**МИНИCTEPCTBO НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ**

**РОССИЙСКОЙ ФЕДЕРАЦИИ**

**Федеральное государственное автономное**

**образовательное учреждение высшего образования**

**«Северо-Кавказский федеральный университет»**

**Институт Перспективной Инженерии**

**Департамент цифровых, робототехнических систем и электроники**

**Отчет по лабораторной работе № 5**

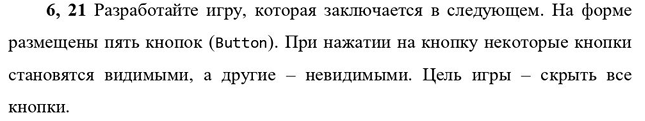
**«Объектное моделирование»**

**по дисциплине «Технологии разработки программного обеспечения»**

|  |
| --- |
| Выполнил студент группы ПИЖ-б-о-23-2 |
| Мальцев В.И. «   »\_\_\_\_\_\_\_\_\_20\_\_г. |
| Подпись студента\_\_\_\_\_\_\_\_\_\_\_ |
| Работа защищена «   »\_\_\_\_\_\_\_\_\_\_\_\_20\_\_г. |
| Проверила Хацукова А.И. \_\_\_\_\_\_\_\_  (подпись) |

Ставрополь 2025

**Задание 1.** Вариант 6



Код проекта:

using System;

using System.Collections.Generic;

using System.Windows.Forms;

namespace WF1

{

public partial class Form1 : Form

{

private Button[] buttons;

private Dictionary<Button, List<Button>> buttonActions; // Какие кнопки влияют какие

private Random random = new Random();

public Form1()

{

InitializeComponent();

buttons = new Button[] { button1, button2, button3, button4, button5 };

buttonActions = new Dictionary<Button, List<Button>>();

RestartGame(); // Запуск новой игры

foreach (var btn in buttons)

{

btn.Click += Button\_Click;

}

}

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

{

var clickedButton = (Button)sender;

// Переключаем состояние всех кнопок из списка действий

foreach (var targetButton in buttonActions[clickedButton])

{

targetButton.Visible = !targetButton.Visible;

}

// Проверяем победу

if (!AnyButtonVisible())

{

MessageBox.Show("Поздравляем! Вы выиграли!", "Выигрыш", MessageBoxButtons.OK, MessageBoxIcon.Information);

RestartGame();

}

}

private bool AnyButtonVisible()

{

foreach (var btn in buttons)

{

if (btn.Visible) return true;

}

return false;

}

private void RestartGame()

{

// Восстанавливаем все кнопки

foreach (var btn in buttons)

{

btn.Visible = true;

}

// Генерируем новые случайные действия для каждой кнопки

GenerateRandomButtonActions();

}

private void GenerateRandomButtonActions()

{

bool isValid = false;

while (!isValid)

{

buttonActions.Clear();

for (int i = 0; i < buttons.Length; i++)

{

var currentButton = buttons[i];

var allButtons = new List<Button>(buttons); // Теперь включая и саму кнопку

int minCount = 1; // Каждая кнопка должна влиять хотя бы на одну

int maxCount = allButtons.Count;

int actionCount = random.Next(minCount, maxCount + 1);

var affectedButtons = new List<Button>();

while (affectedButtons.Count < actionCount)

{

int index = random.Next(allButtons.Count);

affectedButtons.Add(allButtons[index]);

allButtons.RemoveAt(index);

}

buttonActions[currentButton] = affectedButtons;

}

isValid = IsGraphConnected() && CanAllBeHiddenInSequence();

}

}

// Проверяем, что все кнопки могут быть изменены какой-то кнопкой

private bool CanAllBeHiddenInSequence()

{

HashSet<Button> canBeToggled = new HashSet<Button>();

foreach (var pair in buttonActions)

{

foreach (var btn in pair.Value)

{

canBeToggled.Add(btn);

}

}

return canBeToggled.Count == buttons.Length;

}

// Проверяем связность графа (можно добраться до всех узлов)

private bool IsGraphConnected()

{

HashSet<Button> visited = new HashSet<Button>();

Stack<Button> stack = new Stack<Button>();

stack.Push(buttons[0]);

visited.Add(buttons[0]);

while (stack.Count > 0)

{

var current = stack.Pop();

foreach (var neighbor in buttonActions[current])

{

if (!visited.Contains(neighbor))

{

visited.Add(neighbor);

stack.Push(neighbor);

}

}

}

return visited.Count == buttons.Length;

}

}

}

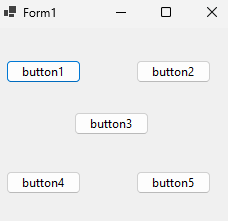


Рисунок 1 – Результат выполнения программы