Уфимский государственный авиационный технический университет

Отчет по лабораторной работе №7

по дисциплине:

«Объектно-ориентированное программирование»

на тему:

«Группировка и сохранение»

Выполнил: бакалавр гр. ПРО-223

Полякова Е.И.

Проверил: Котельников Виталий Александрович

Уфа 2020

**Цель:** Реализация группировки объектов с помощью паттерна Composite, реализация сохранения и восстановления хранилища из Л.Р.3 с помощью паттерна Abstract Factory.

**Ход работы:**

**1.** Реализация группировки объектов с помощью паттерна Composite.

Для реализации паттерна Composite я создала класс CGroup с методами, необходимыми для группировки объектов, такими как: добавление элемента в группу (AddToGroup), удаление из группы (DelFromGroup) и получение из группы объекта (GetFromGroup). Каждый раз, когда я выделяю объекты на форме, они проходят через эти методы и помещаются в хранилище, образуя группу, с которой можно обращаться, как с единым объектом (изменение размера, положения, цвета).

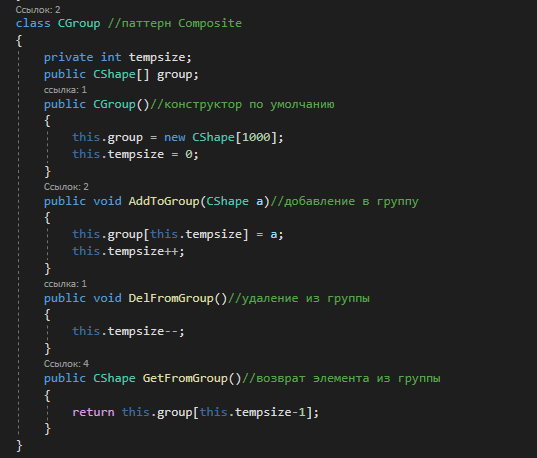
 Пользователь процесс группировки не видит.

Рисунок 1 Паттерн Composite

**2.** Реализация сохранения и восстановления хранилища из Л.Р.3 с помощью паттерна Abstract Factory.

Для реализации паттерна Abstract Factory я создала абстрактный класс (AbstractFactory) и двух его потомков (InPutFile, OutPutFile), которые определяют методы, объявленные в классе-предке. С помощью этого паттерна я реализовала сохранение и восстановление хранилища через текстовый файл. Каждый класс-потомок определяет метод, соответствующий его названию.



Рисунок 2 Паттерн Abstract Factory

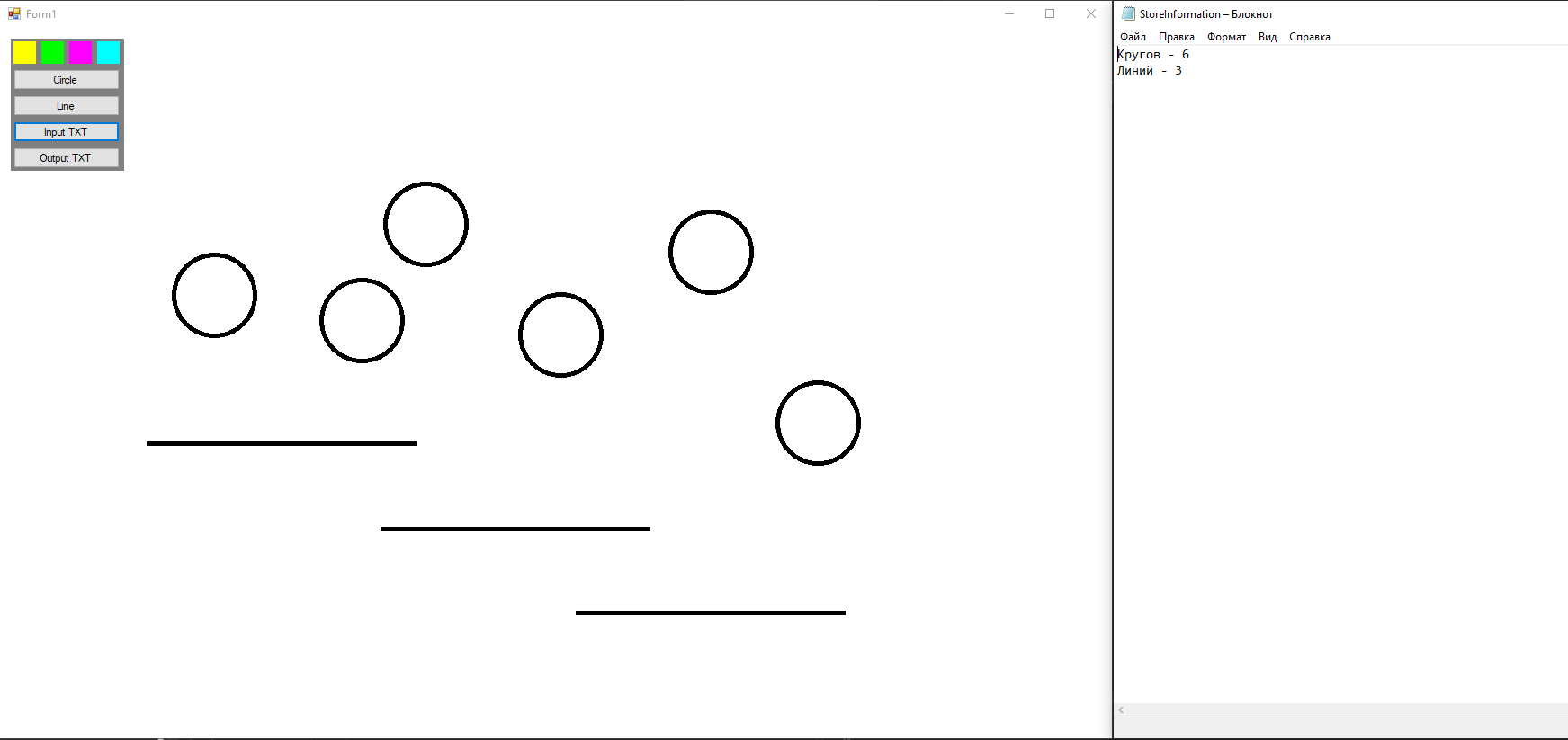
**Пример работы программы**

Рисунок 4 Вывод из текстового файла

Рисунок 3 Вывод в текстовый файл

**Вывод:** Я реализовала группировку объектов с помощью паттерна Composite, а также сохранение и восстановление хранилища из Л.Р.3 с помощью паттерна Abstract Factory.

**Ссылка на репозиторий:** [**https://github.com/Schneiderrrr/LABA6**](https://github.com/Schneiderrrr/LABA6)

**Приложение**

using System;

using System.Collections.Generic;

using System.ComponentModel;

using System.Data;

using System.Drawing;

using System.Drawing.Drawing2D;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

using System.Windows.Forms;

using System.IO;

namespace LABA6

{

public partial class Form1 : Form

{

Graphics gra;//объявляем графику

Pen pen, pentemp;//контур

public int choice = 1;//выбор крга или линии

public Form1()

{

InitializeComponent();

gra = Drawing.CreateGraphics();//привязываем графику к панели

pen = new Pen(Color.Black, 5);//изначальный цвет контура

pentemp = new Pen(Color.White, 5);//текущий цвет контура

}

**public abstract class AbstractFactory//паттерн Abstract Factory**

**{**

**protected string name = @"C:\StoreInformation", info;**

**protected string[] infos;**

**public abstract void inputTXT(int CountC, int CountL);**

**public abstract void outputTXT(int CountC, int CountL);**

**}**

**public class InPutFile: AbstractFactory**

**{**

**public InPutFile()**

**{**

**}**

**public override void inputTXT(int CountC, int CountL)**

**{**

**if (File.ReadAllLines(name) != null)**

**{**

**File.Delete(name);**

**this.info = "Кругов - ";**

**this.info += CountC.ToString();**

**this.info += "\n";**

**File.AppendAllText(name, this.info);**

**this.info = "\0";**

**this.info = "Линий - ";**

**this.info += CountL.ToString();**

**this.info += "\n";**

**File.AppendAllText(name, this.info);**

**}**

**}**

**public override void outputTXT(int CountC, int CountL)**

**{**

**throw new NotImplementedException();**

**}**

**}**

**class OutPutFile : AbstractFactory**

**{**

**public OutPutFile()**

**{**

**}**

**public override void inputTXT(int CountC, int CountL)**

**{**

**throw new NotImplementedException();**

**}**

**public override void outputTXT(int CountC, int CountL)**

**{**

**this.infos = File.ReadAllLines(name);**

**this.info = this.infos[0].Remove(0, 9);**

**CountC += Int32.Parse(this.info);**

**this.info = this.infos[1].Remove(0, 8);**

**CountL += Int32.Parse(this.info);**

**this.CountCircle = CountC;**

**this.CountLine = CountL;**

**}**

**}**

**class CGroup //паттерн Composite**

**{**

**private int tempsize;**

**public CShape[] group;**

**public CGroup()**

**{**

**this.group = new CShape[1000];**

**this.tempsize = 0;**

**}**

**public void AddToGroup(CShape a)**

**{**

**this.group[this.tempsize] = a;**

**this.tempsize++;**

**}**

**public void DelFromGroup()**

**{**

**this.tempsize--;**

**}**

**public CShape GetFromGroup()**

**{**

**return this.group[this.tempsize-1];**

**}**

**}**

class CShape//класс кругов и линий

{

public GraphicsPath circlepath;

public Pen circlepen;

public int x, y, rad;

private int xc, yc;

public CShape()//конструктор по умолчанию

{

this.x = Cursor.Position.X;

this.y = Cursor.Position.Y;

this.rad = 90;

this.circlepen = new Pen(Color.White, 5);

this.circlepath = new GraphicsPath();

}

public bool CircleIsOutside(int xt, int yt, Panel a)//контроль границ круга

{

if ((xt + this.rad <= a.Width) && (yt + this.rad <= a.Height) && (xt - this.rad >= 0) && (yt - this.rad) >= 0)

{

return false;

}

else return true;

}

public bool LineIsOutside(int xt, int yt, Panel a)//контроль границ линии

{

if ((xt <= a.Width) && (xt + 300 <= a.Width) && (yt <= a.Height) && (xt >= 0) && (yt >= 0))

return false;

else return true;

}

public void ChangeLine(int xt)//изменение размера линии

{

this.circlepath.Reset();

this.circlepath.AddLine(this.x, this.y, this.x + xt, this.y);

}

public void DrawShape(Graphics g, Color penc, int c)//отрисовка круга и линии

{

this.circlepen.Color = penc;

if (c == 1)

{

this.xc = this.x - this.rad;

this.yc = this.y - this.rad;

this.circlepath.AddEllipse(this.xc, this.yc, this.rad, this.rad);

}

else

{

this.circlepath.AddLine(this.x, this.y, this.x + 300, this.y);

}

g.DrawPath(this.circlepen, this.circlepath);

}

public void ChangeColor(Color a, Graphics g)//изменение цвета

{

this.circlepen.Color = a;

g.DrawPath(this.circlepen, this.circlepath);

}

~CShape()//деструктор

{

}

}

class KatesStorage : CShape//мое хранилище

{

private int size = 0;

private bool flag;

public int CountCircle = 0, CountLine = 0;

private const int maxsize = 1000;

private CShape[] arr;

private CShape musor = new CShape();

public KatesStorage()//конструктор по умолчанию

{

arr = new CShape[maxsize];

}

public bool empty()//проверка пустоту

{

flag = false;

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

if (this.arr[i] != this.musor)

flag = true;

if (flag == false)

return true;

else return false;

}

public void add(CShape a, int c)//добавление элемента

{

this.arr[this.size] = a;

this.size++;

if (c == 1)

this.CountCircle++;

else this.CountLine++;

}

public void del(CShape a, int c)//удаление элемента

{

for (int i = 0; i < this.size; i++)

if (this.arr[i] == a)

{

if (c == 1)

this.CountCircle--;

else this.CountLine--;

this.arr[i] = this.musor;

}

this.size--;

}

public CShape top()//возвращает верхний элемент

{

if (empty() == false)

return this.arr[this.size - 1];

else return null;

}

public void DelAll(Graphics g, int c)//удаление всех элементов

{

int i = 0;

while (this.size != 0)

{

this.arr[i].DrawShape(g, Color.White, c);

this.arr[i] = null;

i++;

this.size--;

}

}

public CShape search(int xt, int yt, int choice)//поиск необходимого элемента

{

this.flag = false;

int iS = 0;

if (choice == 1)

{

for (int i = 0; i < this.size; i++)

{

if ((this.arr[i].circlepath.IsVisible(xt, yt) == true) && (this.arr[i] != this.musor))

{

iS = i;

flag = true;

}

}

}

else

{

for (int i = 0; i < this.size; i++)

{

if ((this.arr[i].circlepath.IsOutlineVisible(xt, yt, this.circlepen) == true) && (this.arr[i] != this.musor))

{

iS = i;

flag = true;

}

}

}

if (flag == false)

return null;

else return this.arr[iS];

}

~KatesStorage()//деструктор

{

this.arr = null;

}

}

InPutFile inp = new InPutFile();

OutPutFile outp = new OutPutFile();

CGroup group = new CGroup();

KatesStorage store = new KatesStorage();

KatesStorage selected = new KatesStorage();

private void pictureBox1\_Click(object sender, EventArgs e)//здесь задаю текущий цвет

{

PictureBox p = (PictureBox)sender;

pen.Color = p.BackColor;

while (selected.empty() == false)

{

selected.top().ChangeColor(pen.Color, gra);

selected.del(selected.top(), choice);

}

}

private void Form1\_KeyDown(object sender, KeyEventArgs e)//проверка нажатия клавиш клавиатуры

{

if (e.KeyCode == Keys.Delete)//удаление

selected.DelAll(gra, choice);

else if (e.KeyCode == Keys.Subtract)//уменьшение

{

while (selected.empty() == false)

{

selected.top().DrawShape(gra, Color.White, choice);

if (choice == 1)

{

selected.top().rad /= 2;

selected.top().circlepath.Reset();

selected.top().DrawShape(gra, Color.Red, choice);

}

else

{

selected.top().circlepath.Reset();

selected.top().ChangeLine(100);

}

selected.top().ChangeColor(pentemp.Color, gra);

selected.del(selected.top(), choice);

}

}

else if (e.KeyCode == Keys.Add)//увеличение

{

while (selected.empty() == false)

{

if (choice == 1)

{

selected.top().rad \*= 2;

if (selected.top().CircleIsOutside(selected.top().x, selected.top().y, Drawing) == false && selected.top().LineIsOutside(selected.top().x, selected.top().y, Drawing) == false)

{

selected.top().DrawShape(gra, Color.White, choice);

selected.top().circlepath.Reset();

selected.top().DrawShape(gra, Color.Red, choice);

selected.top().ChangeColor(pentemp.Color, gra);

selected.del(selected.top(), choice);

}

else

{

MessageBox.Show("Не влезет");

selected.top().rad /= 2;

break;

}

}

else

{

selected.top().DrawShape(gra, Color.White, choice);

selected.top().circlepath.Reset();

selected.top().ChangeLine(500);

selected.top().ChangeColor(pentemp.Color, gra);

selected.del(selected.top(), choice);

}

}

}

else if (e.KeyCode == Keys.W)//переместить вверх

{

while (selected.empty() == false)

{

selected.top().y -= 50;

if (selected.top().CircleIsOutside(selected.top().x, selected.top().y, Drawing) == false && selected.top().LineIsOutside(selected.top().x, selected.top().y, Drawing) == false)

{

selected.top().DrawShape(gra, Color.White, choice);

selected.top().circlepath.Reset();

selected.top().DrawShape(gra, Color.Red, choice);

selected.top().ChangeColor(pentemp.Color, gra);

selected.del(selected.top(), choice);

}

else

{

MessageBox.Show("Не влезет");

selected.top().y += 50;

break;

}

}

}

else if (e.KeyCode == Keys.S)//переместить вниз

{

while (selected.empty() == false)

{

selected.top().y += 50;

if (selected.top().CircleIsOutside(selected.top().x, selected.top().y, Drawing) == false && selected.top().LineIsOutside(selected.top().x, selected.top().y, Drawing) == false)

{

selected.top().DrawShape(gra, Color.White, choice);

selected.top().circlepath.Reset();

selected.top().DrawShape(gra, Color.Red, choice);

selected.top().ChangeColor(pentemp.Color, gra);

selected.del(selected.top(), choice);

}

else

{

MessageBox.Show("Не влезет");

selected.top().y -= 50;

break;

}

}

}

else if (e.KeyCode == Keys.A)//переместить влево

{

while (selected.empty() == false)

{

selected.top().x -= 50;

if (selected.top().CircleIsOutside(selected.top().x, selected.top().y, Drawing) == false && selected.top().LineIsOutside(selected.top().x, selected.top().y, Drawing) == false)

{

selected.top().DrawShape(gra, Color.White, choice);

selected.top().circlepath.Reset();

selected.top().DrawShape(gra, Color.Red, choice);

selected.top().ChangeColor(pentemp.Color, gra);

selected.del(selected.top(), choice);

}

else

{

MessageBox.Show("Не влезет");

selected.top().x += 50;

break;

}

}

}

else if (e.KeyCode == Keys.D)//переместить вправо

{

while (selected.empty() == false)

{

selected.top().x += 50;

if (selected.top().CircleIsOutside(selected.top().x, selected.top().y, Drawing) == false && selected.top().LineIsOutside(selected.top().x, selected.top().y, Drawing) == false)

{

selected.top().DrawShape(gra, Color.White, choice);

selected.top().circlepath.Reset();

selected.top().DrawShape(gra, Color.Red, choice);

selected.top().ChangeColor(pentemp.Color, gra);

selected.del(selected.top(), choice);

}

else

{

MessageBox.Show("Не влезет");

selected.top().x -= 50;

break;

}

}

}

}

private void button1\_Click(object sender, EventArgs e)//выбрали круг

{

choice = 1;

}

private void button2\_Click(object sender, EventArgs e)//выбрали линию

{

choice = 2;

}

private void button3\_Click(object sender, EventArgs e)

{

inp.inputTXT(store.CountCircle, store.CountLine);

}

private void button4\_Click(object sender, EventArgs e)

{

outp.outputTXT(store.CountCircle, store.CountLine);

int x = 250, y = 250;

for (int i = 0; i < store.CountCircle; i++)

{

CShape a = new CShape();

a.x = x;

a.y = y;

a.DrawShape(gra, pen.Color, 1);

x += 50;

store.add(a, 1);

}

x = 200;

y = 200;

for (int i = 0; i < store.CountLine; i++)

{

CShape a = new CShape();

a.x = x;

a.y = y;

a.DrawShape(gra, pen.Color, 2);

y += 50;

store.add(a, 2);

}

}

private void Drawing\_MouseDown(object sender, MouseEventArgs e)//здесь создаем круги

{

if (store.search(e.X, e.Y, choice) != null)//проверка попадания в какой-либо круг

{

if (Control.ModifierKeys == Keys.Control)//выделение кругов

{

pentemp.Color = store.search(e.X, e.Y, choice).circlepen.Color;

store.search(e.X, e.Y, choice).ChangeColor(Color.Red, gra);

group.AddToGroup(store.search(e.X, e.Y, choice));

selected.add(group.GetFromGroup(), choice);

}

else

{

while (selected.empty() == false)

{

group.GetFromGroup().ChangeColor(pentemp.Color, gra);

selected.del(group.GetFromGroup(), choice);

group.DelFromGroup();

}

pentemp.Color = store.search(e.X, e.Y, choice).circlepen.Color;

store.search(e.X, e.Y, choice).ChangeColor(Color.Red, gra);

group.AddToGroup(store.search(e.X, e.Y, choice));

selected.add(group.GetFromGroup(), choice);

}

}

else

{

CShape circle = new CShape();

if (circle.CircleIsOutside(e.X, e.Y, Drawing) == false && circle.LineIsOutside(e.X, e.Y, Drawing) == false)

{

circle.DrawShape(gra, pen.Color, choice);

store.add(circle, choice);

}

else

{

MessageBox.Show("Выход за границы");

}

}

}

}

}