МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ

РОССИЙСКОЙ ФЕДЕРАЦИИ

ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ БЮДЖЕТНОЕ

ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ

«ОРЛОВСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ

ИМЕНИ И.С. ТУРГЕНЕВА»

Кафедра программной инженерии

**ОТЧЕТ**

по лабораторной работе № 4

на тему: «Тестирование пользовательского интерфейса»

по дисциплине: «Качество и тестирование программного обеспечения»

Вариант 18

Выполнил: Шорин В.Д. Шифр: 171406

Институт приборостроения, автоматизации и информационных технологий

Направление: 09.03.04 «Программная инженерия»

Группа: 71ПГ

Проверили: Ужаринский А.Ю., Конюхова О.В.

Отметка о зачете:

Дата: «\_\_\_\_» \_\_\_\_\_\_\_\_\_\_ 2020 г.

Орел, 2020 г.

**Задание на лабораторную работу**

Разработать сценарии использования и провести тестирование пользовательского интерфейса приложения.

**Описание пользовательского интерфейса**

Приложение состоит из главного окна, которое содержит в себе:

* Поле, в котором будут выведены билеты;
* Кнопку «Сгенерировать новый список», выполняющую генерацию нового списка билетов;
* Кнопку «Выход», по нажатию на которую происходит закрытие окна приложения.

При запуске приложения пользователю представляется пустое поле и две кнопки (рисунок 1). При нажатии на кнопку «Сгенерировать новый список» пустое поле заполняется новым списком билетов (рисунок 2). При повторном нажатии на эту же кнопку в поле будет удален старый список и сгенерирован новый. Также, пользователь может в любой момент нажать на кнопку «Выход», что приведет к закрытию приложения. Нажатие мышкой на строку с билетом приведет к ее выделению синим цветом, но с ней нельзя будет что-либо сделать.

Использование кнопки «Tab» будет переключать фокус между полем, кнопкой «Сгенерировать новый список» и «Выход» и нажатие на кнопку «Enter» при фокусе на одной из кнопок приведет к тому же результату, что и нажатие мышкой, описанное выше. Нажатие на «Enter» при фокусе на поле не приведет ни к какому результату. Однако, при фокусе на поле с помощью стрелок клавиатуры «Вверх» и «Вниз» пользователю будет предоставлена возможность переключаться между билетами, но нажатие на «Enter» и в этом случае не приведет ни к какому результату.

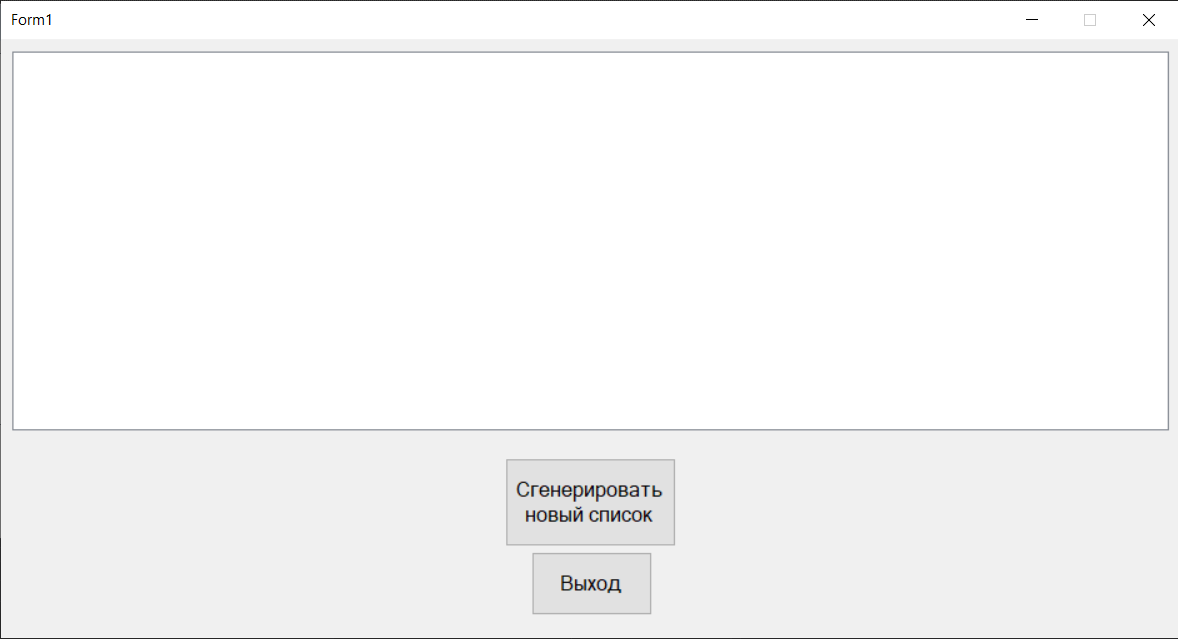


Рисунок 1 – Начальный экран приложения

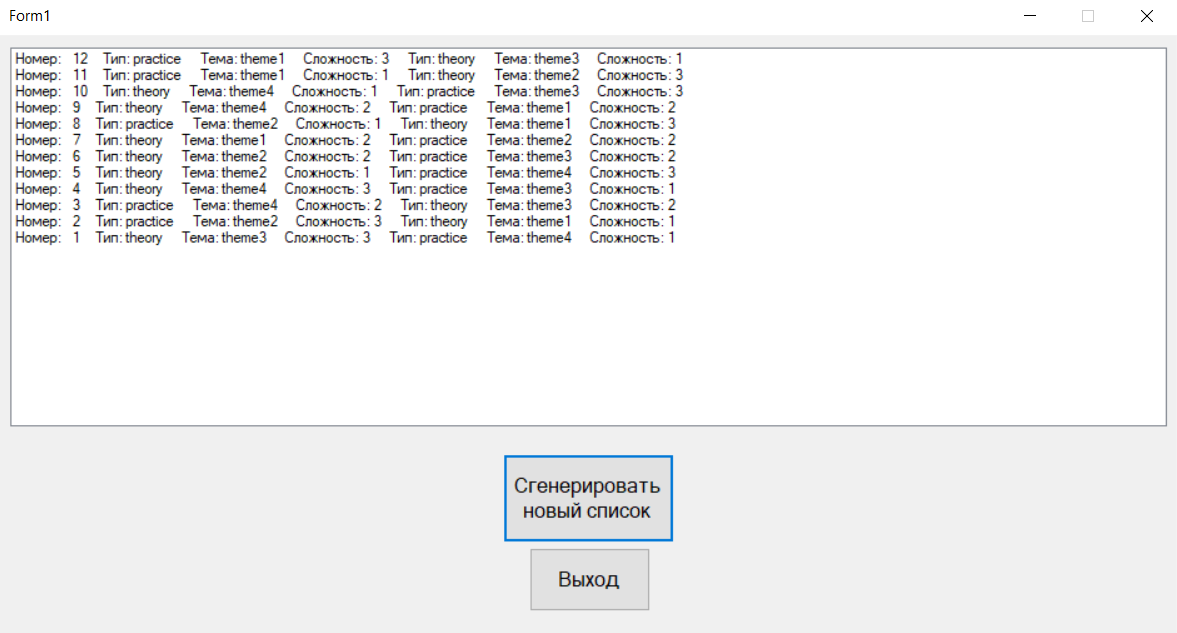


Рисунок 2 – Нажатие на кнопку «Сгенерировать новый список»

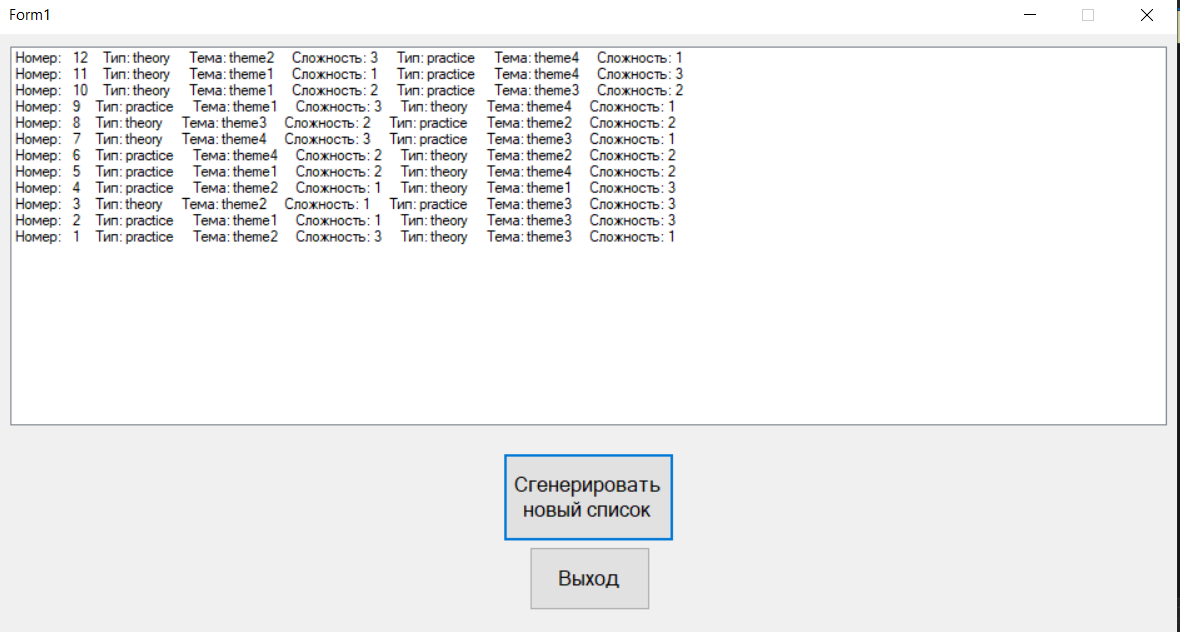


Рисунок 3 – Повторное нажатие на кнопку «Сгенерировать новый список»

**Код**

**«Task.cs»**

using System;

namespace Testing\_Lab\_4.Model

{

class Task

{

public string TaskType { get; private set; }

public string Theme { get; private set; }

public int Difficulty { get; private set; }

public Task(string type, string theme, int difficulty)

{

TaskType = type;

Theme = theme;

Difficulty = difficulty;

}

public Task() { throw new NotImplementedException(); }

public void PrintTask()

{

Console.WriteLine($"TaskType: {TaskType} " + $"Theme: {Theme} " + $"Difficulty: {Difficulty} ");

}

}

}

**«Ticket.cs»**

using System;

using System.Collections.Generic;

namespace Testing\_Lab\_4.Model

{

class Ticket

{

public int Number { get; private set; }

public List<Task> Tasks { get; private set; }

public Ticket(int number, List<Task> tasks) { Tasks = tasks; Number = number; }

public void PrintTicket()

{

Console.WriteLine($"Number: {Number}");

foreach (var task in Tasks) { task.PrintTask(); }

Console.WriteLine();

}

}

}

**«Form1.cs»**

using System;

using System.Collections.Generic;

using System.Linq;

using System.Windows.Forms;

using System.Xml;

using System.Xml.Linq;

namespace Testing\_Lab\_4

{

public partial class Form1 : Form

{

private List<Model.Task> tasks;

public Form1()

{

InitializeComponent();

string path = "../../Model/tasks.xml";

//CreateTasks(path);

tasks = GetTasksFromXML(path);

}

private void button1\_Click(object sender, EventArgs e)

{

listBoxTickets.Items.Clear();

GenerateTickets();

}

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

{

Close();

}

private void GenerateTickets()

{

List<Model.Task> tasksLocal = tasks.ToList();

List<Model.Ticket> tickets = new List<Model.Ticket>();

int ticketNumber = 1;

var random = new Random();

while (true)

{

List<Model.Task> ticketTasks = new List<Model.Task>();

int randomTask = random.Next(tasksLocal.Count);

// Add first task (any, any, any)

ticketTasks.Add(tasksLocal[randomTask]);

tasksLocal.RemoveAt(randomTask);

randomTask = random.Next(tasksLocal.Count);

List<Model.Task> copyTasks = tasksLocal.ToList();

// Add 2rd task (!such in 1st task, !such in 1st task, sum == 4)

while (true)

{

if (copyTasks[randomTask].TaskType != ticketTasks[0].TaskType

&& copyTasks[randomTask].Theme != ticketTasks[0].Theme

&& copyTasks[randomTask].Difficulty + ticketTasks[0].Difficulty == 4

)

{

ticketTasks.Add(copyTasks[randomTask]);

tasksLocal.Remove(copyTasks[randomTask]);

break;

}

copyTasks.RemoveAt(randomTask);

if (copyTasks.Count == 0)

{

break;

}

randomTask = random.Next(copyTasks.Count);

}

tickets.Add(new Model.Ticket(ticketNumber, ticketTasks));

if (tasksLocal.Count == 2)

{

ticketNumber++;

ticketTasks = tasksLocal.ToList();

tickets.Add(new Model.Ticket(ticketNumber, ticketTasks));

break;

}

if (tasksLocal.Count == 0)

{

break;

}

ticketNumber++;

}

//listBoxTickets.Items.AddRange(tickets);

//Console.WriteLine("Tickets");

foreach (var ticket in tickets)

{

//ticket.PrintTicket();

string ticketNum = "Номер: " + ticket.Number + " ";

string ticketTasks = "";

foreach (var task in ticket.Tasks)

{

ticketTasks += $"Тип: {task.TaskType} " +

$"Тема: {task.Theme} " +

$"Сложность: {task.Difficulty} ";

}

//string ticketTask0 = $"Задание1: " +

// $"Тип: {ticket.Tasks[0].TaskType} " +

// $"Тема: {ticket.Tasks[0].Theme} " +

// $"Сложность: {ticket.Tasks[0].Difficulty} ";

//string ticketTask1 = $"Задание2: " +

// $"Тип: {ticket.Tasks[1].TaskType} " +

// $"Тема: {ticket.Tasks[1].Theme} " +

// $"Сложность: {ticket.Tasks[1].Difficulty} ";

string ticketFull = ticketNum + ticketTasks;

listBoxTickets.Items.Insert(0, ticketFull);

}

}

public void CreateTasks(string pathToSave)

{

string[] themes = { "theme1", "theme2", "theme3", "theme4" };

int[] difficulties = { 1, 2, 3 };

string[] taskTypes = { "theory", "practice" };

XDocument xDocument = new XDocument();

XElement tasks = new XElement("tasks");

foreach (var theme in themes)

{

foreach (var difficulty in difficulties)

{

foreach (var taskType in taskTypes)

{

XElement xeTask = new XElement("task");

XElement xeTaskType = new XElement("taskType", taskType);

XElement xeTheme = new XElement("theme", theme);

XElement xeDifficulty = new XElement("difficulty", difficulty);

xeTask.Add(xeTaskType);

xeTask.Add(xeTheme);

xeTask.Add(xeDifficulty);

tasks.Add(xeTask);

}

}

}

xDocument.Add(tasks);

xDocument.Save(pathToSave);

}

private List<Model.Task> GetTasksFromXML(string pathToLoad)

{

List<Model.Task> tasks = new List<Model.Task>();

XmlDocument xmlDocument = new XmlDocument();

xmlDocument.Load(pathToLoad);

XmlElement xRoot = xmlDocument.DocumentElement;

string taskType = "";

string theme = "";

int difficulty = 0;

foreach (XmlNode xmlNode in xRoot)

{

foreach (XmlNode childNode in xmlNode.ChildNodes)

{

taskType = (childNode.Name == "taskType") ? childNode.InnerText : taskType;

theme = (childNode.Name == "theme") ? childNode.InnerText : theme;

difficulty = (childNode.Name == "difficulty") ? Convert.ToInt32(childNode.InnerText) : difficulty;

}

tasks.Add(new Model.Task(taskType, theme, difficulty));

}

return tasks;

}

}

}