Національний технічний університет України

«Київський політехнічний інститут»

Факультет інформатики та обчислювальної техніки

Кафедра обчислювальної техніки

Лабораторна робота №4

З предмету: «Інженерія програмного забезпечення»

Виконав: студент групи ІО-33

Шуркіна А. О.

Перевірив: Антонюк А. І.

Київ 2014р.

**Варіант**

№зк=3330 mod 11=8

Визначити специфікації класів, які подають графічні об'єкти у редакторі векторної графіки (прямокутник) через різні інтерфейси API1 та API2. Забезпечити прозору для користувача можливість заміни реалізації графічних об'єктів. Реалізувати метод малювання елементу.

**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.

4. 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.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:

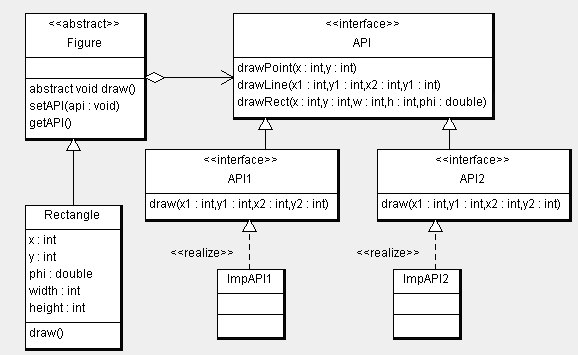
void draw(int x, int y){

    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.

**Діаграма класів**

****

**Код програми**

**package** com.lab111.labwork4;

/\*\*

\*

\* **@author** Nastya Shurkina

\* **@version** 1.8

\*/

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

/\*\*

\* поля реализации

\*/

**private** API api;

/\*\*

\* метод рисования

\*/

**public** **abstract** **void** draw();

/\*\*

\* матод смены реализации

\*

\* **@param** api

\* реализация

\*/

**public** **void** setAPI(API api) {

**this**.api = api;

}

/\*\*

\* возвращает реализацию

\*

\* **@return**

\*/

**public** API getAPI() {

**return** **this**.api;

}

}

**package** com.lab111.labwork4;

/\*\*

\*

\* **@author** Nastya Shurkina

\* **@version** 1.8

\*/

**public** **class** Rectangle **extends** Figure {

**private** **int** x;

**private** **int** y;

**private** **int** width;

**private** **int** height;

**private** **double** phi;

/\*\*

\* конструктор

\*

\* **@param** x

\* центр

\* **@param** y

\* центр

\* **@param** width

\* ширина

\* **@param** height

\* высота

\* **@param** phi

\* угол отклонения от оси х

\* **@param** api

\* реализация

\*/

**public** Rectangle(**int** x, **int** y, **int** width, **int** height, **double** phi, API api) {

setX(x);

setY(y);

setWidth(width);

setHeight(height);

setPhi(phi);

setAPI(api);

}

**public** Rectangle(**int** x, **int** y, **int** width, **int** height, API api) {

**this**(x, y, width, height, 0, api);

}

**public** Rectangle(API api) {

**this**(0, 0, 0, 0, api);

}

/\*\*

\* метод рисования

\*/

**public** **void** draw() {

**if** (getAPI() != **null**)

getAPI().drawRect(x, y, width, height, phi);

}

/\*\*

\* задает х

\*

\* **@param** x

\* центр

\*/

**public** **void** setX(**int** x) {

**this**.x = x;

}

/\*\*

\* задает у

\*

\* **@param** у

\* центр

\*/

**public** **void** setY(**int** y) {

**this**.y = y;

}

/\*\*

\* возвращает х

\*

\* **@return** х

\*/

**public** **int** getX() {

**return** **this**.x;

}

/\*\*

\* возвращает у

\*

\* **@return** у

\*/

**public** **int** getY() {

**return** **this**.y;

}

/\*\*

\* возвращает ширину

\*

\* **@return** ширина

\*/

**public** **int** getWidth() {

**return** **this**.width;

}

/\*\*

\* возвращает высоту

\*

\* **@return** высота

\*/

**public** **int** getHeight() {

**return** **this**.height;

}

/\*\*

\* задает ширину

\*

\* **@param** w

\* ширина

\*/

**public** **void** setWidth(**int** w) {

**this**.width = w;

}

/\*\*

\* задает высоту

\*

\* **@param** h

\* высота

\*/

**public** **void** setHeight(**int** h) {

**this**.height = h;

}

/\*\*

\* задает угол

\*

\* **@param** phi

\* угол

\*/

**public** **void** setPhi(**double** phi) {

**this**.phi = phi;

}

/\*\*

\* возвращает угол

\*

\* **@return** угол

\*/

**public** **double** getPhi() {

**return** **this**.phi;

}

}

**package** com.lab111.labwork4;

/\*\*

\*

\* **@author** Nastya Shurkina

\* **@version** 1.8

\*/

**public** **interface** API {

/\*\*

\* метод рисования точки

\*

\* **@param** x

\* координаты

\* **@param** y

\* координаты

\*/

**public** **void** drawPoint(**int** x, **int** y);

/\*\*

\* метод рисования линии

\*

\* **@param** x1

\* координаты начала

\* **@param** y1

\* координаты начала

\* **@param** x2

\* координаты конца

\* **@param** y2

\* координаты конца

\*/

**public** **void** drawLine(**int** x1, **int** y1, **int** x2, **int** y2);

/\*\*

\* метод рисования прямоугольника

\*

\* **@param** x

\* центр

\* **@param** y

\* центр

\* **@param** width

\* ширина

\* **@param** height

\* высота

\* **@param** phi

\* угол отклонения от оси х

\*/

**public** **void** drawRect(**int** x, **int** y, **int** width, **int** height, **double** phi);

}

**package** com.lab111.labwork4;

/\*\*

\*

\* **@author** Nastya Shurkina

\* **@version** 1.8

\*/

**public** **interface** API1 **extends** API {

/\*\*

\* метод рисавания прямоугольника, фи равно нулю

\*

\* **@param** x1

\* координаты первой вершины

\* **@param** y1

\* координаты первой вершины

\* **@param** x2

\* координаты третей вершины

\* **@param** y2

\* координаты третей вершины

\*/

**public** **void** draw(**int** x1, **int** y1, **int** x2, **int** y2);

}

**package** com.lab111.labwork4;

/\*\*

\*

\* **@author** Nastya Shurkina

\* **@version** 1.8

\*/

**public** **interface** API2 **extends** API {

/\*\*

\* метод рисования линии под произвольным углом

\*

\* **@param** x1

\* начальная координата

\* **@param** y1

\* начальная координата

\* **@param** x2

\* конечная координата

\* **@param** y2

\* конечная координата

\*/

**public** **void** draw(**int** x1, **int** y1, **int** x2, **int** y2);

}

**package** com.lab111.labwork4;

/\*\*

\*

\* **@author** Nastya Shurkina

\* **@version** 1.8

\*/

**public** **class** ImpAPAI1 **implements** API1 {

@Override

**public** **void** draw(**int** x1, **int** y1, **int** x2, **int** y2) {

System.***out***.println("draw rectangle: (" + x1 + "," + y1 + ") (" + x2

+ "," + y1 + ") (" + x2 + "," + y2 + ") (" + x1 + "," + y2

+ ")");

}

@Override

**public** **void** drawLine(**int** x1, **int** y1, **int** x2, **int** y2) {

**int** x = Math.*abs*(x1 - x2);

**int** y = Math.*abs*(y1 - y2);

**if** (x >= y) {

draw(x1, y1, x2, y1);

draw(x2, y1, x2, y2);

} **else** {

draw(x1, y1, x1, y2);

draw(x1, y2, x2, y2);

}

}

@Override

**public** **void** drawPoint(**int** x, **int** y) {

draw(x, y, x, y);

}

@Override

**public** **void** drawRect(**int** x, **int** y, **int** width, **int** height, **double** phi) {

**int** x1 = x - width / 2;

**int** y1 = y - height / 2;

**int** x2 = x + width / 2;

**int** y2 = y + height / 2;

draw(x1, y1, x2, y2);

}

}

**package** com.lab111.labwork4;

/\*\*

\*

\* **@author** Nastya Shurkina

\* **@version** 1.8

\*/

**public** **class** ImpAPI2 **implements** API2 {

@Override

**public** **void** draw(**int** x1, **int** y1, **int** x2, **int** y2) {

System.***out***.println("draw line: begin x=" + x1 + " y=" + y1 + ", end x="

+ x2 + " y=" + y2);

}

@Override

**public** **void** drawLine(**int** x1, **int** y1, **int** x2, **int** y2) {

draw(x1, y1, x2, y2);

}

@Override

**public** **void** drawPoint(**int** x, **int** y) {

draw(x, y, x, y);

}

@Override

**public** **void** drawRect(**int** x, **int** y, **int** width, **int** height, **double** phi) {

**double** cos = Math.*cos*(phi);

**double** sin = Math.*sin*(phi);

**int** x1 = (**int**) (x + width / 2 \* cos);

**int** y1 = (**int**) (y + width / 2 \* sin);

**int** x2 = (**int**) (x - height / 2 \* sin);

**int** y2 = (**int**) (y - height / 2 \* cos);

**int** x3 = (**int**) (x - width \* 2 / cos);

**int** y3 = (**int**) (y - width / 2 \* sin);

**int** x4 = (**int**) (x + height / 2 \* sin);

**int** y4 = (**int**) (y + height / 2 \* cos);

System.***out***.println("draw rectangle:" + " phi=" + phi);

draw(x1, y1, x2, y2);

draw(x2, y2, x3, y3);

draw(x3, y3, x4, y4);

draw(x4, y4, x1, y1);

}

}