Министерство науки и высшего образования Российской Федерации

Федеральное государственное бюджетное

образовательное учреждение высшего образования

«Уфимский государственный авиационный технический университет»

Факультет информатики и робототехники

Кафедра вычислительной математики и кибернетики

Отчет по лабораторной работе №7

по дисциплине «Объектно-ориентированное программирование»

на тему: «Группировка и сохранение»

Выполнил:

Студент группы ПРО-228Б И.И.Гареев

Проверил: В.А. Котельников

Уфа – 2022

Содержание

[1 Цель работы 3](#_Toc94725638)

[2 Программная часть 4](#_Toc94725639)

[2.1 Базовый класс Figure 4](#_Toc94725640)

[2.2 Класс круга CCircle, класс квадрата Square и класс треугольника Triangle 4](#_Toc94725641)

[2.3 Хранилище 4](#_Toc94725642)

[2.4 Класс CGroup 4](#_Toc94725643)

[2.5 Событие panel2\_MouseClick 5](#_Toc94725644)

[2.6 Cобытие btn\_groop\_Click 5](#_Toc94725645)

[2.7 Cобытие btn\_ungroop\_Click 6](#_Toc94725646)

[2.8 Событие btn\_save\_Click 7](#_Toc94725647)

[2.9 Событие btn\_load\_Click 7](#_Toc94725648)

[2.10 Событие panel2\_Paint 8](#_Toc94725649)

[3 Заключение 9](#_Toc94725650)

[4 Приложение А 10](#_Toc94725651)

# Цель работы

Развитие приложения из Л.Р.6:

* Группировка, разгруппировка объектов и групп объектов (сгруппированные объекты изменяются только совместно).
* Реализация сохранения и восстановления из файла

# Программная часть

## **Базовый класс Figure**

Создаю базовый класс Figure как в Лр 6, в который добавляем методы Save для сохранения объектов в файл, Load для восстановления объектов из файла.

## Класс круга CCircle, класс квадрата Square и класс треугольника Triangle

Создаю классы CCircle, Square, Triangle, потомки класса Figure как в ЛР6, в которые добавляем методы Save (запись в файл), Load (выгрузка из файла).

## Хранилище

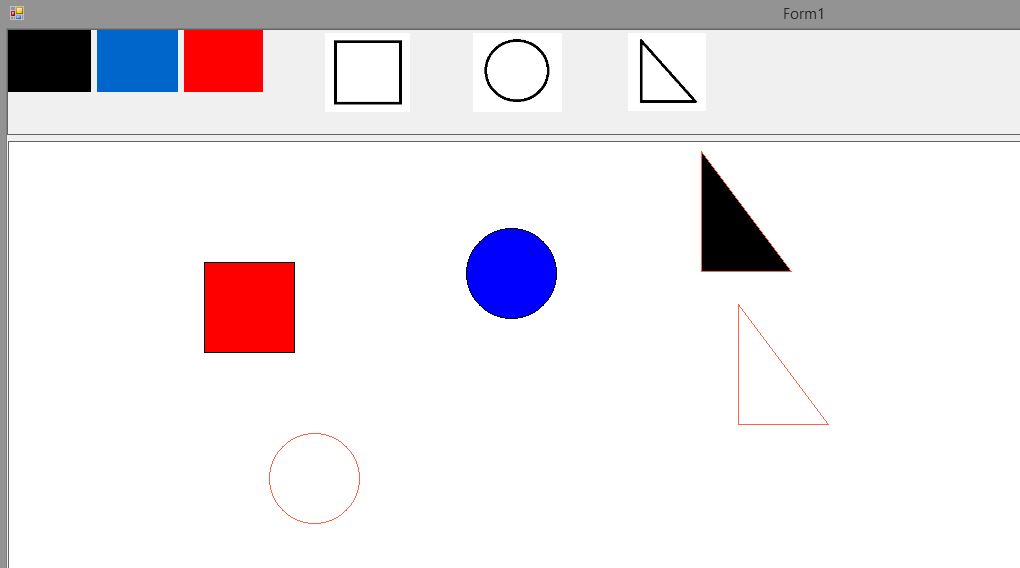
Как в Лр 3 создаю класс MyStorage, в котором есть конструктор без параметров, конструктор с параметрами, метод SetObject и AddObject для добавления объектов, метод DeleteObject для удаления объектов и DeleteCheckObject для удаления выделенных объектов. Также в хранилище есть переменная типа bool isCheckedStorage, в которой я проверяю нажатие на круг, и если true, то вызываю у объекта метод DoCheckedTrue, метод NotChecked, который снимает выделение с объектов, метод Draw, который вызывает у объектов метод Draw, метод Move, проверяющий, если объект выделен, то вызывается у него метод Move, метод ChangeSize, проверяющий, если объект выделен, то вызывается у него метод ChangeSize, метод ChangeColor, проверяющий, если объект выделен, то вызывается у него метод ChangeColor. Добавляем метод createGroup, в котором группируются объекты, deleteGroup, который удаляет группу объектов. Метод saveAll для сохранения хранилища в файл, loadAll для восстаноления из файла.

## Класс CGroup

Создаю класс CGroup, потомок класса Figure, в котором есть конструктор для создания группы объектов, метод Move, который вызывает метод Move у объектов группы, метод Draw, который вызывает у объектов метод Draw, метод Size, который вызывает метод Size у объектов группы, метод ChangeColor. Также добавляем методы Save (запись в файл), Load (выгрузка из файла).

## Событие panel2\_MouseClick

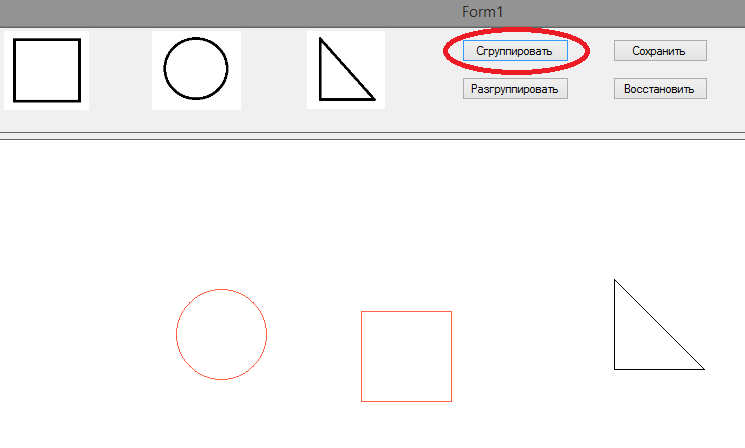
Проверяется условие, если нажато на пустое место, то с помощью switch выбирается какая фигура будет нарисована, в зависимости на какой pictureBox нажали, иначе если зажата клавиша ctrl, то определенные круги выделяются, иначе снимаю выделение и, проверяю нажатие на pictureBox в котором определены цвета и соответственно меняю цвет.



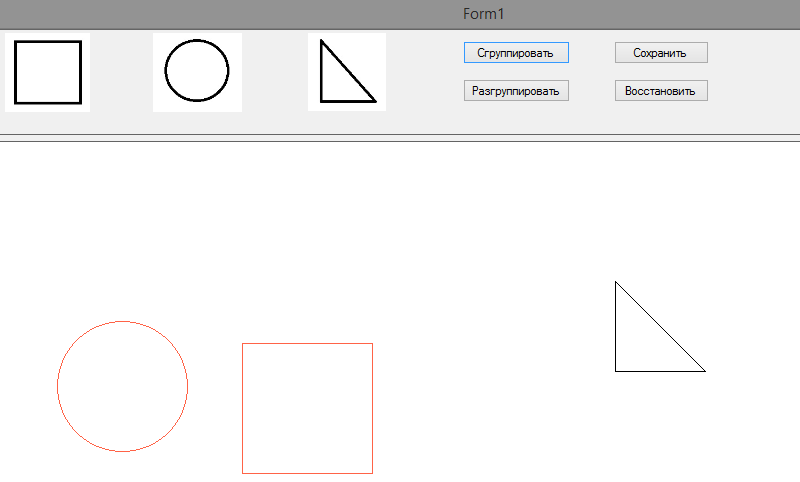
Рисунок

## Cобытие btn\_groop\_Click

При нажатии на кнопку группировки вызывается метод создания группы createGroup и объекты объединяются в одну группу.



Рисунок

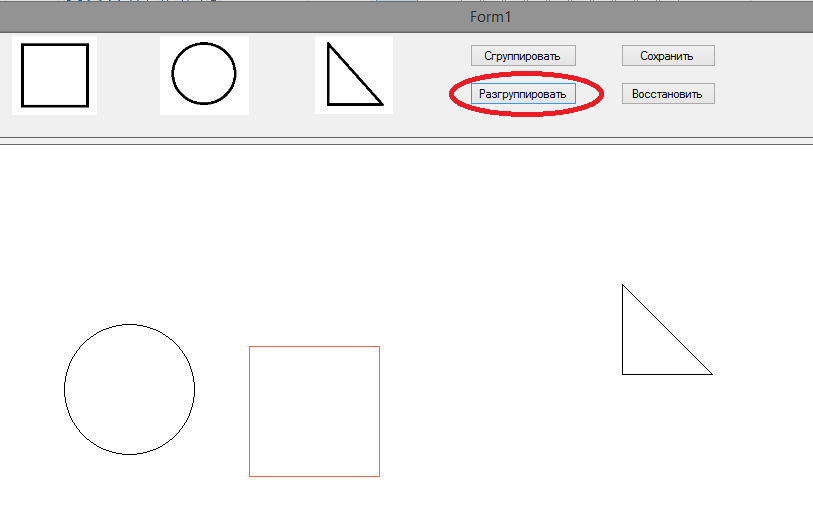


Рисунок

На Рисунок 3 видно, что при перемещении и изменении размера, перемещаются и меняют размер сразу обе фигуры.

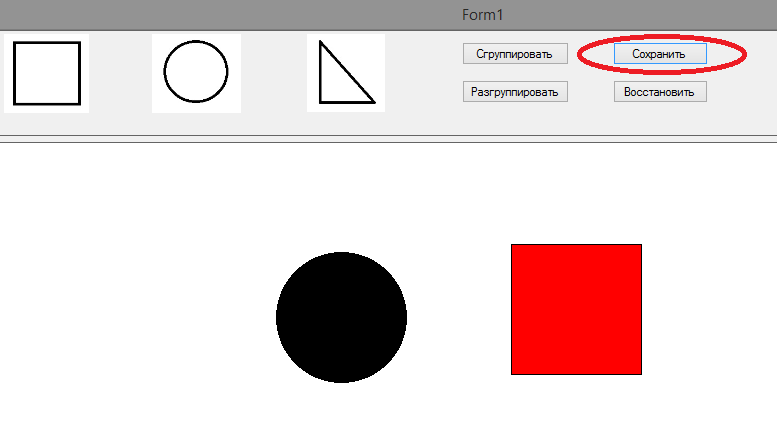
## Cобытие btn\_ungroop\_Click

При нажатии на кнопку разгруппировки вызывается метод удаления группы deleteGroup и выделенная группа распадается.



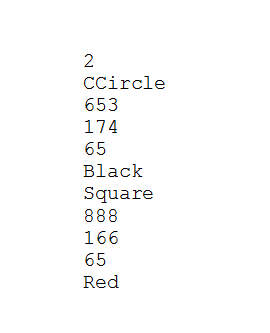
Рисунок

## Событие btn\_save\_Click



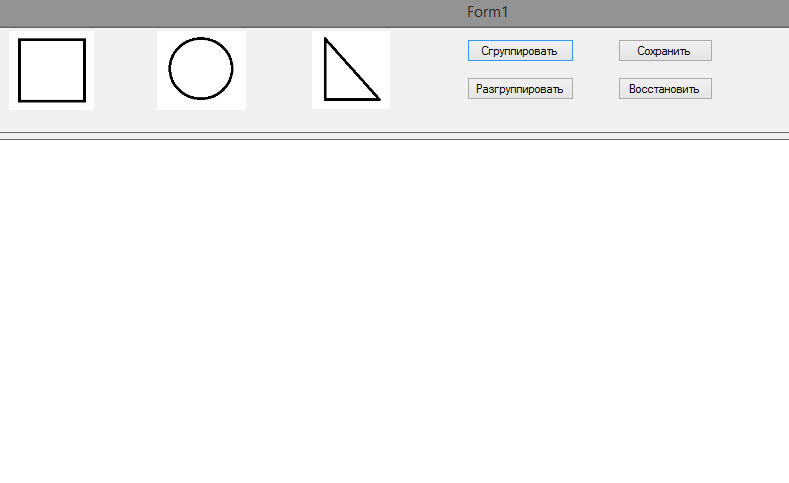
Рисунок

При нажатии на кнопку сохранить (Рисунок 5) объекты записываются в текстовый файл save.txt (Рисунок 6).



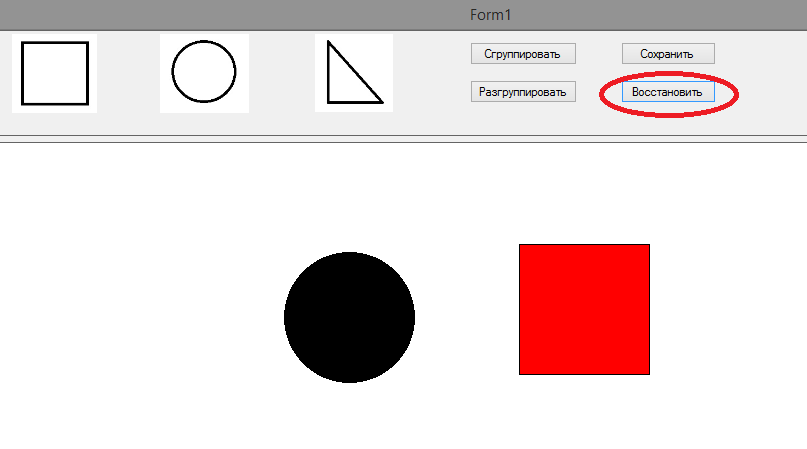
Рисунок

## Событие btn\_load\_Click



Рисунок

При нажатии на кнопку «Восстановить» фигуры восстанавливаются из файла (Рисунок 8).



Рисунок

## Событие panel2\_Paint

Здесь просто рисуются фигуры.

# Заключение

В ходе данной лабораторной работы я дополнил ЛР6, реализовав группировку, разгруппировку объектов. Также реализовал сохранение и восстановление из файла.

<https://github.com/IlyasGareev/LR6>

# Приложение А

using System;

using System.Collections.Generic;

using System.ComponentModel;

using System.Data;

using System.Drawing;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

using System.Windows.Forms;

using System.IO;

namespace LR6

{

public partial class Form1 : Form

{

public Form1()

{

InitializeComponent();

this.KeyPreview = true;

this.KeyDown += Form1\_KeyDown;

}

class Figure //базовый класс

{

protected Color color = Color.White;

virtual public bool isClicked(MouseEventArgs e)

{

return true;

}

virtual public void Draw(Panel panel1, Graphics g)

{

}

virtual public void DoCheckTrue()

{

}

virtual public void DoCheckFalse()

{

}

virtual public bool isChecked()

{

return true;

}

virtual public void Move(KeyEventArgs e)

{

}

virtual public void ChangeSize(KeyEventArgs e)

{

}

virtual public void ChangeColor(int color)

{

}

virtual public void Save(StreamWriter file)

{

}

virtual public void Load(StreamReader file)

{

}

virtual public int getCount()

{

return 1;

}

virtual public Figure getObject(int i)

{

Figure a = new Figure();

return a;

}

public int x, y, r;

}

class CCircle : Figure //класс круга

{

private bool Checked;

private Pen pen;

private SolidBrush brush;

public CCircle(int x, int y, int r)

{

this.x = x;

this.y = y;

this.r = r;

Checked = false;

brush = new SolidBrush(Color.White);

}

override public bool isClicked(MouseEventArgs e)

{

if (((e.X - x) \* (e.X - x) + (e.Y - y) \* (e.Y - y)) <= r \* r)

{

return true;

}

else

{

return false;

}

}

public override void Draw(Panel panel1, Graphics g)

{

Rectangle rect = new Rectangle(x - r, y - r, r \* 2, r \* 2);

if (Checked == true)

{

pen = new Pen(Color.Tomato);

}

else

{

pen = new Pen(Color.Black);

}

brush.Color = color;

g.FillEllipse(brush, rect);

g.DrawEllipse(pen, rect);

}

override public void DoCheckTrue()

{

Checked = true;

}

override public void DoCheckFalse()

{

Checked = false;

}

override public bool isChecked()

{

return Checked;

}

override public void Move(KeyEventArgs e)

{

if (e.KeyCode == Keys.A)

{

x = x - 5;

}

if (e.KeyCode == Keys.D)

{

x = x + 5;

}

if (e.KeyCode == Keys.W)

{

y = y - 5;

}

if (e.KeyCode == Keys.S)

{

y = y + 5;

}

}

public override void ChangeSize(KeyEventArgs e)

{

if (e.KeyValue == 187)

{

r = r + 5;

}

if (e.KeyValue == 189)

{

r = r - 5;

}

}

public override void ChangeColor(int col)

{

switch (col)

{

case 1:

brush.Color = Color.Black;

color = Color.Black;

break;

case 2:

brush.Color = Color.Blue;

color = Color.Blue;

break;

case 3:

brush.Color = Color.Red;

color = Color.Red;

break;

}

}

public override void Save(StreamWriter file)

{

file.WriteLine("CCircle");

file.WriteLine(x);

file.WriteLine(y);

file.WriteLine(r);

file.WriteLine(color.ToKnownColor());

}

public override void Load(StreamReader file)

{

x = Convert.ToInt32(file.ReadLine());

y = Convert.ToInt32(file.ReadLine());

r = Convert.ToInt32(file.ReadLine());

color = Color.FromName(file.ReadLine());

}

}

class Square : Figure //класс квадрата

{

private bool Checked;

private SolidBrush brush;

public Square(int x, int y, int r)

{

this.x = x;

this.y = y;

this.r = r;

Checked = false;

brush = new SolidBrush(Color.White);

}

override public bool isClicked(MouseEventArgs e)

{

if (e.X >= x - r && e.Y >= y - r && e.X <= x + r && e.Y <= y + r)

{

return true;

}

else

{

return false;

}

}

public override void Draw(Panel panel1, Graphics g)

{

Rectangle rect = new Rectangle(x - r, y - r, r \* 2, r \* 2);

if (Checked == true)

{

brush.Color = color;

g.FillRectangle(brush, rect);

g.DrawRectangle(Pens.Tomato, rect);

}

else

{

brush.Color = color;

g.FillRectangle(brush, rect);

g.DrawRectangle(Pens.Black, rect);

}

}

override public void Move(KeyEventArgs e)

{

if (e.KeyCode == Keys.A)

{

x = x - 5;

}

if (e.KeyCode == Keys.D)

{

x = x + 5;

}

if (e.KeyCode == Keys.W)

{

y = y - 5;

}

if (e.KeyCode == Keys.S)

{

y = y + 5;

}

}

public override void Save(StreamWriter file)

{

file.WriteLine("Square");

file.WriteLine(x);

file.WriteLine(y);

file.WriteLine(r);

file.WriteLine(color.ToKnownColor());

}

public override void Load(StreamReader file)

{

x = Convert.ToInt32(file.ReadLine());

y = Convert.ToInt32(file.ReadLine());

r = Convert.ToInt32(file.ReadLine());

color = Color.FromName(file.ReadLine());

}

override public void DoCheckTrue()

{

Checked = true;

}

override public void DoCheckFalse()

{

Checked = false;

}

override public bool isChecked()

{

return Checked;

}

public override void ChangeSize(KeyEventArgs e)

{

if (e.KeyValue == 187)

{

r = r + 5;

}

if (e.KeyValue == 189)

{

r = r - 5;

}

}

public override void ChangeColor(int col)

{

switch (col)

{

case 1:

brush.Color = Color.Black;

color = Color.Black;

break;

case 2:

brush.Color = Color.Blue;

color = Color.Blue;

break;

case 3:

brush.Color = Color.Red;

color = Color.Red;

break;

}

}

}

class Triangle : Figure //класс треугольника

{

private Point[] p;

private bool Checked;

private SolidBrush brush;

public Triangle(int x, int y, int r)

{

p = new Point[3];

p[0].X = x;

p[0].Y = y - r;

p[1].X = x;

p[1].Y = y + r;

p[2].X = x + r \* 2;

p[2].Y = y + r;

Checked = false;

this.r = r;

this.x = x;

this.y = y;

brush = new SolidBrush(Color.White);

}

override public bool isClicked(MouseEventArgs e)

{

if (e.X >= x - r && e.Y >= y - r && e.X <= x + r && e.Y <= y + r )

{

return true;

}

else

{

return false;

}

}

public override void Draw(Panel panel1, Graphics g)

{

if (Checked == true)

{

brush.Color = color;

g.FillPolygon(brush, p);

g.DrawPolygon(Pens.Tomato, p);

}

else

{

brush.Color = color;

g.FillPolygon(brush, p);

g.DrawPolygon(Pens.Black, p);

}

}

override public void DoCheckTrue()

{

Checked = true;

}

override public void DoCheckFalse()

{

Checked = false;

}

override public bool isChecked()

{

return Checked;

}

public override void Move(KeyEventArgs e)

{

if (e.KeyCode == Keys.W)

{

p[0].Y = p[0].Y - 5;

p[1].Y = p[1].Y - 5;

p[2].Y = p[2].Y - 5;

}

if (e.KeyCode == Keys.S)

{

p[0].Y = p[0].Y + 5;

p[1].Y = p[1].Y + 5;

p[2].Y = p[2].Y + 5;

}

if (e.KeyCode == Keys.A)

{

p[0].X = p[0].X - 5;

p[1].X = p[1].X - 5;

p[2].X = p[2].X - 5;

}

if (e.KeyCode == Keys.D)

{

p[0].X = p[0].X + 5;

p[1].X = p[1].X + 5;

p[2].X = p[2].X + 5;

}

}

public override void ChangeSize(KeyEventArgs e)

{

if (e.KeyValue == 187)

{

p[0].X = p[0].X;

p[0].Y = p[0].Y - 3;

p[1].X = p[1].X;

p[1].Y = p[1].Y + 3;

p[2].X = p[2].X + 6;

p[2].Y = p[2].Y + 3;

}

if (e.KeyValue == 189)

{

p[0].X = p[0].X;

p[0].Y = p[0].Y + 3;

p[1].X = p[1].X;

p[1].Y = p[1].Y - 3;

p[2].X = p[2].X - 6;

p[2].Y = p[2].Y - 3;

}

}

public override void ChangeColor(int col)

{

switch (col)

{

case 1:

brush.Color = Color.Black;

color = Color.Black;

break;

case 2:

brush.Color = Color.Blue;

color = Color.Blue;

break;

case 3:

brush.Color = Color.Red;

color = Color.Red;

break;

}

}

public override void Save(StreamWriter file)

{

file.WriteLine("Triangle");

file.WriteLine(p[0].X);

file.WriteLine(p[0].Y);

file.WriteLine(p[1].X);

file.WriteLine(p[1].Y);

file.WriteLine(p[2].X);

file.WriteLine(p[2].Y);

file.WriteLine(x);

file.WriteLine(y);

file.WriteLine(r);

file.WriteLine(color.ToKnownColor());

}

public override void Load(StreamReader file)

{

p[0].X = Convert.ToInt32(file.ReadLine());

p[0].Y = Convert.ToInt32(file.ReadLine());

p[1].X = Convert.ToInt32(file.ReadLine());

p[1].Y = Convert.ToInt32(file.ReadLine());

p[2].X = Convert.ToInt32(file.ReadLine());

p[2].Y = Convert.ToInt32(file.ReadLine());

x = Convert.ToInt32(file.ReadLine());

y = Convert.ToInt32(file.ReadLine());

r = Convert.ToInt32(file.ReadLine());

color = Color.FromName(file.ReadLine());

}

}

class Factory

{

public Figure createShape(string name)

{

Figure obj = null;

switch (name)

{

case "CCircle":

obj = new CCircle(0, 0, 0);

break;

case "Square":

obj = new Square(0, 0, 0);

break;

case "CGroup":

obj = new CGroup(0);

break;

case "Triangle":

obj = new Triangle(0, 0, 0);

break;

default:

break;

}

return obj;

}

}

class MyStorage

{

private int size;

public Figure[] storage;

public MyStorage()

{

size = 0;

}

public MyStorage(int size)

{

this.size = size;

storage = new Figure[size];

}

public void SetObjects(int index, Figure obj)

{

storage[index] = obj;

}

public void AddObject(Figure obj)

{

Figure[] new\_storage = new Figure[size + 1];

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

{

new\_storage[i] = storage[i];

}

storage = new\_storage;

storage[size] = obj;

size = size + 1;

}

public void DeleteObject(int index)

{

Figure[] new\_storage = new Figure[size - 1];

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

{

new\_storage[i] = storage[i];

}

for (int i = index; i < size - 1; i++)

{

new\_storage[i] = storage[i + 1];

}

size = size - 1;

storage = new\_storage;

}

public void MakeCheckedbyIndex(int i)

{

storage[i].DoCheckTrue();

}

public void MakenotCheckedbyIndex(int i)

{

storage[i].DoCheckFalse();

}

public bool isCheckedStorage(MouseEventArgs e)

{

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

{

if (storage[i].isClicked(e) == true)

{

storage[i].DoCheckTrue();

return true;

}

}

return false;

}

public void DeleteCheckObject(MyStorage storage)

{

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

{

if (this.storage[i].isChecked() == true)

{

storage.DeleteObject(i);

i = i - 1;

}

}

}

public void NotChecked()

{

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

{

storage[i].DoCheckFalse();

}

}

public void Draw(Panel panel1, Graphics g)

{

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

{

storage[i].Draw(panel1, g);

}

}

public void Move(KeyEventArgs e)

{

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

{

if (storage[i].isChecked() == true)

{

storage[i].Move(e);

}

}

}

public void ChangeSize(KeyEventArgs e)

{

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

{

if (storage[i].isChecked() == true)

{

storage[i].ChangeSize(e);

}

}

}

public void ChangeColor(int col)

{

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

{

if (storage[i].isChecked() == true)

{

storage[i].ChangeColor(col);

}

}

}

public Figure getObject(int i)

{

return storage[i];

}

public void createGroup()

{

int count = 0;

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

{

if (storage[i].isChecked() == true)

count++;

}

if (count >= 2)

{

CGroup group = new CGroup(count);

for (int i = size - 1; i >= 0; i--)

{

if (storage[i].isChecked() == true)

{

group.addShape(storage[i]); DeleteObject(i);

}

}

AddObject(group);

}

}

public void deleteGroup()

{

for (int i = size - 1; i >= 0; i--)

{

if (storage[i] is CGroup && storage[i].isChecked())

{

CGroup group = (CGroup)storage[i];

for (int j = group.getCount() - 1; j >= 0; j--)

{

AddObject(group.\_figures[j]);

}

DeleteObject(i);

}

}

}

public void saveAll()

{

string way = @"save.txt";

StreamWriter file = new StreamWriter(way, false);

file.WriteLine(size);

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

{

storage[i].Save(file);

}

file.Close();

}

public void loadAll()

{

string way = @"save.txt";

Factory factory = new Factory();

StreamReader file = new StreamReader(way);

int count = Convert.ToInt32(file.ReadLine());

string name;

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

{

name = Convert.ToString(file.ReadLine());

AddObject(factory.createShape(name));

if (storage[i] != null)

{

storage[i].Load(file);

}

}

file.Close();

}

public int getCount()

{

return size;

}

}

class CGroup : Figure

{

private int \_count;

private int \_maxcount;

public Figure[] \_figures;

private bool Checked;

public CGroup(int maxcount)

{

Checked = true;

\_maxcount = maxcount; \_count = 0;

\_figures = new Figure[\_maxcount];

}

~CGroup()

{

for (int i = 0; i < \_maxcount; ++i)

{

\_figures = null;

}

\_figures = null;

}

public bool addShape(Figure obj)

{

if (\_count >= \_maxcount)

return false;

\_count++;

\_figures[\_count - 1] = obj;

return true;

}

public override void Move(KeyEventArgs e)

{

for (int i = 0; i < \_count; i++)

{

\_figures[i].Move(e);

}

}

public override void DoCheckFalse()

{

Checked = false;

for (int i = 0; i < \_count; i++)

{

\_figures[i].DoCheckFalse();

}

}

public override void Draw(Panel panel1, Graphics g)

{

for (int i = 0; i < \_count; i++)

{

\_figures[i].Draw(panel1, g);

}

}

public override void ChangeSize(KeyEventArgs e)

{

for (int i = 0; i < \_count; i++)

{

\_figures[i].ChangeSize(e);

}

}

public override void DoCheckTrue()

{

for (int i = 0; i < \_count; i++)

{

\_figures[i].DoCheckTrue();

}

}

public override bool isClicked(MouseEventArgs e)

{

for (int i = 0; i < \_count; i++)

{

if (\_figures[i].isClicked(e) == true)

{

Checked = true;

return true;

}

}

return false;

}

public override bool isChecked()

{

return Checked;

}

public override void ChangeColor(int col)

{

for (int i = 0; i < \_count; i++)

{

\_figures[i].ChangeColor(col);

}

}

public override int getCount()

{

return \_count;

}

public override Figure getObject(int i)

{

return \_figures[i];

}

public override void Save(StreamWriter file)

{

file.WriteLine("CGroup");

file.WriteLine(\_count);

for (int i = 0; i < \_count; i++)

{

\_figures[i].Save(file);

}

}

public override void Load(StreamReader file)

{

\_maxcount = Convert.ToInt32(file.ReadLine());

\_count = 0;

string name;

Factory factory = new Factory();

\_figures = new Figure[\_maxcount];

for (int i = 0; i < \_maxcount; i++)

{

name = Convert.ToString(file.ReadLine());

addShape(factory.createShape(name));

\_figures[i].Load(file);

}

}

}

MyStorage storage = new MyStorage();

int figure;

int col;

private void panel2\_Paint(object sender, PaintEventArgs e)

{

Graphics g = panel2.CreateGraphics();

storage.Draw(panel2, g);

}

private void panel2\_MouseClick(object sender, MouseEventArgs e)

{

if (storage.isCheckedStorage(e) == false)

{

storage.NotChecked();

switch (figure)

{

case 1:

CCircle ellipse = new CCircle(e.X, e.Y, 45);

storage.AddObject(ellipse);

break;

case 2:

Square square = new Square(e.X, e.Y, 45);

storage.AddObject(square);

break;

case 3:

Triangle triangle = new Triangle(e.X, e.Y, 45);

storage.AddObject(triangle);

break;

}

}

else

{

if (Control.ModifierKeys == Keys.Control)

{

storage.isCheckedStorage(e);

}

else

{

storage.NotChecked();

storage.isCheckedStorage(e);

if (col == 1 || col == 2 || col == 3)

{

storage.ChangeColor(col);

}

}

}

Refresh();

}

private void Form1\_KeyDown(object sender, KeyEventArgs e)

{

Graphics g = panel1.CreateGraphics();

if (e.KeyValue == 46)

{

storage.DeleteCheckObject(storage);

g.Clear(Color.White);

}

if (e.KeyCode == Keys.W || e.KeyCode == Keys.S || e.KeyCode == Keys.D || e.KeyCode == Keys.A)

{

g.Clear(Color.White);

storage.Move(e);

}

if (e.KeyValue == 187 || e.KeyValue == 189)

{

g.Clear(Color.White);

storage.ChangeSize(e);

}

Refresh();

}

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

{

figure = 2;

}

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

{

figure = 1;

}

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

{

figure = 3;

}

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

{

col = 1;

}

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

{

col = 2;

}

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

{

col = 3;

}

private void btn\_ungroop\_Click(object sender, EventArgs e)

{

storage.deleteGroup();

}

private void btn\_save\_Click(object sender, EventArgs e)

{

storage.saveAll();

}

private void btn\_load\_Click(object sender, EventArgs e)

{

storage.loadAll();

Refresh();

}

private void btn\_groop\_Click(object sender, EventArgs e)

{

storage.createGroup();

}

}

}