Національний технічний університет України «Київський політехнічний інститут»
Кафедра обчислювальної техніки

Лабораторна робота №3

з дісципліни ***«Інженерія програмного забезпечення»***

Виконав:
студент 2 курсу
ФІОТ гр. ІО-32
Довгаль Д.С.
Залікова книжка №3211

Київ 2014 р.

Завдання

1. Ознайомитись з призначенням та видами шаблонів проектування ПЗ. Вивчити класифікацію шаблонів проектування ПЗ. Знати назви шаблонів, що відносяться до певного класу.

2. Вивчити структурні шаблонів проектування ПЗ. Знати загальну характеристику структурних шаблонів та призначення кожного з них.

3. Детально вивчити структурні шаблони проектування Composite, Decorator та Proxy. 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.

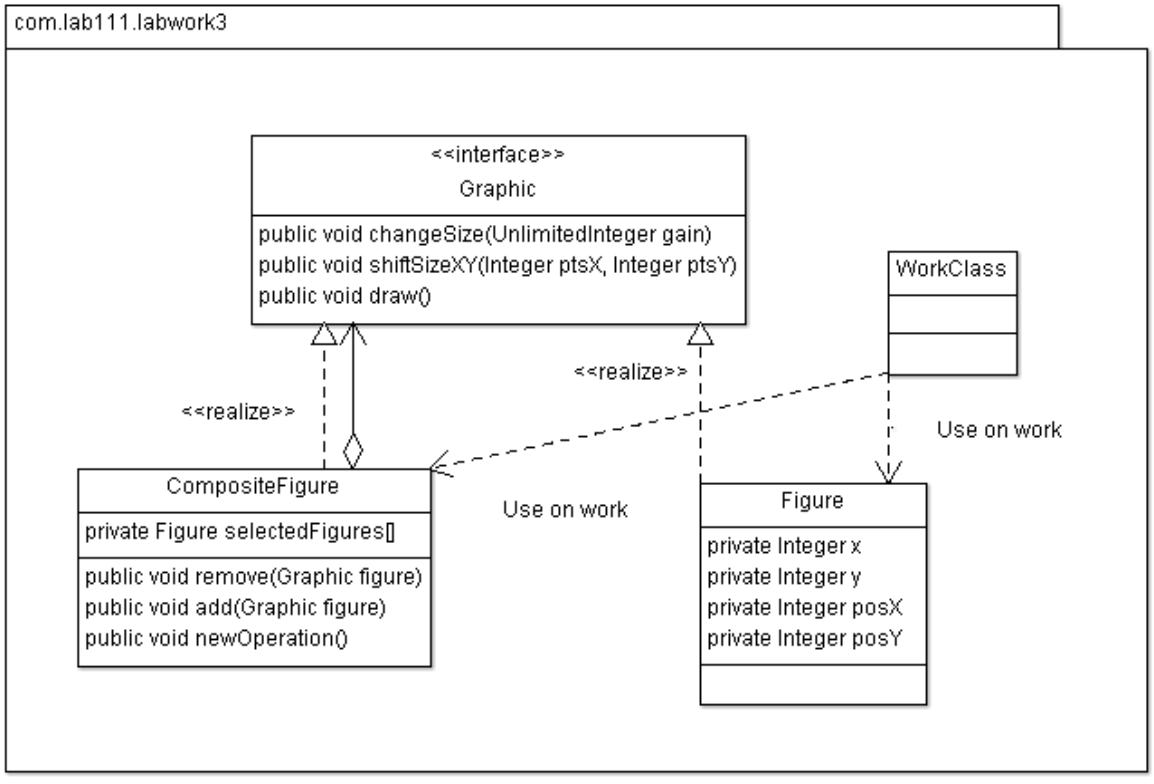
4. В підготованому проекті (ЛР1) створити програмний пакет com.lab111.labwork3. В пакеті розробити інтерфейси і класи, що реалізують завдання (згідно варіанту) з застосуванням одного чи декількох шаблонів (п.3). В розроблюваних класах повністю реалізувати методи, пов'язані з функціюванням Шаблону. Методи, 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);     }
```

5. 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.

Варіанти (3211 mod 12)

7. Визначити специфікації класів для подання графічних маніпуляторів геометричних властивостей(положення, розмір) у редакторі векторної графіки.



```java
package lab111.labwork3;

/**
 * Interface for single and group elements.
 * Realise patern "Composite". Its a component part.
 *
 * @author Error_404
 */
public interface Graphic {

    /**
     * Prints position X/Y of all selected figure.
     */
    public void draw();

    /**
     * Shifts a figure on <code>ptsX</code>/<code>ptsY</code> points to the right/up if
<code>pts</code>
     * is more than 0, to the left/down, if <code>ptsX</code>/<code>ptsy</code> less than
0.
     * @param ptsX defines on how many points to shift on X scale a figure and the
direction of its shift
     * @param ptsY defines on how many points to shift on Y scale a figure and the
direction of its shift
     */
    public void shiftSizeXY(int ptsX, int ptsY);

    /**
     * Increases/reduces the figure sizes by the <code>gain</code> of percent.
     * @param gain defines on how many percent to increase/reduce the figure sizes.
     */
    public void changeSize(double gain);

}
```

```java
package lab111.labwork3;

import java.util.ArrayList;
import java.util.List;

/**
 * Class for group of objects.
 * Realise pattern "Composite". Its a composite part.
 *
 * @author Error_404
 */
public class CompositeFigure implements Graphic {

    /**
     * Collection of selected figures, which you allocate on your graphic redactor GUI.
     */
    private List<Graphic> selectedFigures = new ArrayList<Graphic>();

    @Override
    public void shiftSizeXY(int ptsX, int ptsY) {
        for (Graphic graphic : selectedFigures) {
            graphic.shiftSizeXY(ptsX,ptsY);
        }
    }

    @Override
    public void changeSize(double gain) {
        for (Graphic graphic : selectedFigures) {
            graphic.changeSize(gain);
        }
    }

    /**
```

```
     * Prints positions of all selected/allocated figures. This method works,
     * when you check figure position in your graphic redactor GUI.
     */
    @Override
    public void draw() {
        for (Graphic graphic : selectedFigures) {
            graphic.draw();
        }
    }

    /**
     * Adds the figure to the composition. This method works,
     * when a certain figure gets to area that you allocate.
     */
     public void add(Graphic graphic) {
        selectedFigures.add(graphic);
    }

    /**
     * Remove the figure from the composition. This method works,
     * when a certain figure leaves from area that you allocate.
     */
    public void remove(Graphic graphic) {
        selectedFigures.remove(graphic);
    }

}



package lab111.labwork3;

/**
 * Class for figure.
 * Realise pattern "Composite". Its a leaf part.
 *
 * @author Error_404
 */
public class Figure implements Graphic {

    /**
     * x/y - size; posX/posY - position of the figure
     */
    private int x,y,posX,posY;

    /**
     * When you create someone figure on your graphic redactor, constructor set x and y
position.
     */
    Figure(int posX, int posY, int x, int y){
        this.posX=posX;
        this.posY=posY;
        this.x=x;
        this.y=y;
    }

    @Override
    public void shiftSizeXY(int ptsX, int ptsY) {
        posX+=ptsX;
        posY+=ptsY;
    }

    @Override
    public void changeSize(double gain) {
            x*=(1+gain);
            y*=(1+gain);
    }

    /**
     * Prints the figure X/Y position. This method works,
```

```
     * when you check figure position in your graphic redactor GUI.
     */
    @Override
    public void draw() {
        System.out.println("Xpos: "+ posX+ " Ypos: "+ posY+ " Xsize: "+ x+ " Ysize: "+ y+
";");
    }

}
```

```
package lab111.labwork3;

/**
 * Only workclass.
 * Realise pattern "Composite". Its a client part.
 *
 * @author Error_404
 */
public class WorkClass {
    public static void main(String[] args) {

        //Create on your field 4 figures for manipulations.
        Figure figure1 = new Figure(1,1,2,2);
        Figure figure2 = new Figure(2,2,2,2);
        Figure figure3 = new Figure(3,3,2,2);
        Figure figure4 = new Figure(0,0,23,23);

        //3 different areas of allocation. I.e. it is meant that the user,
        // allocating a certain area of a field of editing, touches different sets of
figures.
        CompositeFigure area1 = new CompositeFigure();
        CompositeFigure area2 = new CompositeFigure();

        //Business methods for "Leaf"/single figure. If we check single figure properties.
        figure1.draw();
        System.out.println("fig1\n");
        figure3.changeSize(0.75);
        figure3.shiftSizeXY(1,-1);
        figure3.draw();
        System.out.println("fig3 changed\n");

        //Imitation of allocation of some figures. I.e. it is meant that we, allocating,
        // we take figures 1,2,3. They, respectively, are added in our field of the
allocated figures.
        area1.add(figure1);
        area1.add(figure2);
        area1.add(figure3);

        //Now we work with the allocated group of figures. But we address to it,
        // as with a single figure. Just "Composition" allows to make such counter.
        area1.draw();
        System.out.println("area1\n");
        area1.shiftSizeXY(2,2);
        area1.changeSize(-0.5);
        area1.draw();
        System.out.println("area1 changed\n");

        area2.add(area1);
        area2.add(figure4);
        area2.draw();
        System.out.println("area2\n");
    }
}
```