Національний технічний університет України

«Київський політехнічний інститут»

Факультет інформатики та обчислювальної техніки

Звіт з лабораторної роботи 4

з дисципліни

«[Технології розробки програмного забезпечення - 1. Основи розробки програмного забезпечення на платформі Java](http://wiki.kpi.ua/index.php/Технології%20розробки%20програмного%20забезпечення%20-%201.%20Основи%20розробки%20програмного%20забезпечення%20на%20платформі%20Microsoft.NET/Java/Node.js_(20202020))»

Виконав:

студент 3-го курсу

групи ІП-52, ФІОТ

Шуліков Дмитро Дмитрович

5224

Київ 2017

# Завдання

* 1. Закріпити призначення шаблонів проектування ПЗ, їх класифікацію. Знати назву і коротку характеристику кожного з шаблонів, що відносяться до певного класу.
  2. Повторити структурні шаблони проектування ПЗ. Знати загальну характеристику структурних шаблонів та призначення кожного з них.
  3. Детально вивчити структурні шаблони проектування Flyweight, Adapter,

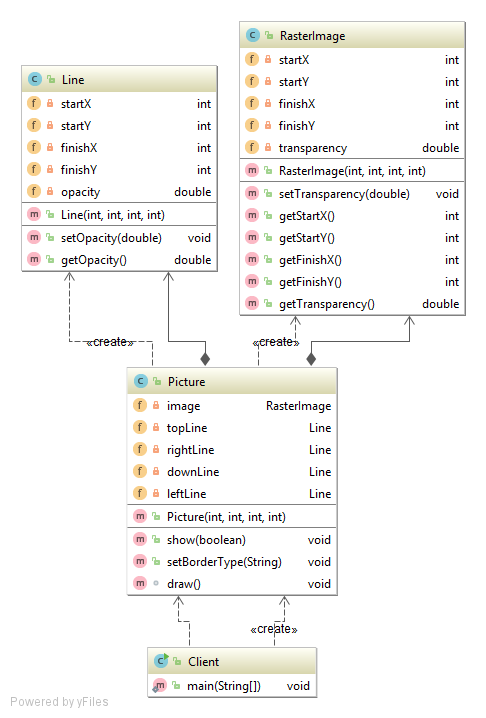
Bridge, Facade. Для кожного з них:

* + - 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.
  1. В підготованому проекті (ЛР1) створити програмний пакет com.lab111.labwork4. В пакеті розробити інтерфейси і класи, що реалізують завдання (згідно варіанту) з застосуванням одного чи декількох шаблонів (п.З). У класах, що розробляються, 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.
  2. 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 10. 5224 mod 11 = 10**

10.Визначити специфікації класів, які подають графічні об'єкти у редакторі векторної графіки – лінія (Line) та растрове зображення (Image). Інтерфейс лінії містить метод setOpacity(double op), що регулює рівень її непрозорості між повністю непрозорою (ор – 1.0) та невидимою (ор – 0.0). Інтерфейс растрового зображення містить метод setTransparency (double tr), що регулює рівень її прозорості між повністю прозорою (tr – 1.0) та повністю непрозорою (tr – 0.0). Задати підсистему з зображення та ліній, які його обрамляють. Забезпечити можливість вмикання/вимикання відображення підсистеми через метод show(boolean vis), та зміни типу обрамлення (горизонтальне, вертикальне, повне, тощо) без впливу на користувача такої підсистеми.

**ДІАГРАМА КЛАСІВ**



**ПРОГРАМНИЙ КОД**

RasterImage.java

**package** com.lab111.labwork4;  
  
*/\*\*  
 \** ***@author*** *dima\_  
 \*/***public class** RasterImage {  
 **private int startX**, **startY**, **finishX**, **finishY**;  
 **private double transparency** = 0;  
  
 */\*\*  
 \*  
 \** ***@param startX*** *\** ***@param startY*** *\** ***@param finishX*** *\** ***@param finishY*** *\*/* **public** RasterImage(**int** startX, **int** startY, **int** finishX, **int** finishY) {  
 **this**.**startX** = startX;  
 **this**.**startY** = startY;  
 **this**.**finishX** = finishX;  
 **this**.**finishY** = finishY;  
 }  
  
 */\*\*  
 \*  
 \** ***@param tr*** *transparency  
 \*/* **public void** setTransparency (**double** tr) {  
 **this**.**transparency** = tr;  
 }  
  
 **public int** getStartX() {  
 **return startX**;  
 }  
  
 **public int** getStartY() {  
 **return startY**;  
 }  
  
 **public int** getFinishX() {  
 **return finishX**;  
 }  
  
 **public int** getFinishY() {  
 **return finishY**;  
 }  
  
 */\*\*  
 \*  
 \** ***@return*** *transparency  
 \*/* **public double** getTransparency() {  
 **return transparency**;  
 }  
}

Picture.java

**package** com.lab111.labwork4;  
  
*/\*\*  
 \** ***@author*** *dima\_  
 \* Facade class  
 \*/***public class** Picture {  
 **private** RasterImage **image**;  
 **private** Line **topLine**;  
 **private** Line **rightLine**;  
 **private** Line **downLine**;  
 **private** Line **leftLine**;  
  
 */\*\*  
 \*  
 \** ***@param startX*** *\** ***@param startY*** *\** ***@param finishX*** *\** ***@param finishY*** *\*/* **public** Picture(**int** startX, **int** startY, **int** finishX, **int** finishY) {  
 **this**.**image** = **new** RasterImage(startX, startY, finishX, finishY);  
  
 **this**.**topLine** = **new** Line(**image**.getStartX() - 1,  
 **image**.getStartY() + 1,  
 **image**.getFinishX() + 1,  
 **image**.getStartY() + 1);  
  
 **this**.**leftLine** = **new** Line(**image**.getStartX() - 1,  
 **image**.getStartY(),  
 **image**.getStartX() - 1,  
 **image**.getFinishY());  
  
 **this**.**downLine** = **new** Line(**image**.getStartX() - 1,  
 **image**.getFinishY() - 1,  
 **image**.getFinishX() + 1,  
 **image**.getFinishY() -1);  
  
 **this**.**rightLine** = **new** Line(**image**.getFinishX() + 1,  
 **image**.getStartY(),  
 **image**.getFinishX() + 1,  
 **image**.getFinishY());  
 }  
  
 */\*\*  
 \*  
 \** ***@param vis*** *\*/* **public void** show(**boolean** vis) {  
 **if** (vis) {  
 **this**.**image**.setTransparency(0);  
 **this**.**topLine**.setOpacity(1);  
 **this**.**rightLine**.setOpacity(1);  
 **this**.**leftLine**.setOpacity(1);  
 **this**.**downLine**.setOpacity(1);  
 } **else** {  
 **this**.**image**.setTransparency(1);  
 **this**.**topLine**.setOpacity(0);  
 **this**.**rightLine**.setOpacity(0);  
 **this**.**leftLine**.setOpacity(0);  
 **this**.**downLine**.setOpacity(0);  
 }  
 }  
  
 */\*\*  
 \*  
 \** ***@param type*** *of border  
 \*/* **public void** setBorderType(String type) {  
 **switch**(type) {  
 **case "horizontal"**:  
 **this**.**topLine**.setOpacity(1);  
 **this**.**rightLine**.setOpacity(0);  
 **this**.**leftLine**.setOpacity(0);  
 **this**.**downLine**.setOpacity(1);  
 **break**;  
 **case "vertical"**:  
 **this**.**topLine**.setOpacity(0);  
 **this**.**rightLine**.setOpacity(1);  
 **this**.**leftLine**.setOpacity(1);  
 **this**.**downLine**.setOpacity(0);  
 **break**;  
 **case "left"**:  
 **this**.**topLine**.setOpacity(0);  
 **this**.**rightLine**.setOpacity(0);  
 **this**.**leftLine**.setOpacity(1);  
 **this**.**downLine**.setOpacity(0);  
 **break**;  
 **case "right"**:  
 **this**.**topLine**.setOpacity(0);  
 **this**.**rightLine**.setOpacity(1);  
 **this**.**leftLine**.setOpacity(0);  
 **this**.**downLine**.setOpacity(0);  
 **break**;  
 **case "full"**:  
 **this**.**topLine**.setOpacity(1);  
 **this**.**rightLine**.setOpacity(1);  
 **this**.**leftLine**.setOpacity(1);  
 **this**.**downLine**.setOpacity(1);  
 **break**;  
 **default**:  
 **this**.**topLine**.setOpacity(0);  
 **this**.**rightLine**.setOpacity(0);  
 **this**.**leftLine**.setOpacity(0);  
 **this**.**downLine**.setOpacity(0);  
 **break**;  
 }  
 }  
  
 */\*\*  
 \* draw Picture  
 \*/* **void** draw() {  
 System.***out***.println(**"Рисование картины..."**);  
 System.***out***.println(**"RasterImage with transparency = "** + **this**.**image**.getTransparency());  
 System.***out***.println(**"TopLine with opacity = "** + **this**.**topLine**.getOpacity());  
 System.***out***.println(**"RightLine with opacity = "** + **this**.**rightLine**.getOpacity());  
 System.***out***.println(**"DownLine with opacity = "** + **this**.**downLine**.getOpacity());  
 System.***out***.println(**"LeftLine with opacity = "** + **this**.**leftLine**.getOpacity());  
 System.***out***.println();  
 }  
}

Line.java

**package** com.lab111.labwork4;  
  
*/\*\*  
 \** ***@author*** *dima\_  
 \*/***public class** Line {  
 **private int startX**, **startY**, **finishX**, **finishY**;  
 **private double opacity** = 1;  
  
 */\*\*  
 \*  
 \** ***@param startX*** *\** ***@param startY*** *\** ***@param finishX*** *\** ***@param finishY*** *\*/* **public** Line(**int** startX, **int** startY, **int** finishX, **int** finishY) {  
 **this**.**startX** = startX;  
 **this**.**startY** = startY;  
 **this**.**finishX** = finishX;  
 **this**.**finishY** = finishY;  
 }  
  
 */\*\*  
 \*  
 \** ***@param op*** *opacity  
 \*/* **public void** setOpacity(**double** op) {  
 **this**.**opacity** = op;  
 }  
  
 */\*\*  
 \*  
 \** ***@return*** *opacity  
 \*/* **public double** getOpacity() {  
 **return opacity**;  
 }  
}

Client.java

**package** com.lab111.labwork4;  
  
*/\*\*  
 \** ***@author*** *dima\_  
 \** ***@version*** *1.0  
 \*/***public class** Client {  
 **public static void** main(String[] args) {  
 Picture picture = **new** Picture(1, 4, 5, 2);  
 picture.draw();  
  
 picture.show(**false**);  
 picture.draw();  
  
 picture.show(**true**);  
 picture.draw();  
  
 picture.setBorderType(**"horizontal"**);  
 picture.draw();  
 }  
}

**ВИСНОВОК**

Під час виконання даної лабораторної роботи я детально ознайомився зі структурними шаблонами проектування Flyweight, Adapter, Bridge, Facade.

Закріпив знання реалізацією підсистеми «Картина», з використанням шаблону Facade.