Міністерство освіти і науки, молоді та спорту України

Національний технічний університет України

«Київський політехнічний інститут»

Факультет інформатики та обчислювальної техніки

Кафедра обчислювальної техніки

**Лабораторна робота №7**

По дисципліні «Інженерія програмного забезпечення»

Виконав: Перевірив:

Студент групи ІО-21 Викладач

Коноз А.О. Абу Усбах О. Н.

Дата здачі\_\_\_\_\_\_\_\_\_\_\_\_\_

Захищено з балом\_\_\_\_\_

Київ 2013

**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 - Memento, State, Command та Interpreter. Для кожного з них:

* вивчити Шаблон, його призначення, альтернативні назви, мотивацію, випадки коли його застосування є доцільним та результати такого застосування;
* знати особливості реалізації Шаблону, споріднені шаблони, відомі випадки його застосування в програмних додатках;
* вільно володіти структурою Шаблону, призначенням його класів та відносинами між ними;
* 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.labwork7. 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:

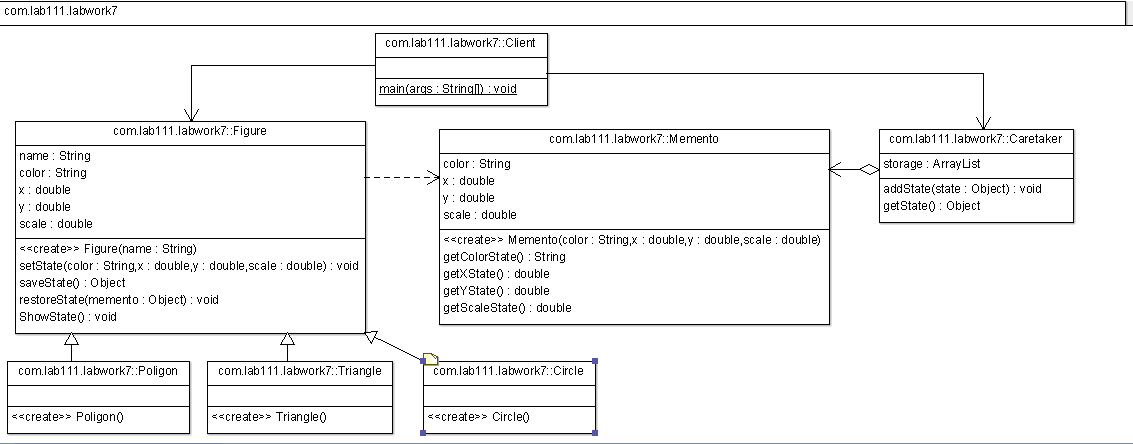
void draw(int x, int y){

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.

2112 mod 8 = 0



**package** com.lab111.labwork7;

/\*\*

\* **@author** User

\* Main class of program

\*/

**public** **class** Client {

/\*\*

\* **@param** args

\*/

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

Circle circle = **new** Circle();

Poligon poligon = **new** Poligon();

Triangle triangle = **new** Triangle();

Caretaker circleCaretaker = **new** Caretaker();

Caretaker poligoncaretaker = **new** Caretaker();

Caretaker trianglecaretaker = **new** Caretaker();

circle.setState("red", 1, 2, 100);

circle.ShowState();

circleCaretaker.addState(circle.saveState());

circle.setState("green", 2, 1, 50);

circle.ShowState();

circle.restoreState(circleCaretaker.getState());

circle.ShowState();

System.*out*.println();

poligon.setState("yelow", 5, 6, 1000);

poligon.ShowState();

poligoncaretaker.addState(poligon.saveState());

poligon.setState("black", 6, 5, 500);

poligon.ShowState();

poligon.restoreState(poligoncaretaker.getState());

poligon.ShowState();

}

}

**package** com.lab111.labwork7;

**import** java.util.ArrayList;

/\*\*

\* **@author** User

\* Class implements the functionality of caretaker

\*/

**public** **class** Caretaker {

/\*\*

\* **@param** storage is list for store state of object

\*/

**private** ArrayList storage = **new** ArrayList();

/\*\*

\* Add new state for storage

\* **@param** state new state

\*/

**public** **void** addState(Object state){

storage.add(state);

}

/\*\*

\* Return state and delete state from list

\* **@return** memento as Object

\* **@throws** Exception when list of storage is empty

\*/

**public** Object getState() **throws** Exception{

Object result = **null**;

**try**{

result = storage.get(storage.size() - 1);

storage.remove(storage.size() - 1);

}

**catch**(Exception e){

System.*out*.println("Storage of states is empty");

**throw** **new** Exception();

}

**finally**{

**return** result;

}

}

}

**package** com.lab111.labwork7;

/\*\*

\* **@author** User

\* Class-stub for triangle

\*/

**public** **class** Triangle **extends** Figure{

**public** Triangle(){

**super**("Triangle");

}

}

**package** com.lab111.labwork7;

/\*\*

\* **@author** User

\* Class-stub for circle

\*/

**public** **class** Circle **extends** Figure{

**public** Circle(){

**super**("Circle");

}

}

**package** com.lab111.labwork7;

/\*\*

\* **@author** User

\* Class-stub for poligon

\*/

**public** **class** Poligon **extends** Figure{

**public** Poligon(){

**super**("Poligon");

}

}

**package** com.lab111.labwork7;

/\*\*

\* **@author** User

\* This class implement the functionality of figure

\*/

**public** **class** Figure {

/\*\*

\* **@param** name name of figure

\* **@param** color color of object

\* **@param** x coordinate of object

\* **@param** y coordinate of object

\* **@param** scale is scale of object

\*/

**private** String name;

**private** String color;

**private** **double** x;

**private** **double** y;

**private** **double** scale;

/\*\*

\* Constructor set name of figure

\* **@param** name is name of figure

\*/

**public** Figure(String name) {

**this**.name = name;

}

/\*\*

\* **@param** color color of object

\* **@param** x coordinate of object

\* **@param** y coordinate of object

\* **@param** scale is scale of object

\*/

**public** **void** setState(String color, **double** x, **double** y, **double** scale) {

**this**.color = color;

**this**.x = x;

**this**.y = y;

**this**.scale = scale;

}

/\*\*

\* Save state of figure in memento

\* **@return** new memento as Object

\*/

**public** Object saveState() {

**return** (Object) **new** Memento(color, x, y, scale);

}

/\*\*

\* Restore state from memento

\* **@param** memento state for restored

\*/

**public** **void** restoreState(Object memento) {

Memento state = (Memento) memento;

color = state.getColorState();

x = state.getXState();

y = state.getYState();

scale = state.getScaleState();

}

/\*\*

\* Show information of figure

\*/

**public** **void** ShowState() {

System.*out*.println("Figure " + name

+ " with next parametrers: color = " + color + ", x = " + x

+ ", y = " + y + ", scale = " + scale);

}

}

**package** com.lab111.labwork7;

/\*\*

\* **@author** User

\*This class implements the functionality of Memento.

\*/

**public** **class** Memento {

/\*\*

\* **@param** color color of saved object

\* **@param** x coordinate of saved object

\* **@param** y coordinate of saved object

\* **@param** scale is scale of saved object

\*/

**private** String color;

**private** **double** x;

**private** **double** y;

**private** **double** scale;

/\*\*

\* It is constructor of memento. Constructor set state of saved object

\* **@param** color color of saved object

\* **@param** x coordinate of saved object

\* **@param** y coordinate of saved object

\* **@param** scale is scale of saved object

\*/

**public** Memento(String color, **double** x, **double** y, **double** scale ){

**this**.color = color;

**this**.x = x;

**this**.y =y;

**this**.scale = scale;

}

/\*\*

\*

\* **@return** This method return color of saved object

\*/

**public** String getColorState(){

**return** color;

}

/\*\*

\* **@return** This method return coordinate x of saved object

\*/

**public** **double** getXState(){

**return** x;

}

/\*\*

\* **@return** This method return coordinate y of saved object

\*/

**public** **double** getYState(){

**return** y;

}

/\*\*

\* **@return** This method return scale of saved object

\*/

**public** **double** getScaleState(){

**return** scale;

}

}