|  |  |
| --- | --- |
| Gerb-BMSTU_01 | **Министерство науки и высшего образования Российской Федерации**  **Федеральное государственное бюджетное образовательное учреждение**  **высшего образования**  **«Московский государственный технический университет**  **имени Н.Э. Баумана**  **(национальный исследовательский университет)»**  **(МГТУ им. Н.Э. Баумана)** |

ФАКУЛЬТЕТ **Информатика и системы управления**

КАФЕДРА **Компьютерные системы и сети (ИУ6)**

НАПРАВЛЕНИЕ ПОДГОТОВКИ **09.03.01 Информатика и вычислительная техника**

**Отчет**

|  |  |
| --- | --- |
| **По домашней работе №** | 1 |

**Название:**

Программирование на Object Pascal с использованием классов

**Дисциплина:** Объектно-ориентированное программирование

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Студент | ИУ6-22Б |  |  | И.А. Люляев |
|  | (Группа) |  | (Подпись, дата) | (И.О. Фамилия) |
|  |  |  |  |  |
| Преподаватель |  |  |  |  |
|  |  |  | (Подпись, дата) | (И.О. Фамилия) |

Москва, 2020

***Часть 1.1. Графический редактор***

*Разработать иерархию классов. Поместить определение классов в отдельном модуле.*

*Класс, позволяющий рисовать ломаную линию. Положение очередной точки определяется нажатием левой клавиши мыши. Нажатие правой клавиши заканчивает рисование текущей линии и позволяет начать новую линию.*

*Класс, позволяющий рисовать такую же ломаную линию, каждый отрезок которой оканчивается стрелкой.*

*Тип линии и толщину выбирать с использованием интерфейсных элементов.*

*В отчете показать иерархии используемых классов VCL и разработанных классов, граф состояния пользовательского интерфейса и объектную декомпозицию.*

**Код программы:**

**Модуль 1:**

unit Unit3;

interface

uses

Windows, Messages, SysUtils, Variants, Classes, Graphics, Controls, Forms,

Dialogs, ExtCtrls, Unit4, StdCtrls, Math;

type

TForm3 = class(TForm)

Image1: TImage;

Button1: TButton;

Edit1: TEdit;

Label1: TLabel;

Label2: TLabel;

Edit2: TEdit;

procedure Button1Click(Sender: TObject);

procedure Image1MouseDown(Sender: TObject; Button: TMouseButton;

Shift: TShiftState; X, Y: Integer);

private

{ Private declarations }

public

{ Public declarations }

end;

var

Form3: TForm3;

x1,y1,x2,y2:integer; //координаты начальной и конечной точек текущей линии

implementation

var

line1: Line;

line2: Line\_arrow;

{$R \*.dfm}

procedure TForm3.Button1Click(Sender: TObject);

begin

Close;

end;

procedure TForm3.Image1MouseDown(Sender: TObject; Button: TMouseButton;

Shift: TShiftState; X, Y: Integer);

var r, l: word;

begin

Image1.Canvas.Brush.Color := clWhite;

r := strtoint(edit1.Text); // получение толщины линии

l := strtoint(edit2.Text); // получение типа линии (1 или 2)

if l = 1 then begin

if button = mbLeft then begin

line1 := line.init(x, y, r, Image1);

line1.draw;

end;

if button = mbRight then begin

Image1.Canvas.MoveTo(x,y);

end;

end;

if l = 2 then begin

if button = mbLeft then begin

x1 := x2; y1 := y2;

x2 := x; y2 := y;

line2 := line\_arrow.init(x, y, r, Image1);

line2.draw(x1, y1);

end;

if button = mbRight then begin

Image1.Canvas.MoveTo(x, y);

x1 := x; y1 := y;

x2 := x; y2 := y;

end;

end;

end;

end.

**Модуль 2:**

unit Unit4;

interface

uses

Windows, Messages, SysUtils, Variants, Classes, Graphics, Controls, Forms,

Dialogs, ExtCtrls, Math;

type Line = class

x: integer;

y: integer;

width: integer;

connect: TImage;

constructor init(xd, yd, widthd: integer; AImage: TImage);

procedure draw;

end;

Line\_arrow = class(Line)

procedure DrawArrowHead(Canvas: TCanvas; X,Y: Integer; Angle,LW: Extended);

procedure draw(x0, y0:integer);

end;

implementation

constructor Line.init(xd, yd, widthd: Integer; AImage: TImage);

begin

inherited Create;

x := xd;

y := yd;

width := widthd;

connect := AImage;

end;

procedure Line.draw;

begin

self.connect.Canvas.Pen.Width := width;

self.connect.Canvas.LineTo(x, y);

end;

procedure Line\_arrow.DrawArrowHead(Canvas: TCanvas; X: Integer; Y: Integer; Angle: Extended; LW: Extended);

var

A1,A2: Extended;

Arrow: array[0..3] of TPoint;

OldWidth: Integer;

const

Beta=0.322;

LineLen=4.74;

CentLen=3;

begin

Angle:=Pi+Angle;

Arrow[0]:=Point(X,Y);

A1:=Angle-Beta;

A2:=Angle+Beta;

Arrow[1]:=Point(X+Round(LineLen\*LW\*Cos(A1)),Y-Round(LineLen\*LW\*Sin(A1)));

Arrow[2]:=Point(X+Round(CentLen\*LW\*Cos(Angle)),Y-Round(CentLen\*LW\*Sin(Angle)));

Arrow[3]:=Point(X+Round(LineLen\*LW\*Cos(A2)),Y-Round(LineLen\*LW\*Sin(A2)));

OldWidth:=Canvas.Pen.Width;

Canvas.Pen.Width:=1;

Canvas.Polygon(Arrow);

Canvas.Pen.Width:=OldWidth

end;

procedure Line\_arrow.draw(x0, y0:integer); // x0, y0 - к-ты начала линии

var

angle, LW: Extended;

x1, y1, x2, y2: integer;

begin

x1 := x0; y1 := y0;

x2 := x; y2 := y;

LW := 4;

angle:= ArcTan2(Y1 - Y2, X2 - X1);

connect.Canvas.MoveTo(X1, Y1);

connect.Canvas.LineTo(X2 - Round(2 \* LW \* cos(angle)),

Y2 + Round(2 \* LW \* sin(angle)));

DrawArrowHead(connect.Canvas, X2, Y2, angle, LW);

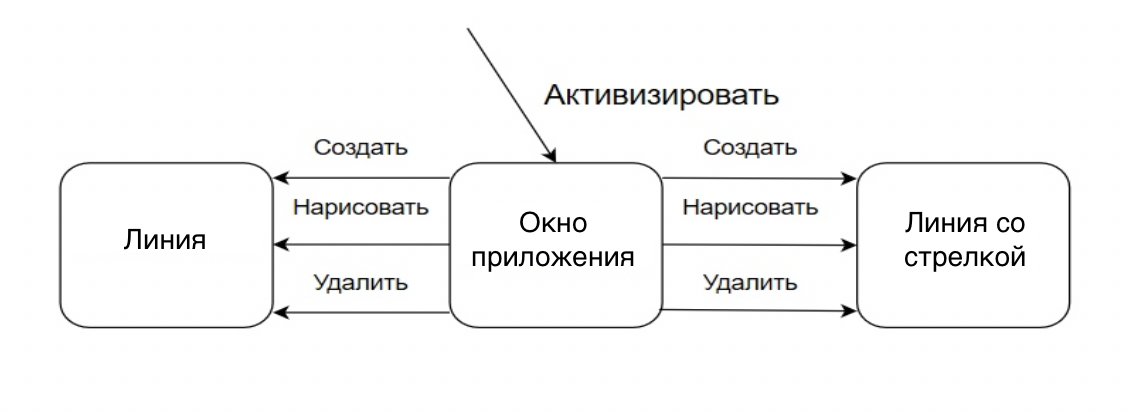
end;

end.

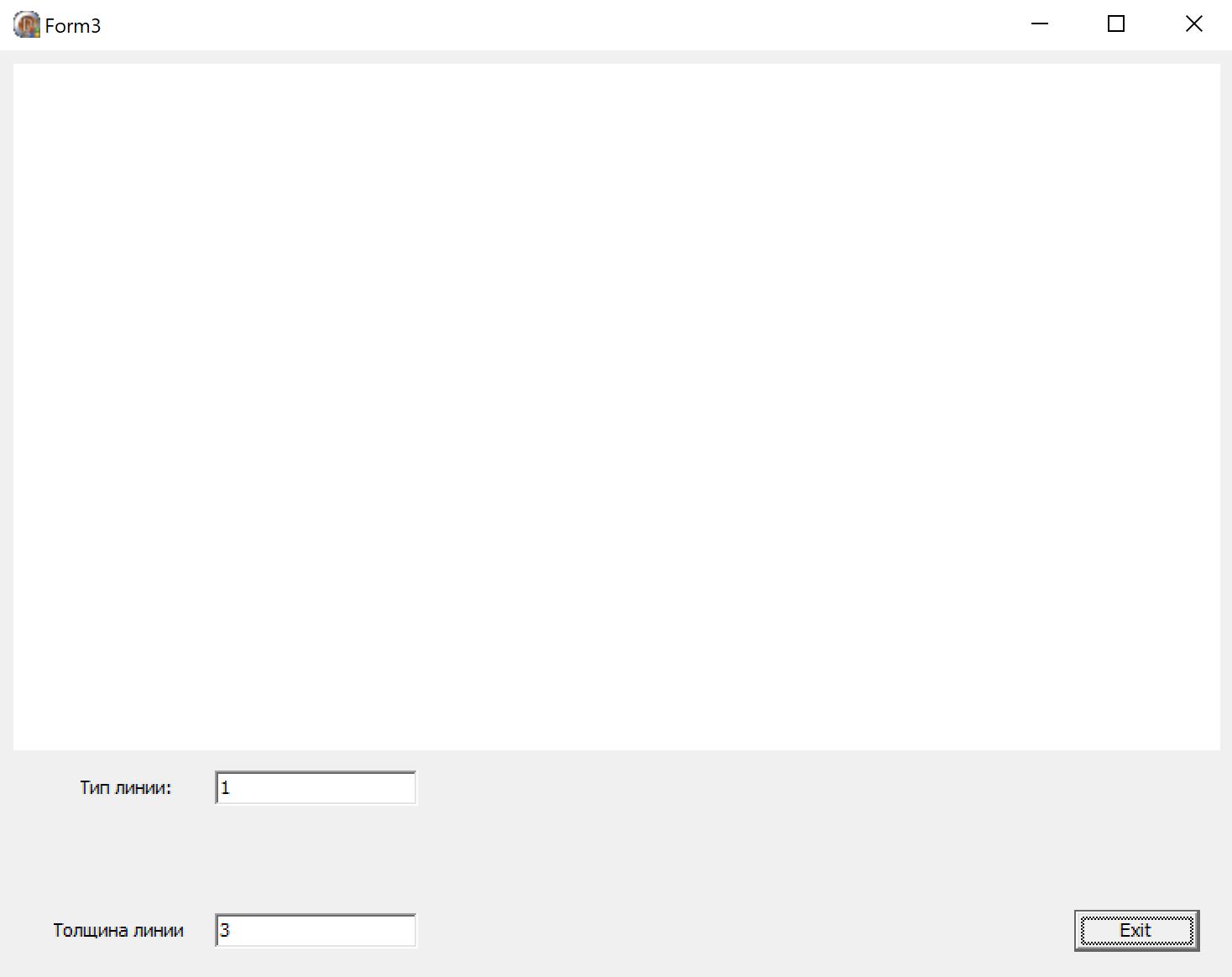
**Project Source code:**

program Project5;  
  
uses  
Forms,  
Unit3 in 'Unit3.pas' {Form1},  
Unit4 in 'Unit4.pas';  
  
{$R \*.res}  
  
begin  
Application.Initialize;  
Application.CreateForm(TForm3, Form3);  
[Application.Run](https://vk.com/away.php?to=http%3A%2F%2FApplication.Run&cc_key=);  
end.

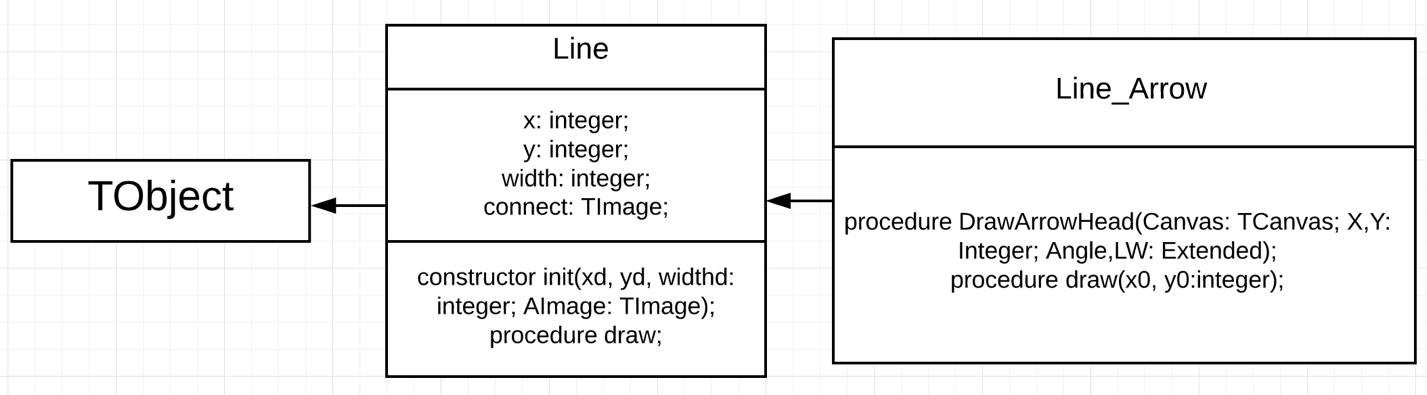
**Диаграмма объектов:**

****

**Внешний вид формы:**

****

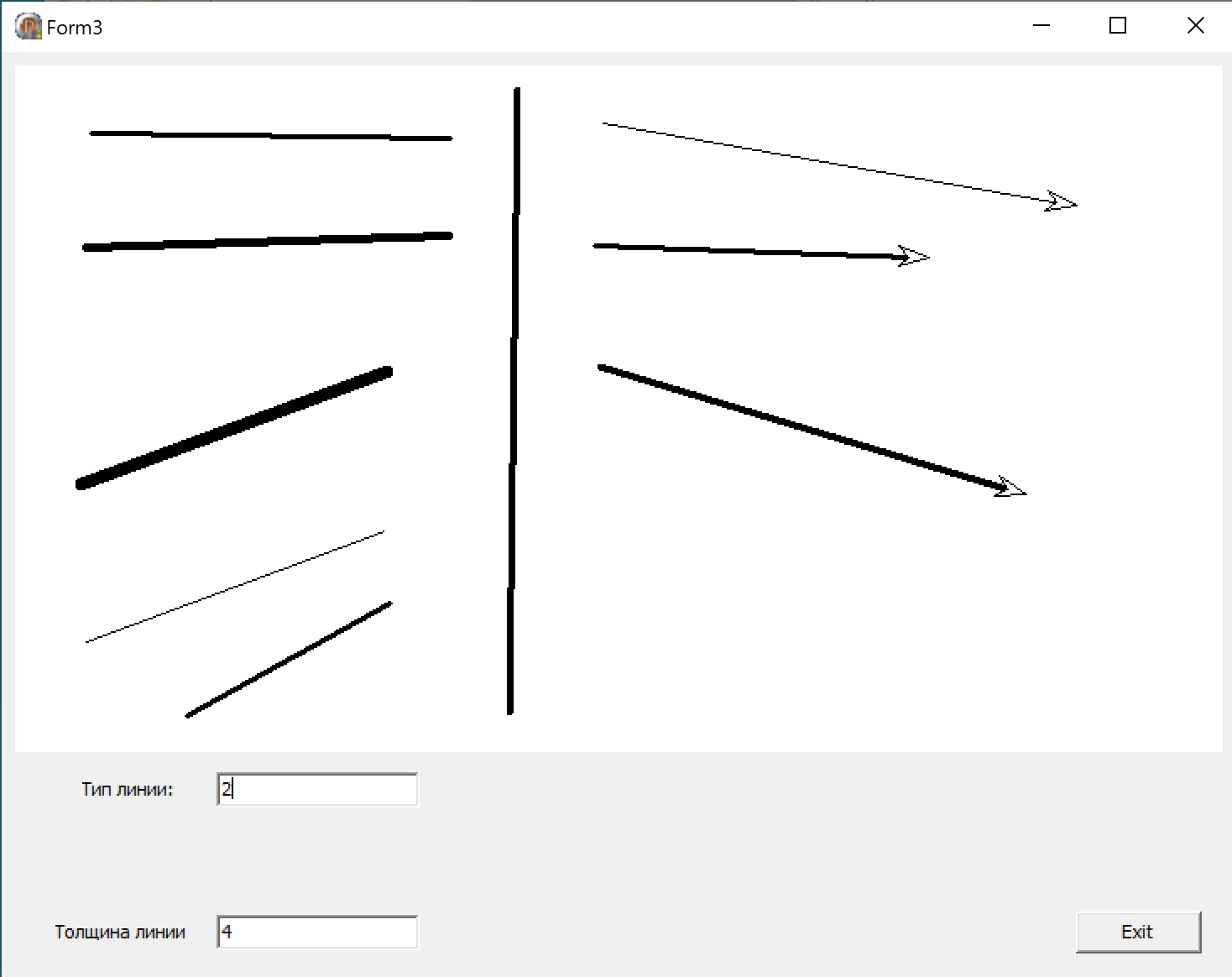
**Диаграмма классов:**

****

**Граф состояний интерфейса:**

****

**Результаты тестирования программы:**

****

**Вывод:** по результатам выполнения первой части домашнего задания я создал и реализовал классы, позволяющие отрисовывать линии разных типов. А также создал понятный для пользователя интерфейс программы.

**Часть 1.2. Полиморфное наследование**

Разработать программу, содержащую описание трех графических объектов:

изображения букв: С, Ф, О.

Реализуя механизм полиморфизма, выполнять многократное масштабирование объектов в автоматическом режиме с различными коэффициентами.

В отчете привести диаграмму используемых классов VCL и разработанных классов, граф состояний пользовательского интерфейса и объектную декомпозицию.

**Код программы:**

**Модуль 1:**

unit Unit1;

interface

uses

Windows, Messages, SysUtils, Variants, Classes, Graphics, Controls, Forms,

Dialogs, StdCtrls, ExtCtrls, unit2;

type

TForm1 = class(TForm)

Image1: TImage;

Edit1: TEdit;

Edit2: TEdit;

Button1: TButton;

Timer1: TTimer;

Label1: TLabel;

Label2: TLabel;

procedure Image1MouseDown(Sender: TObject; Button: TMouseButton;

Shift: TShiftState; X, Y: Integer);

procedure Button1Click(Sender: TObject);

procedure FormActivate(Sender: TObject);

procedure Timer1Timer(Sender: TObject);

private

{ Private declarations }

public

{ Public declarations }

end;

var

Form1: TForm1;

t : integer;

implementation

var Tc: TS\_C;

Tf: TS\_F;

To1: TS\_O;

{$R \*.dfm}

procedure TForm1.Button1Click(Sender: TObject);

begin

Close;

end;

procedure TForm1.FormActivate(Sender: TObject);

begin

image1.Canvas.brush.Color := clwhite;

end;

procedure TForm1.Image1MouseDown(Sender: TObject; Button: TMouseButton;

Shift: TShiftState; X, Y: Integer);

var r: integer;

p: string; // ·ÛÍ‚‡

begin

t:=0;

r := strtoint(Edit2.Text); // poluchenye koeficienta

p := Edit1.Text;

if p = '—' then Tc := TS\_C.Init(x, y, r, image1);

if p = '‘' then Tf := TS\_F.Init(x, y, r, image1);

if p = 'Œ' then To1 := TS\_O.Init(x, y, r, image1);

Timer1.Enabled:=true;

end;

procedure TForm1.Timer1Timer(Sender: TObject);

var p: string;

begin

p := Edit1.Text;

if p = '—' then begin

if Tc.k > 33 then Timer1.Enabled := false;

t:=t+Timer1.Interval;

Tc.Move;

end;

if p = '‘' then begin

if Tf.k > 33 then Timer1.Enabled := false;

t:=t+Timer1.Interval;

Tf.Move;

end;

if p = 'Œ' then begin

if To1.k > 33 then Timer1.Enabled := false;

t:=t+Timer1.Interval;

To1.Move;

end;

end;

end.

**Модуль 2:**

unit Unit2;

interface

uses

Windows, Messages, SysUtils, Variants, Classes, Graphics, Controls, Forms,

Dialogs, ExtCtrls;

Type TSymbol = class

text: string;

Image: TImage;

k:integer;

r: integer;

x, y: integer;

constructor Init(x0, y0:integer; r0:integer; AImage: TImage); virtual;

procedure Draw;

procedure Move;

end;

TS\_C = class(TSymbol)

constructor Init(x0, y0:integer; r0:integer; AImage: TImage); override;

end;

TS\_F = class(TSymbol)

constructor Init(x0, y0:integer; r0:integer; AImage: TImage); override;

end;

TS\_O = class(TSymbol)

constructor Init(x0, y0:integer; r0:integer; AImage: TImage); override;

end;

implementation

constructor TSymbol.Init(x0, y0:integer; r0:integer; AImage: TImage);

begin

Image := AImage;

x := x0; y := y0; k := 2; r := r0;

end;

constructor TS\_C.Init(x0, y0:integer; r0:integer; AImage: TImage);

begin

inherited Init(x0, y0,r0,AImage);

text := '—';

end;

constructor TS\_F.Init(x0, y0:integer; r0:integer; AImage: TImage);

begin

inherited Init(x0, y0,r0,AImage);

text := '‘';

end;

constructor TS\_O.Init(x0, y0:integer; r0:integer; AImage: TImage);

begin

inherited Init(x0, y0,r0,AImage);

text := 'Œ';

end;

procedure TSymbol.Draw;

begin

Image.Canvas.Font.Size := k;

Image.Canvas.TextOut(x, y, text);

end;

procedure TSymbol.Move;

begin

k := k\*r;

Draw;

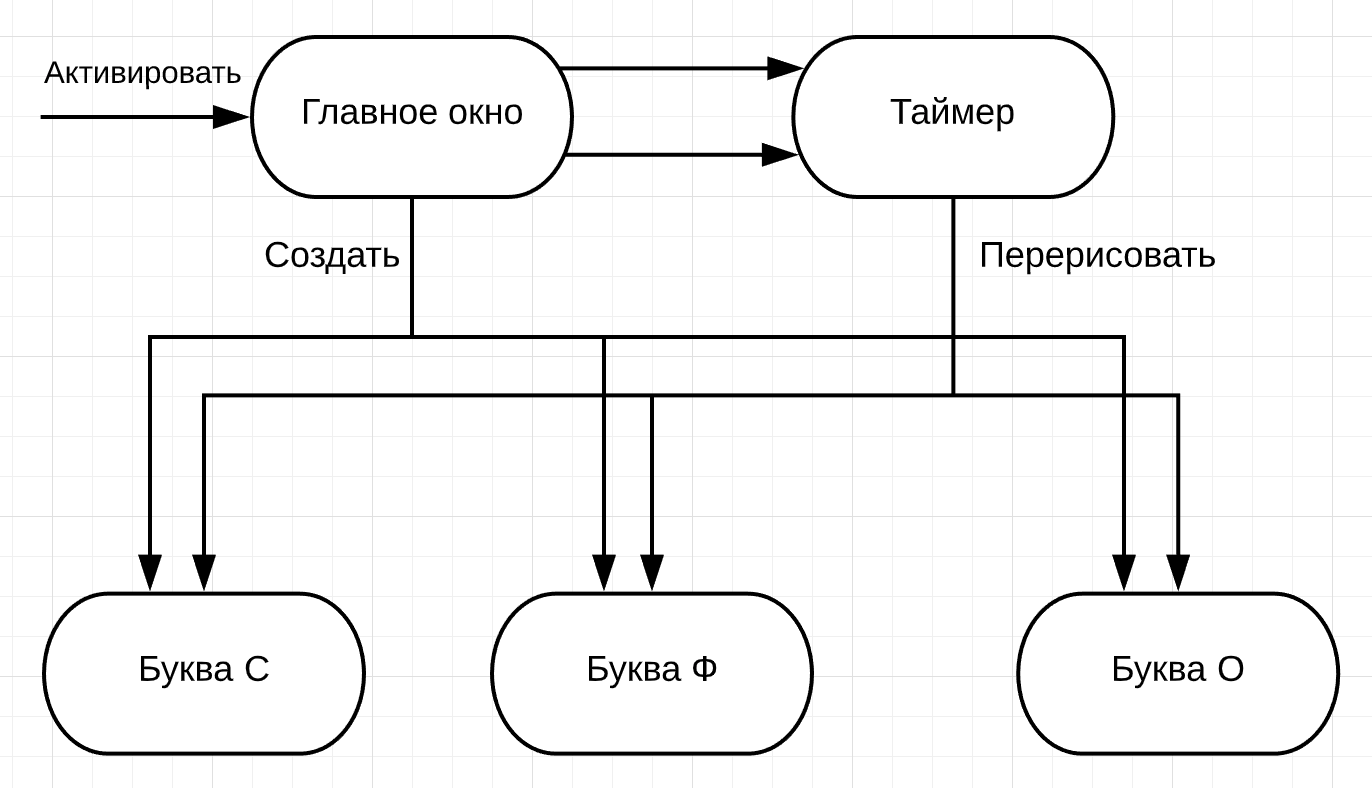
end;

end.

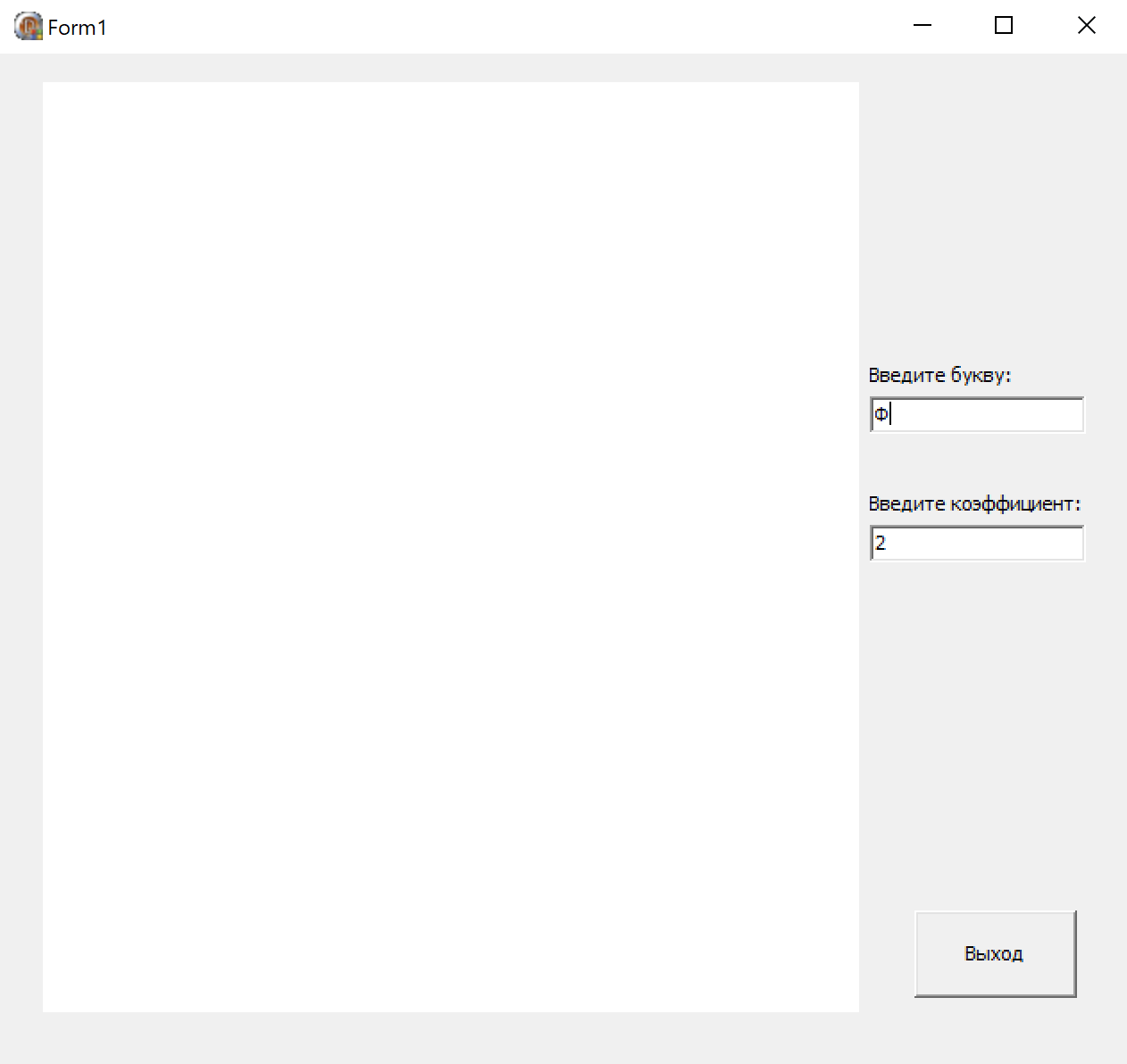
**Project Source code:**

program Project1;  
uses  
Forms,  
Unit1 in 'Unit1.pas' {Form1},  
Unit2 in 'Unit2.pas';  
{$R \*.res}  
begin  
Application.Initialize;  
Application.CreateForm(TForm1, Form1);  
[Application.Run](https://vk.com/away.php?to=http%3A%2F%2FApplication.Run&cc_key=);  
end.

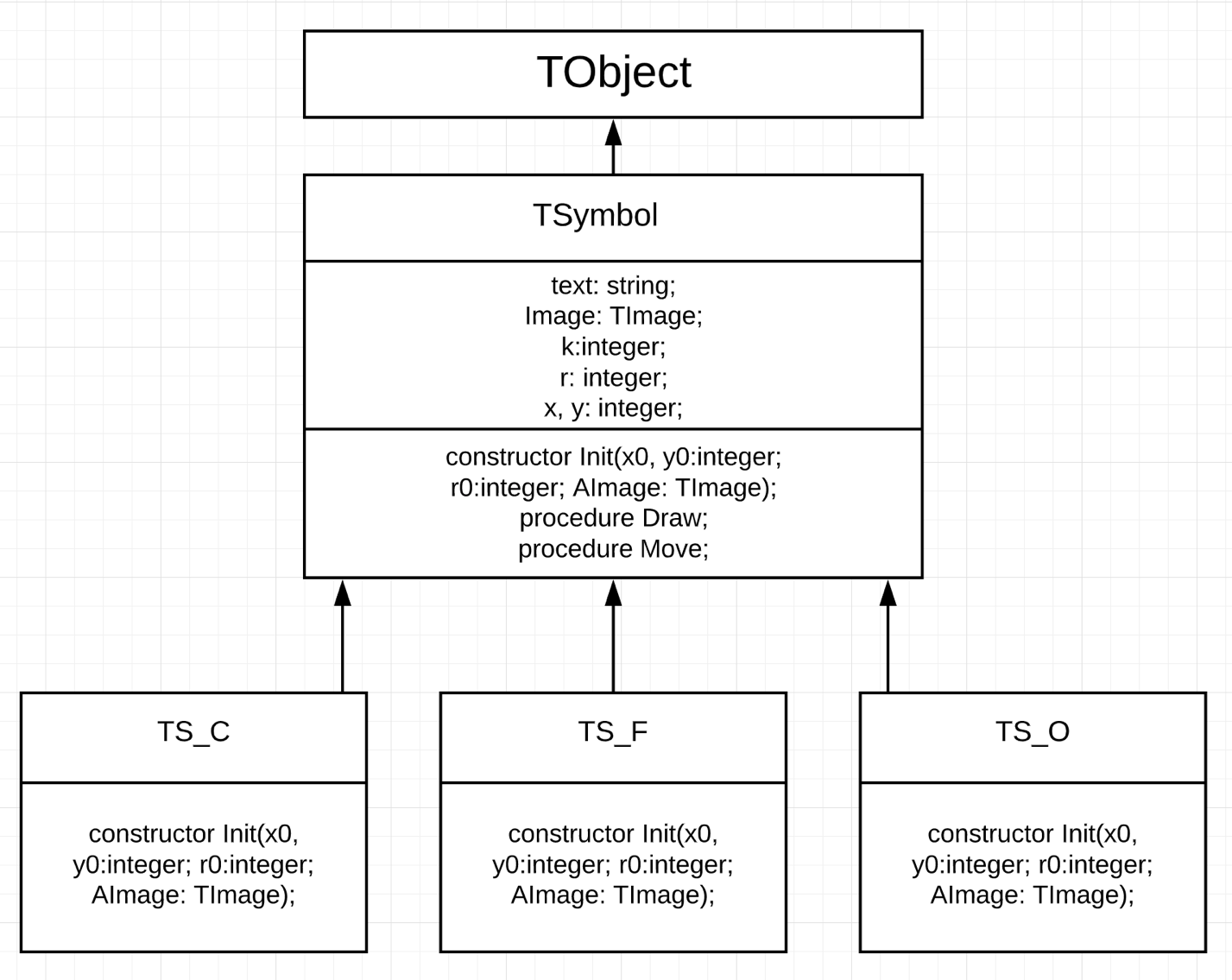
**Диаграмма объектов:**

****

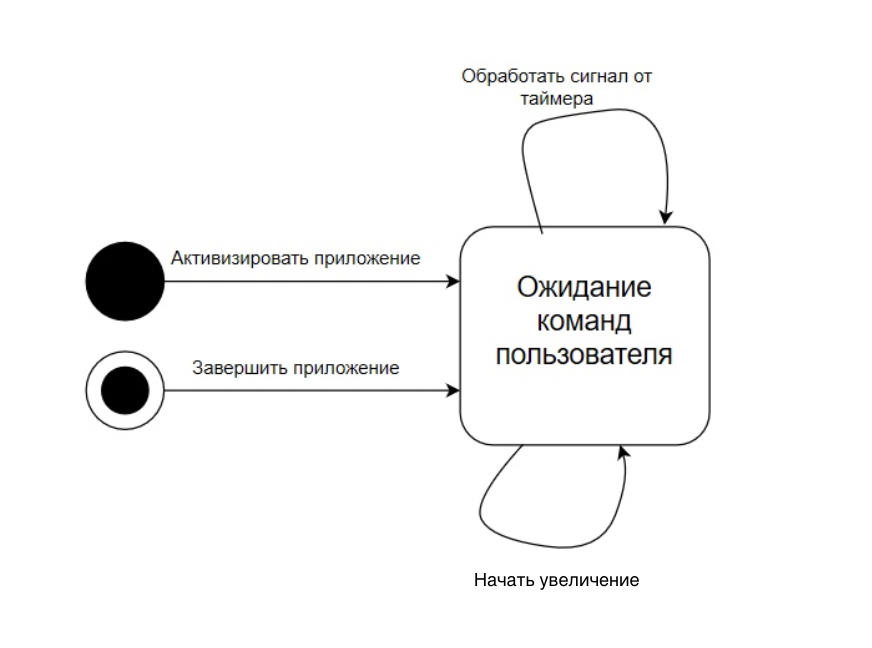
**Внешний вид формы:**

****

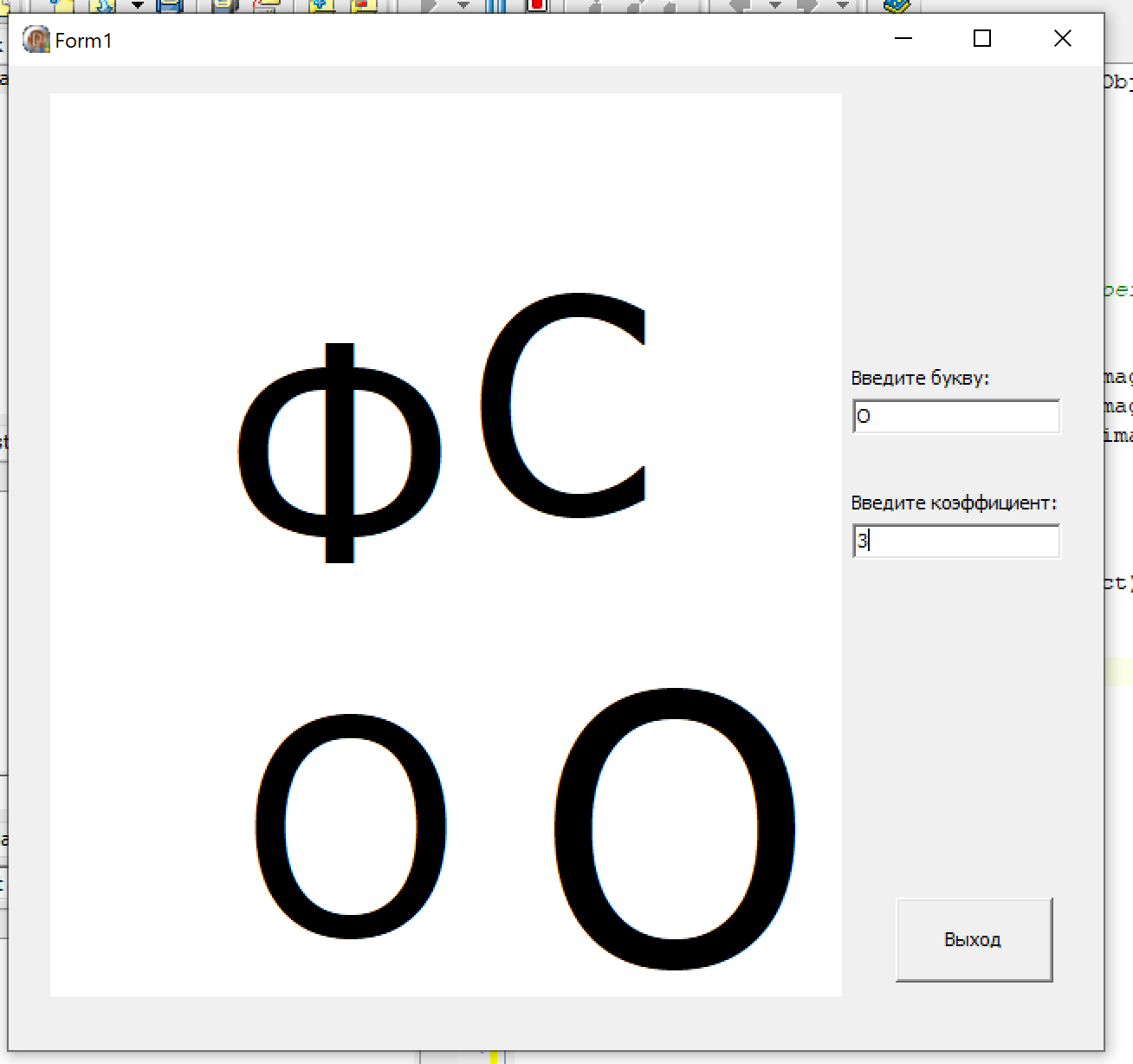
**Диаграмма классов:**

****

**Граф состояний интерфейса:**

****

**Результаты тестирования программы:**

****

**Вывод:** По результату выполнения второй части домашнего задания, я реализовал классы, с использованием полиморфного наследования – классы-потомки переопределяли конструктор базового класса. При заданных исходных параметрах программа успешно справляется со своей работой.