Федеральное агентство по образованию

ФГБОУ ВО Уфимский государственный авиационный технический

университет

Кафедра ВМиК

ЛАБОРАТОРНАЯ РАБОТА №8

По дисциплине: «Объектно-Ориентированное Программирование»

«Дерево объектов, подписка»

Выполнил: студент группы ПРО-222

Сафронов И.А.

Проверил: Котельников В.А.

Уфа 2020

1. **Цель работы:** на основе Л.Р.7 реализовать отображение объектов из хранилища в TreeView и создать липкий объект при помощи паттерна Observer.
2. **Задачи:**

* Добавить на форму приложения объект TreeView для отображения текущего содержания хранилища
* Реализовать синхронизацию объекта TreeView с хранилищем с помощью паттерна Observer, при этом должна выполняться синхронизация в обоих направлениях: при выборе объекта в дереве он должен выбираться в рабочей области и наоборот, при выборе объекта в рабочей области он должен выбираться в дереве.
* Реализовать с помощью паттерна Observer специальный «липкий» вид объекта: при касании/пересечении которого другие объекты «приклеиваются» к нему и при перемещении «липкого» объекта, вместе с ним перемещаются и «приклеенные» объекты.

1. **Ход** **работы**
2. Создаем 3 круга, выделяем их и нажимаем на кнопку сгруппировать. После этого они будут реагировать как один объект. (Рис.1)

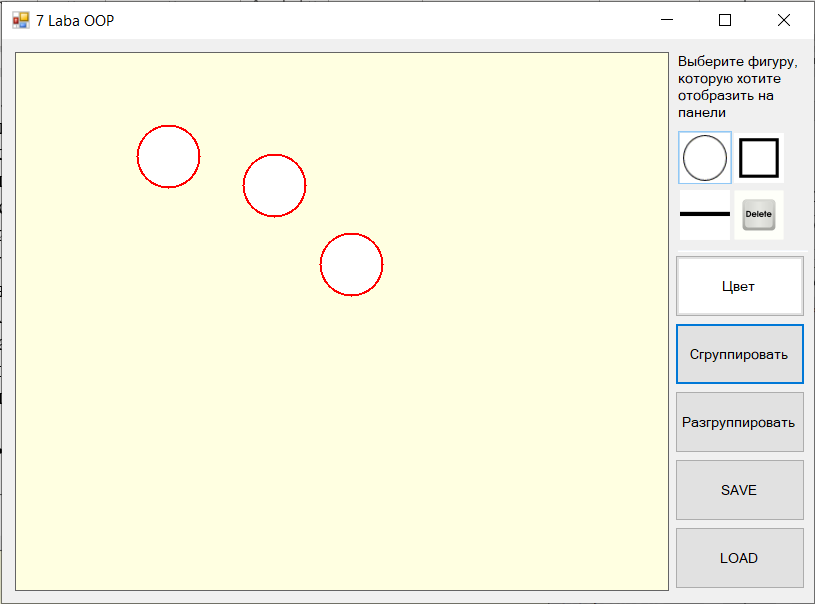


Рис. 1 Демонстрация работы программы

1. Кнопка «Разгруппировать» разделяет объекты группы. Теперь они будут не будут реагировать как один объект. (Рис.2)

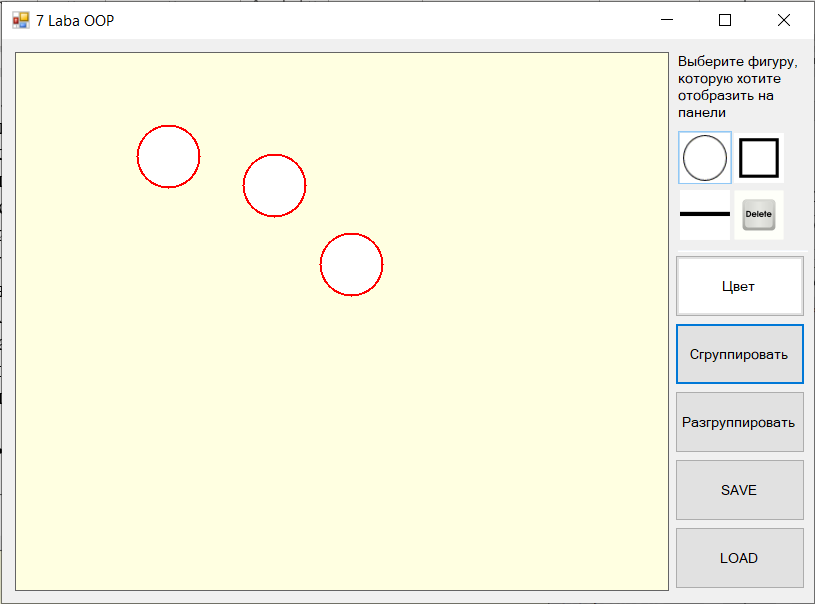


Рис. 2 Демонстрация работы программы

1. Также в одну группу могут входить другие группы. (Рис.3)

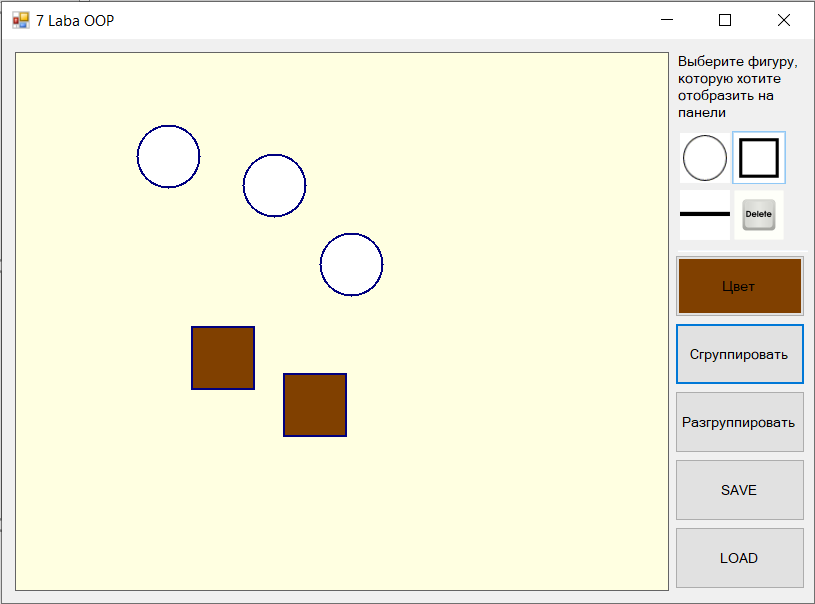


Рис. 3 Демонстрация работы программы

1. Также есть функция чтения из файла и запись в него хранилища. Текст в файле представлен в человеко-читаемом виде. (Рис. 4)

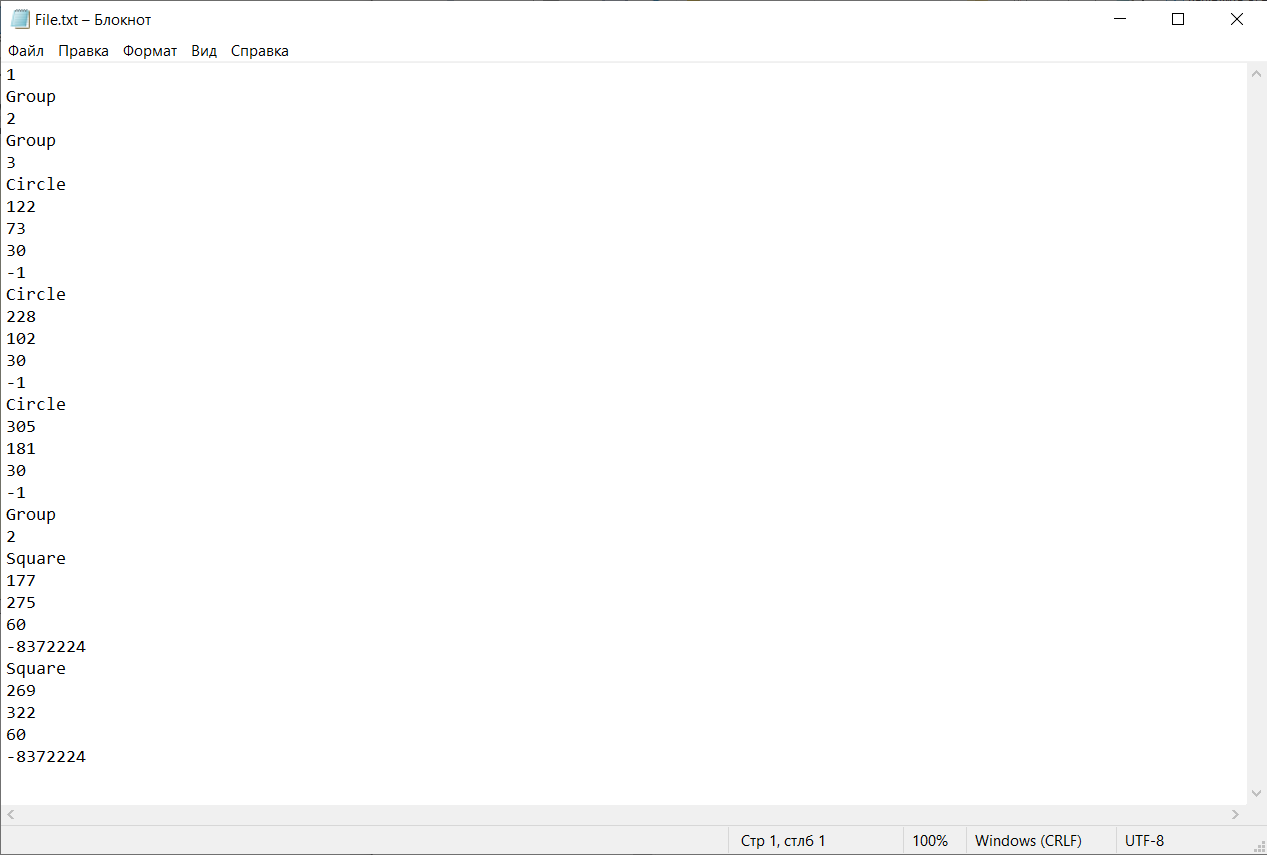


Рис. 4 Демонстрация работы программы

1. При нажатии на кнопку “LOAD” (Рис. 5)

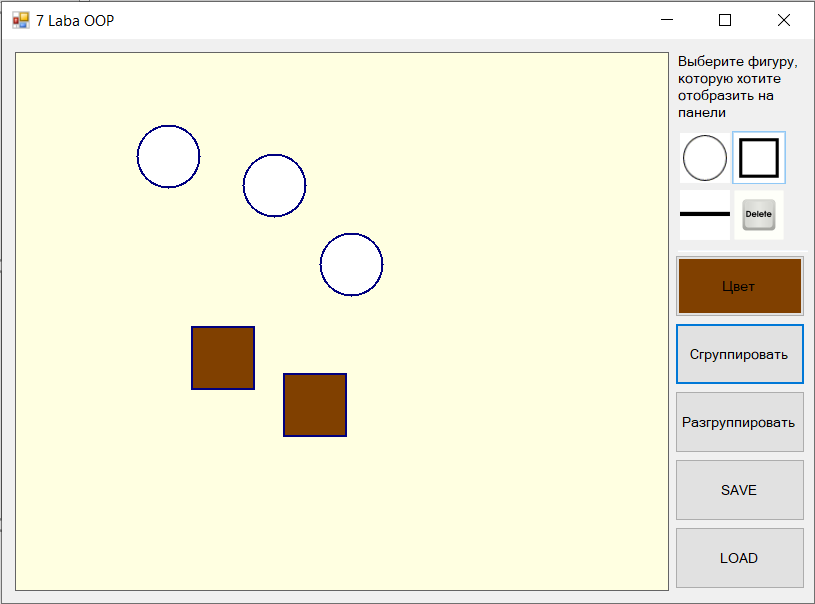


Рис. 5 Демонстрация работы программы

1. **Вывод:**

В рамках лабораторной работы я разобрался с паттернами Composite и Factory Method, научился группировать/разгруппировывать объекты, а также работать с текстовыми файлами.

Приложение 1. Листинг программы:

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 \_7\_laba\_OOP

{

public partial class laba7 : Form

{

public laba7()

{

InitializeComponent();

}

public class Figure

{ // Composite

public int x, y;

public Color color = Color.Navy;

public Color fillcolor;

public Figure() { }

public virtual string save() { return ""; }

public virtual void load(string x, string y, string c, string fillcolor) { }

public virtual void load(ref StreamReader sr, Figure figure, CreateFigure createFigure) { }

public virtual void GroupAddFigure(ref Figure object1) { }

public virtual void UnGroup(ref Storage stg, int c) { }

public virtual void paint\_figure(Pen pen, Panel panel\_drawing) { }

public virtual void move\_x(int x, Panel panel\_drawing) { }

public virtual void move\_y(int y, Panel panel\_drawing) { }

public virtual void changesize(int size) { }

public virtual bool checkfigure(int x, int y) { return false; }

public virtual void setcolor(Color color) { }

public virtual void caseswitch(ref StreamReader sr, ref Figure figure, CreateFigure createFigure) { }

public virtual void get\_min\_x(ref int f) { }

public virtual void get\_max\_x(ref int f) { }

public virtual void get\_min\_y(ref int f) { }

public virtual void get\_max\_y(ref int f) { }

}

class Group : Figure

{ // Группа

public int maxcount = 10;

public Figure[] group;

public int count;

int min\_x=99999, max\_x=0, min\_y=99999, max\_y=0;

public Group()

{ // Выделяем maxcount мест в хранилище

count = 0;

group = new Figure[maxcount];

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

group[i] = null;

}

public override string save()

{ // Функция сохранения

string str = "Group" + "\n" + count;

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

str += "\n" + group[i].save();

return str;

}

public override void load(ref StreamReader sr, Figure figure, CreateFigure createFigure)

{ // Функция загрузки

int chislo = Convert.ToInt32(sr.ReadLine());

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

{

createFigure.caseswitch(ref sr, ref figure, createFigure);

GroupAddFigure(ref figure);

}

}

public override void GroupAddFigure(ref Figure object1)

{ // Добавляет фигуру в группу

if (count >= maxcount)

return;

group[count] = object1;

++count;

}

public override void UnGroup(ref Storage stg, int c)

{ // Разгруппировка

stg.delete\_object(c);

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

{

stg.add\_object(index, ref group[i], k, ref indexin);

}

}

public override void paint\_figure(Pen pen, Panel panel\_drawing)

{ // Отображение группы

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

{

group[i].paint\_figure(pen, panel\_drawing);

}

}

public void getsize()

{

min\_x = 99999; max\_x = 0; min\_y = 99999; max\_y = 0;

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

{

int f = 0;

group[i].get\_min\_x(ref f);

if (f < min\_x)

min\_x = f;

group[i].get\_max\_x(ref f);

if (f > max\_x)

max\_x = f;

group[i].get\_min\_y(ref f);

if (f < min\_y)

min\_y = f;

group[i].get\_max\_y(ref f);

if (f > max\_y)

max\_y = f;

}

}

public override void move\_x(int x, Panel panel\_drawing)

{ // Перемещение по оси x

getsize();

if ((min\_x + x) > 0 && (max\_x + x) < panel\_drawing.ClientSize.Width)

{

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

{

group[i].move\_x(x, panel\_drawing);

}

}

}

public override void get\_min\_x(ref int f)

{

f = min\_x;

}

public override void get\_max\_x(ref int f)

{

f = max\_x;

}

public override void get\_min\_y(ref int f)

{

f = min\_y;

}

public override void get\_max\_y(ref int f)

{

f = max\_y;

}

public override void move\_y(int y, Panel panel\_drawing)

{ // Перемещение по оси y

getsize();

if ((min\_y + y) > 0 && (max\_y + y) < panel\_drawing.ClientSize.Height)

{

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

{

group[i].move\_y(y, panel\_drawing);

}

}

}

public override void changesize(int size)

{ // Изменение размера

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

{

group[i].changesize(size);

}

}

public override bool checkfigure(int x, int y)

{ // Проверка на фигуры

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

{

if(group[i].checkfigure(x, y))

return true;

}

return false;

}

public override void setcolor(Color color)

{ // Установка цвета

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

{

group[i].setcolor(color);

}

}

}

class Circle: Figure

{

public int rad; // Радиус круга

public Circle() { }

public Circle(int x, int y, int rad)

{

this.rad = rad;

this.x = x - rad;

this.y = y - rad;

}

public override string save()

{ // Функция сохранения

return "Circle" + "\n" + x + "\n" + y + "\n" + rad + "\n" + fillcolor.ToArgb().ToString();

}

public override void load(string x, string y, string rad, string fillcolor)

{ // Функция загрузки

this.x = Convert.ToInt32(x);

this.y = Convert.ToInt32(y);

this.rad = Convert.ToInt32(rad);

this.fillcolor = Color.FromArgb(Convert.ToInt32(fillcolor));

}

public override void paint\_figure(Pen pen, Panel panel\_drawing)

{ // Отображение фигуры

SolidBrush figurefillcolor = new SolidBrush(fillcolor);

panel\_drawing.CreateGraphics().DrawEllipse(

pen, x, y, rad \* 2, rad \* 2);

panel\_drawing.CreateGraphics().FillEllipse(

figurefillcolor, x, y, rad \* 2, rad \* 2);

}

public override void get\_min\_x(ref int f)

{

f = x;

}

public override void get\_max\_x(ref int f)

{

f = x + (rad\*2);

}

public override void get\_min\_y(ref int f)

{

f = y;

}

public override void get\_max\_y(ref int f)

{

f = y + (rad \* 2);

}

public override void move\_x(int x, Panel panel\_drawing)

{ // Перемещение по оси x

int c = this.x + x;

int gran = panel\_drawing.ClientSize.Width - (rad \* 2);

check(c, x, gran, gran - 2, ref this.x);

}

public override void move\_y(int y, Panel panel\_drawing)

{ // Перемещение по оси y

int c = this.y + y;

int gran = panel\_drawing.ClientSize.Height - (rad \* 2);

check(c, y, gran, gran - 2, ref this.y);

}

public override void changesize(int size)

{ // Изменение размера

rad += size;

}

public override bool checkfigure(int x, int y)

{ // Проверка на фигуры

return ((x - this.x - rad) \* (x - this.x - rad) + (y - this.y - rad) \*

(y - this.y - rad)) < (rad \* rad);

}

public override void setcolor(Color color)

{ // Установка цвета

fillcolor = color;

}

}

class Line: Figure

{

public int lenght = 60;

public int wight = 10;

public Line() { }

public Line(int x, int y)

{

this.x = x - lenght/2;

this.y = y;

}

public override string save()

{ // Функция сохранения

return "Line" + "\n" + x + "\n" + y + "\n" + lenght + "\n" + fillcolor.ToArgb().ToString();

}

public override void load(string x, string y, string lenght, string fillcolor)

{ // Функция загрузки

this.x = Convert.ToInt32(x);

this.y = Convert.ToInt32(y);

this.lenght = Convert.ToInt32(lenght);

this.fillcolor = Color.FromArgb(Convert.ToInt32(fillcolor));

}

public override void paint\_figure(Pen pen, Panel panel\_drawing)

{ // Отображение фигуры

SolidBrush figurefillcolor = new SolidBrush(fillcolor);

panel\_drawing.CreateGraphics().DrawRectangle(pen, x,

y, lenght, wight);

panel\_drawing.CreateGraphics().FillRectangle(figurefillcolor, x,

y, lenght, wight);

}

public override void get\_min\_x(ref int f)

{

f = x;

}

public override void get\_max\_x(ref int f)

{

f = x + lenght;

}

public override void get\_min\_y(ref int f)

{

f = y;

}

public override void get\_max\_y(ref int f)

{

f = y + wight;

}

public override void move\_x(int x, Panel panel\_drawing)

{ // Перемещение по оси x

int l = this.x + x;

int gran = panel\_drawing.ClientSize.Width - lenght;

check(l, x, gran, --gran, ref this.x);

}

public override void move\_y(int y, Panel panel\_drawing)

{ // Перемещение по оси y

int l = this.y + y;

int gran = panel\_drawing.ClientSize.Height - wight;

check(l, y, gran, --gran, ref this.y);

}

public override void changesize(int size)

{ // Изменение размера

lenght += size;

}

public override bool checkfigure(int x, int y)

{ // Проверка на фигуры

return (this.x <= x && x <= (this.x + lenght) && (this.y - 2) <= y &&

y <= (this.y + wight));

}

public override void setcolor(Color color)

{ // Установка цвета

fillcolor = color;

}

}

class Square: Figure

{

public int size = 60;

public Square() { }

public Square(int x, int y)

{

this.x = x - size/2;

this.y = y - size/2;

}

public override string save()

{ // Функция сохранения

return "Square" + "\n" + x + "\n" + y + "\n" + size + "\n" + fillcolor.ToArgb().ToString();

}

public override void load(string x, string y, string size, string fillcolor)

{ // Функция загрузки

this.x = Convert.ToInt32(x);

this.y = Convert.ToInt32(y);

this.size = Convert.ToInt32(size);

this.fillcolor = Color.FromArgb(Convert.ToInt32(fillcolor));

}

public override void paint\_figure(Pen pen, Panel panel\_drawing)

{ // Отображение фигуры

SolidBrush figurefillcolor = new SolidBrush(fillcolor);

panel\_drawing.CreateGraphics().DrawRectangle(pen,

x, y, size, size);

panel\_drawing.CreateGraphics().FillRectangle(figurefillcolor,

x, y, size, size);

}

public override void get\_min\_x(ref int f)

{

f = x;

}

public override void get\_max\_x(ref int f)

{

f = x + size;

}

public override void get\_min\_y(ref int f)

{

f = y;

}

public override void get\_max\_y(ref int f)

{

f = y + size;

}

public override void move\_x(int x, Panel panel\_drawing)

{ // Перемещение по оси x

int s = this.x + x;

int gran = panel\_drawing.ClientSize.Width - size;

check(s, x, gran, --gran, ref this.x);

}

public override void move\_y(int y, Panel panel\_drawing)

{ // Перемещение по оси y

int s = this.y + y;

int gran = panel\_drawing.ClientSize.Height - size;

check(s, y, gran, --gran, ref this.y);

}

public override void changesize(int size)

{ // Изменение размера

this.size += size;

}

public override bool checkfigure(int x, int y)

{ // Проверка на фигуры

return (this.x <= x && x <= (this.x + size) &&

this.y <= y && y <= (this.y + size));

}

public override void setcolor(Color color)

{ // Установка цвета

fillcolor = color;

}

}

public class CreateFigure: Figure

{ // Используем Factory Method

public override void caseswitch(ref StreamReader sr, ref Figure figure, CreateFigure createFigure)

{

string str = sr.ReadLine();

switch (str)

{ // В зависимости какая фигура выбрана

case "Circle":

figure = new Circle();

figure.load(sr.ReadLine(), sr.ReadLine(), sr.ReadLine(), sr.ReadLine());

break;

case "Line":

figure = new Line();

figure.load(sr.ReadLine(), sr.ReadLine(), sr.ReadLine(), sr.ReadLine());

break;

case "Square":

figure = new Square();

figure.load(sr.ReadLine(), sr.ReadLine(), sr.ReadLine(), sr.ReadLine());

break;

case "Group":

figure = new Group();

figure.load(ref sr, figure, createFigure);

break;

}

}

}

static public void check(int f, int chislo, int gran, int gran1, ref int x)

{ // Проверка на выход фигуры за границы

if (f > 0 && f < gran)

x += chislo;

else

{

if (f <= 0)

x = 1;

else

if (f >= gran1)

x = gran1;

}

}

int p = 0; // Нажат ли был ранее Ctrl

static int k = 10; // Кол-во ячеек в хранилище

Storage storag = new Storage(k); // Создаем объект хранилища

static int index = 0; // Кол-во нарисованных фигур

static int indexin = 0; // Индекс, в какое место была помещена фигура

int figure\_now = 1; // Какая фигура выбрана

public class Storage

{

public Figure[] objects;

public Storage(int count)

{ // Выделяем count мест в хранилище

objects = new Figure[count];

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

objects[i] = null;

}

public void initialisat(int count)

{ // Выделяем count мест в хранилище

objects = new Figure[count];

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

objects[i] = null;

}

public void add\_object(int ind, ref Figure object1, int count, ref int indexin)

{ // Добавляет ячейку в хранилище

// Если ячейка занята ищем свободное место

while (objects[ind] != null)

{

ind = (ind + 1) % count;

}

objects[ind] = object1;

indexin = ind;

}

public void delete\_object(int ind)

{ // Удаляет объект из хранилища

objects[ind] = null;

if (index > 0)

index--;

}

public bool check\_empty(int index)

{ // Проверяет занято ли место хранилище

if (objects[index] == null)

return true;

else return false;

}

public int occupied(int size)

{ // Определяет кол-во занятых мест в хранилище

int count\_occupied = 0;

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

if (!check\_empty(i))

++count\_occupied;

return count\_occupied;

}

public void doubleSize(ref int size)

{ // Функция для увеличения кол-ва элементов в хранилище в 2 раза

Storage storage1 = new Storage(size \* 2);

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

storage1.objects[i] = objects[i];

for (int i = size; i < (size \* 2) - 1; ++i)

storage1.objects[i] = null;

size = size \* 2;

initialisat(size);

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

objects[i] = storage1.objects[i];

}

~Storage() { }

};

private void panel\_drawing\_MouseClick(object sender, MouseEventArgs e)

{

//Проверка на наличие фигуры на данных координатах

int c = check\_figure(ref storag, k, e.X, e.Y);

if (c != -1)

{ // Если на этом месте уже нарисована фигура

if (Control.ModifierKeys == Keys.Control)

{ // Если нажат ctrl, то выделяем несколько объектов

if (p == 0)

{

paint\_figure(Color.Navy, 4, ref storag, indexin);

p = 1;

}

// Вызываем функцию отрисовки фигуры

paint\_figure(Color.Red, 4, ref storag, c);

}

else

{ // Иначе выделяем только один объект

// Снимаем выделение у всех объектов хранилища

remove\_selection\_circle(ref storag);

paint\_figure(Color.Red, 4, ref storag, c);

}

return;

}

Figure figure = new Figure();

switch (figure\_now)

{ // В зависимости какая фигура выбрана

case 0:

return;

case 1:

figure = new Circle(e.X, e.Y, 30);

break;

case 2:

figure = new Line(e.X, e.Y);

break;

case 3:

figure = new Square(e.X, e.Y);

break;

}

if (index == k)

storag.doubleSize(ref k);

// Добавляем фигуру в хранилище

storag.add\_object(index, ref figure, k, ref indexin);

// Снимаем выделение у всех объектов хранилища

remove\_selection\_circle(ref storag);

storag.objects[indexin].fillcolor = colorDialog1.Color;

paint\_figure(Color.Red, 4, ref storag, indexin);

++index;

p = 0;

}

private void remove\_selection\_circle(ref Storage stg)

{ // Снимает выделение у всех элементов хранилища

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

{

if (!stg.check\_empty(i))

{ // Вызываем функцию отрисовки круга

paint\_figure(Color.Navy, 4, ref storag, i);

}

}

}

private void move\_y(ref Storage stg, int y)

{ // Функция для перемещения фигур по оси Y

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

{

if (!stg.check\_empty(i))

{

if(stg.objects[i].color == Color.Red)

{ // Если объект выделен

stg.objects[i].move\_y(y, panel\_drawing);

}

}

}

}

private void move\_x(ref Storage stg, int x)

{ // Функция для перемещения фигур по оси X

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

{

if (!stg.check\_empty(i))

{

if (stg.objects[i].color == Color.Red)

{ // Если объект выделен

stg.objects[i].move\_x(x, panel\_drawing);

}

}

}

}

private void changesize(ref Storage stg, int size)

{ // Увеличивает или уменьшает размер фигур, в зависимости от size

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

{

if (!stg.check\_empty(i))

{ // Если под i индексом в хранилище есть объект

if (stg.objects[i].color == Color.Red)

{

stg.objects[i].changesize(size);

}

}

}

}

private void remove\_selected\_circle(ref Storage stg)

{ // Удаляет выделенные элементы из хранилища

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

{

if (!stg.check\_empty(i))

{

if (stg.objects[i].color == Color.Red)

stg.delete\_object(i);

}

}

}

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

{ // Очистка хранилища и панели

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

{

if (!storag.check\_empty(i))

{

storag.delete\_object(i);

}

}

panel\_drawing.Refresh();

}

private void paint\_figure(Color name, int size, ref Storage stg, int index)

{ // Рисует фигуру на панели

// Объявляем объект - карандаш, которым будем рисовать контур

if (!stg.check\_empty(index))

{

Pen pen = new Pen(name, size);

stg.objects[index].color = name;

stg.objects[index].paint\_figure(pen, panel\_drawing);

}

}

private void paint\_all(ref Storage stg)

{ // Рисует все фигуры на панели

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

if (!stg.check\_empty(i))

paint\_figure(stg.objects[i].color, 4, ref storag, i);

}

private int check\_figure(ref Storage stg, int size, int x, int y)

{ // Проверяет есть ли уже фигура с такими же координатами в хранилище

if (stg.occupied(size) != 0)

{

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

{

if (!stg.check\_empty(i))

{ // Если под i индексом в хранилище есть объект

if (stg.objects[i].checkfigure(x, y))

return i;

}

}

}

return -1;

}

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

{

drawline.Checked = false;

drawsquare.Checked = false;

figure\_now = 1;

if (drawellipse.Checked == false) // Если не выбрана фигура

figure\_now = 0;

}

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

{

drawsquare.Checked = false;

drawellipse.Checked = false;

figure\_now = 2;

if (drawline.Checked == false) // Если не выбрана фигура

figure\_now = 0;

}

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

{

drawline.Checked = false;

drawellipse.Checked = false;

figure\_now = 3;

if (drawsquare.Checked == false) // Если не выбрана фигура

figure\_now = 0;

}

private void laba7\_KeyDown(object sender, KeyEventArgs e)

{

if(e.KeyCode == Keys.Delete)

{ // Удаление выделенных фигур, если нажата кнопка delete

remove\_selected\_circle(ref storag);

}

if(e.KeyCode == Keys.W)

{ // Перемещение по оси X вверх

move\_y(ref storag, -10);

}

if (e.KeyCode == Keys.S)

{ // Перемещение по оси X вниз

move\_y(ref storag, +10);

}

if (e.KeyCode == Keys.A)

{ // Перемещение по оси Y вниз

move\_x(ref storag, -10);

}

if (e.KeyCode == Keys.D)

{ // Перемещение по оси Y вверх

move\_x(ref storag, +10);

}

if (e.KeyCode == Keys.Oemplus)

{ // Увеличиваем размер фигуры

changesize(ref storag, 10);

}

if (e.KeyCode == Keys.OemMinus)

{ // Уменьшаем размер фигуры

changesize(ref storag, -10);

}

panel\_drawing.Refresh();

paint\_all(ref storag);

}

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

{ // Смена цвета у фигур

if (colorDialog1.ShowDialog() == DialogResult.Cancel)

return;

btn\_select\_color.BackColor = colorDialog1.Color;

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

{

if (!storag.check\_empty(i))

if (storag.objects[i].color == Color.Red)

{

storag.objects[i].setcolor(colorDialog1.Color);

paint\_figure(storag.objects[i].color, 4, ref storag, i);

}

}

}

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

{ // Создаём группу из выделенных фигур

Figure group = new Group();

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

{

if (!storag.check\_empty(i))

if (storag.objects[i].color == Color.Red)

{

group.GroupAddFigure(ref storag.objects[i]);

storag.delete\_object(i);

}

}

storag.add\_object(index, ref group, k, ref indexin);

}

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

{ // Разгруппировка группы

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

{

if (!storag.check\_empty(i))

if (storag.objects[i].color == Color.Red)

{

storag.objects[i].UnGroup(ref storag, i);

return;

}

}

}

string path = @"D:\Projects\7\_laba\_OOP\7\_laba\_OOP\File.txt";

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

{ // Сохраяем хранилище в файл

using (StreamWriter sw = new StreamWriter(path, false, System.Text.Encoding.Default))

{

sw.WriteLine(storag.occupied(k));

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

{

if (!storag.check\_empty(i))

{

sw.WriteLine(storag.objects[i].save());

}

}

}

}

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

{ // Загружаем данные из файла

StreamReader sr = new StreamReader(path, System.Text.Encoding.Default);

{

string str = sr.ReadLine();

int strend = Convert.ToInt32(str);

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

{

Figure figure = new Figure();

CreateFigure create = new CreateFigure();

create.caseswitch(ref sr, ref figure, create);

if (index == k)

storag.doubleSize(ref k);

storag.add\_object(index, ref figure, k, ref indexin);

++index;

}

paint\_all(ref storag);

sr.Close();

}

}

}

}

Приложение 2. Скриншот коммитов

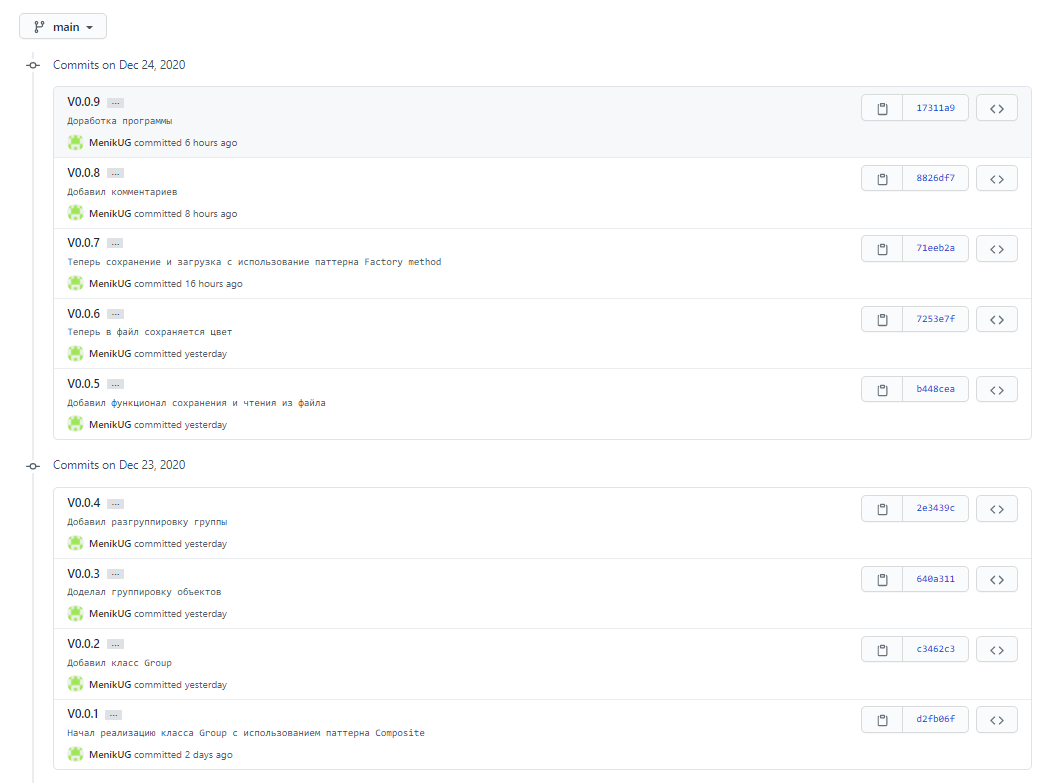


Рис. 6 Скриншот коммитов

Приложение 3. Ссылка на GitHub

<https://github.com/MenikUG/7_laba_OOP>