МИНИСТЕРСТВО ОБРАЗОВАНИЯ И НАУКИ РОССИЙСКОЙ ФЕДЕРАЦИИ ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ АВТОНОМНОЕ ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ПРОФЕССИОНАЛЬНОГО ОБРАЗОВАНИЯ

«ЮЖНЫЙ ФЕДЕРАЛЬНЫЙ УНИВЕРСИТЕТ»

ИНЖЕНЕРНО-ТЕХНОЛОГИЧЕСКАЯ АКАДЕМИЯ

Институт компьютерных технологий и информационной безопасности

Кафедра математического обеспечения и применения ЭВМ

**Лабораторная работа № 6**

по курсу «Объектно-ориентированное программирование»

**«Создание графического интерфейса»**

Выполнили:

студенты гр. КТбо2-7

Миронченко П.Д.

Проверил:

Тарасов С. А.

**Таганрог 2020**

1. **Спецификации классов  
   Спецификация класса Program** Класс является точкой входа в программу и имеет лишь один статический метод Main

**Спецификация класса GameBuilder**

Класс, задачей которого является построение объекта класса Game. Содержит приватное поле типа Game и методы, манипулирующие с ним

**Спецификация класса Game**

Класс, наследник Form. Является главным компонентом программы, содержит в себе все классы системы, кроме Program и Builder, а также методы манипуляции с ними и считывания значений с клавиатуры

**Спецификация класса Snake**

Класс, который содержит в себе поле типа List <PictureBox> и является программной моделью “Змейки”. Имеет методы передвижение, увеличения «хвоста» и методы пересечения с границей поля

**Спецификация класса Field**

Содержит в себе поле типа List <PictureBox> для отрисовки поля игры

**Спецификация класса Fruit** Является наследником класса PictureBox, при создании объекта генерируется случайная позиция на карте

**2. Используемые математические зависимости и алгоритмы**

В программе не использовались какие-либо математические зависимости и алгоритмы общего назначения

1. **Диаграмма класса**

Graphical user interface, application

Description automatically generated

1. **Листинг программы**

**Program.cs**using System;

using System.Collections.Generic;

using System.Linq;

using System.Threading.Tasks;

using System.Windows.Forms;

namespace lab6

{

static class Program

{

*/// <summary>*

*/// Главная точка входа для приложения.*

*/// </summary>*

[STAThread]

static void Main()

{

Application.EnableVisualStyles();

Application.SetCompatibleTextRenderingDefault(false);

GameBuilder builder = new GameBuilder();

builder.SetWigthAndHeight();

builder.SetLabelHeight();

builder.SetSizeOfSizes();

builder.SetWindowParameters();

builder.SetDirection();

builder.SetSnake();

builder.SetFruit();

builder.SetScore();

builder.SetScoreLabel();

builder.SetField();

builder.AddItemsToControls();

Application.Run(builder.Game);

}

}

}

**Game.cs**

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;

namespace lab6

{

public partial class Game : Form

{

private int \_widthAndHeight;

public int WidthAndHeight

{

get { return \_widthAndHeight; }

set { \_widthAndHeight = value; }

}

private int \_labelHeight;

public int LabelHeight

{

get { return \_labelHeight; }

set { \_labelHeight = value; }

}

private int \_sizeOfSides;

public int SizeOfSides

{

get { return \_sizeOfSides; }

set { \_sizeOfSides = value; }

}

private Field \_field;

public Field Field

{

get { return \_field; }

set { \_field = value; }

}

private Fruit \_fruit;

public Fruit Fruit

{

get { return \_fruit; }

set { \_fruit = value; }

}

private Direction \_direction;

public Direction Direction

{

get { return \_direction; }

set { \_direction = value; }

}

private Snake \_snake;

public Snake Snake

{

get { return \_snake; }

set { \_snake = value; }

}

private int \_score;

public int Score

{

get { return \_score; }

set { \_score = value; }

}

private Label \_scoreLabel;

public Label ScoreLabel

{

get { return \_scoreLabel; }

set { \_scoreLabel = value; }

}

private Timer \_timer;

public Game()

{

InitializeComponent();

\_timer = new Timer();

\_timer.Tick += new EventHandler(\_Update);

\_timer.Interval = 200;

\_timer.Start();

this.KeyDown += new KeyEventHandler(\_ChangeDirection);

}

private void \_Update(object sender, EventArgs eventsArgs)

{

try

{

\_snake.Move(\_direction);

\_EatFruit();

}

catch (GameOverException e)

{

\_timer.Stop();

MessageBox.Show(e.Message);

Application.Exit();

}

}

private void \_EatFruit()

{

if (\_snake[0].Location == \_fruit.Location)

{

this.Controls.Remove(\_fruit);

\_fruit = new Fruit(\_widthAndHeight, \_sizeOfSides);

this.Controls.Add(\_fruit);

\_scoreLabel.Text = "Score: " + ++\_score;

\_snake.AddNewTail(\_direction);

this.Controls.Add(\_snake[\_score]);

}

}

private void \_ChangeDirection(object sender, KeyEventArgs e)

{

switch (e.KeyCode.ToString())

{

case "Right":

\_direction = Direction.RIGHT;

break;

case "Left":

\_direction = Direction.LEFT;

break;

case "Up":

\_direction = Direction.UP;

break;

case "Down":

\_direction = Direction.DOWN;

break;

}

}

}

}

**GameBuilder.cs**

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

using System.Drawing;

using System.Windows.Forms;

namespace lab6

{

public class GameBuilder

{

private Game \_game;

public Game Game

{

get

{

return \_game;

}

}

public GameBuilder()

{

\_game = new Game();

}

public void SetWigthAndHeight(int widthAndHeight = 600)

{

\_game.WidthAndHeight = widthAndHeight;

}

public void SetLabelHeight(int labelHeight = 50)

{

\_game.LabelHeight = labelHeight;

}

public void SetSizeOfSizes(int sizeOfSizes = 30)

{

\_game.SizeOfSides = sizeOfSizes;

}

public void SetWindowParameters()

{

\_game.Text = "Snake";

\_game.Width = \_game.WidthAndHeight + \_game.WidthAndHeight / \_game.SizeOfSides;

\_game.Height = \_game.Width + \_game.LabelHeight;

}

public void SetField()

{

\_game.Field = new Field(\_game.WidthAndHeight, \_game.SizeOfSides);

}

public void SetDirection()

{

\_game.Direction = Direction.LEFT;

}

public void SetSnake()

{

\_game.Snake = new Snake(\_game.WidthAndHeight, \_game.SizeOfSides);

}

public void SetFruit()

{

\_game.Fruit = new Fruit(\_game.WidthAndHeight, \_game.SizeOfSides);

}

public void SetScore()

{

\_game.Score = 0;

}

public void SetScoreLabel()

{

\_game.ScoreLabel = new Label

{

Text = "Score: 0",

Location = new Point(\_game.WidthAndHeight / 2, \_game.WidthAndHeight + 1),

Size = new Size(\_game.WidthAndHeight, \_game.LabelHeight)

};

}

public void AddItemsToControls()

{

\_game.Controls.Add(\_game.Fruit);

\_game.Controls.Add(\_game.Snake[0]);

\_game.Controls.Add(\_game.ScoreLabel);

foreach (PictureBox p in \_game.Field.GetField)

{

\_game.Controls.Add(p);

}

}

}

}

**Field.cs**

using System;

using System.Drawing;

using System.Windows.Forms;

using System.Collections.Generic;

namespace lab6

{

public class Field

{

private int \_widthAndHeight;

private int \_sizeOfSides;

private List<PictureBox> \_field;

public List<PictureBox> GetField

{

get { return \_field; }

}

public Field(int widthAndHeight, int sizeOfSides)

{

\_widthAndHeight = widthAndHeight;

\_sizeOfSides = sizeOfSides;

\_field = new List<PictureBox>();

for (int i = 0; i < \_widthAndHeight / \_sizeOfSides + 1; i++)

{

PictureBox pic = new PictureBox();

pic.BackColor = Color.Black;

pic.Location = new Point(0, \_sizeOfSides \* i);

pic.Size = new Size(\_widthAndHeight, 1);

\_field.Add(pic);

}

for (int i = 0; i <= \_widthAndHeight / \_sizeOfSides; i++)

{

PictureBox pic = new PictureBox();

pic.BackColor = Color.Black;

pic.Location = new Point(\_sizeOfSides \* i, 0);

pic.Size = new Size(1, \_widthAndHeight);

\_field.Add(pic);

}

}

}

}

**Snake.cs**

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

using System.Drawing;

using System.Windows.Forms;

namespace lab6

{

public class Snake

{

private int \_widthAndHeight;

private int \_sizeOfSides;

private int \_size;

private List<PictureBox> \_snake;

public PictureBox this[int index]

{

get

{

return \_snake[index];

}

set

{

\_snake[index] = value;

}

}

public int Size

{

get

{

return \_size;

}

}

public Snake(int widthAndHeight, int sizeOfSides)

{

\_widthAndHeight = widthAndHeight;

\_sizeOfSides = sizeOfSides;

\_snake = new List<PictureBox>();

\_snake.Add(new PictureBox());

\_snake[0].Location = new Point(\_widthAndHeight / 2 + 1, \_widthAndHeight / 2 + 1);

\_snake[0].Size = new Size(\_sizeOfSides - 1, \_sizeOfSides - 1);

\_snake[0].BackColor = Color.Red;

\_size = \_snake.Count();

}

public void AddNewTail(Direction direction)

{

\_snake.Add(new PictureBox());

\_snake[\_size].Location = new Point(\_snake[\_size - 1].Location.X, \_snake[\_size - 1].Location.Y);

switch (direction)

{

case Direction.UP:

\_snake[\_size].Location = new Point(\_snake[\_size].Location.X, \_snake[\_size].Location.Y - \_sizeOfSides);

break;

case Direction.DOWN:

\_snake[\_size].Location = new Point(\_snake[\_size].Location.X, \_snake[\_size].Location.Y + \_sizeOfSides);

break;

case Direction.LEFT:

\_snake[\_size].Location = new Point(\_snake[\_size].Location.X - \_sizeOfSides, \_snake[\_size].Location.Y);

break;

case Direction.RIGHT:

\_snake[\_size].Location = new Point(\_snake[\_size].Location.X + \_sizeOfSides, \_snake[\_size].Location.Y);

break;

}

\_snake[\_size].Size = new Size(\_sizeOfSides - 1, \_sizeOfSides - 1);

\_snake[\_size].BackColor = Color.Red;

\_size++;

}

private void \_CheckBorders()

{

if (\_snake[0].Location.X < 1)

{

\_snake[0].Location = new Point(\_widthAndHeight - \_sizeOfSides + 1, \_snake[0].Location.Y);

}

if (\_snake[0].Location.Y < 1)

{

\_snake[0].Location = new Point(\_snake[0].Location.X, \_widthAndHeight - \_sizeOfSides + 1);

}

if (\_snake[0].Location.Y > \_widthAndHeight)

{

\_snake[0].Location = new Point(\_snake[0].Location.X, 1);

}

if (\_snake[0].Location.X > \_widthAndHeight)

{

\_snake[0].Location = new Point(1, \_snake[0].Location.Y);

}

}

private void \_CheckTailDamage(Direction direction)

{

if (\_snake.Count > 2 && (\_snake[0].Location == \_snake[2].Location))

{

\_snake.Reverse();

}

for (int i = 3; i < \_size; ++i)

{

if (\_snake[i].Location == \_snake[0].Location)

{

throw new GameOverException("Game over!");

}

}

}

public void Move(Direction direction)

{

for (int i = \_size - 1; i > 0; --i)

{

\_snake[i].Location = \_snake[i - 1].Location;

}

switch (direction)

{

case Direction.UP:

\_snake[0].Location = new Point(\_snake[0].Location.X, \_snake[0].Location.Y - \_sizeOfSides);

break;

case Direction.DOWN:

\_snake[0].Location = new Point(\_snake[0].Location.X, \_snake[0].Location.Y + \_sizeOfSides);

break;

case Direction.LEFT:

\_snake[0].Location = new Point(\_snake[0].Location.X - \_sizeOfSides, \_snake[0].Location.Y);

break;

case Direction.RIGHT:

\_snake[0].Location = new Point(\_snake[0].Location.X + \_sizeOfSides, \_snake[0].Location.Y);

break;

}

\_CheckBorders();

\_CheckTailDamage(direction);

}

}

}

**Direction.cs**

using System;

namespace lab5

{

public enum Direction

{

LEFT,

RIGHT,

UP,

DOWN

}

}

**Fruit.cs**

¸ using System;

using System.Drawing;

using System.Windows.Forms;

namespace lab6

{

public class Fruit : PictureBox

{

public Fruit(int widthAndHeight, int sizeOfSides)

{

int rI, rJ;

this.BackColor = Color.Yellow;

Random r = new Random();

rI = r.Next(0, widthAndHeight - sizeOfSides);

int tempI = rI % sizeOfSides;

rI -= tempI;

rJ = r.Next(0, widthAndHeight - sizeOfSides);

int tempJ = rJ % sizeOfSides;

rJ -= tempJ;

rI++;

rJ++;

this.Location = new Point(rI, rJ);

this.Size = new Size(sizeOfSides - 1, sizeOfSides - 1);

}

}

}

**GameOverException.cs**

using System;

namespace lab5

{

public class GameOverException : Exception

{

public GameOverException()

{

}

public GameOverException(string message) : base(message)

{

}

public GameOverException(string message, Exception inner)

: base(message, inner)

{

}

}

}