(name, **null**);

}

/\*\*

\* **@param** element

\*/

**public** **void** add(Component element) {

elements[size] = element;

size++;

}

/\*\*

\* **@param** n

\*/

**public** **void** remove(**int** n) {

**for** (**int** i = n; i < elements.length - 1; i++) {

elements[i] = elements[i + 1];

}

elements[elements.length - 1] = **null**;

size--;

}

/\*\*

\* **@param** n

\* **@return**

\*/

@SuppressWarnings("javadoc")

**public** Component getChild(**int** n) {

**return** elements[n];

}

/\*\*

\* **@param** value

\*/

**public** **void** handleUp(**boolean** value) {

**this**.setState(value);

System.***out***.println("name: " + name + " setFocuseState: " + value);

**if** (parent != **null**) {

parent.handleUp(value);

}

}

**public** **void** handleDown() {

**if** ((state == **true**) && (size == 0)) {

showInfo();

} **else** {

**if** ((state == **true**) && (size > 0)) {

**for** (**int** i = 0; i < size; i++) {

Component temp = getChild(i);

**if** (temp.getState() == **true**) {

counter++;

temp.handleDown();

}

}

**if** (counter == 0)

showInfo();

counter = 0;

}

}

}

}