

СЪДЪРЖАНИЕ

I. Изграждане на графичен потребителски интерфейс с Windows Forms.

1. Какво е Windows Forms?
2. Контролите в Windows Forms
3. Библиотеките на .NET за изграждане на GUI

II. Програмни компоненти.

1. Компоненти и контейнери
2. Контроли и контейнер-контроли

III. Програмен модел на Windows Forms.

1. Форми
2. Контроли
3. Събития
4. Жизнен цикъл на Windows Forms приложенията

IV. Създаване на Windows Forms проект

I. ИЗГРАЖДАНЕ НА ГРАФИЧЕН ПОТРЕБИТЕЛСКИ ИНТЕРФЕЙС С WINDOWS FORMS



Какво е Windows Forms?

Windows Forms е стандартната библиотека на .NET Framework за изграждане на прозоречно-базиран графичен потребителски интерфейс (GUI) за настолни (desktop) приложения. Windows Forms дефинира набор от класове и типове, позволяващи изграждане на прозорци и диалози с графични контроли в тях, чрез които се извършва интерактивно взаимодействие с потребителя.

При настолните приложения графичният потребителски интерфейс позволява потребителят директно да взаимодейства с програмата чрез мишката и клавиатурата, а програмата прихваща неговите действия и ги обработва по подходящ начин.

В .NET Framework и особено в Windows Forms се поддържа концепцията за Rapid Application Development (RAD). RAD е подход за разработка, при който приложенията се създават визуално чрез сглобяване на готови компоненти посредством помощници и инструменти за автоматично генериране на голяма част от кода.

I. ИЗГРАЖДАНЕ НА ГРАФИЧЕН ПОТРЕБИТЕЛСКИ ИНТЕРФЕЙС С WINDOWS FORMS



При компонентно-ориентираната разработка **всеки компонент решава някаква определена задача, която е част от проекта**. Компонентите се поставят в приложението, след което се интегрират един с друг чрез настройка на техните свойства и събития. Свойствата на всеки компонент определят различни негови характеристики, а събитията служат за управление на действията, които са предизвикани от него.

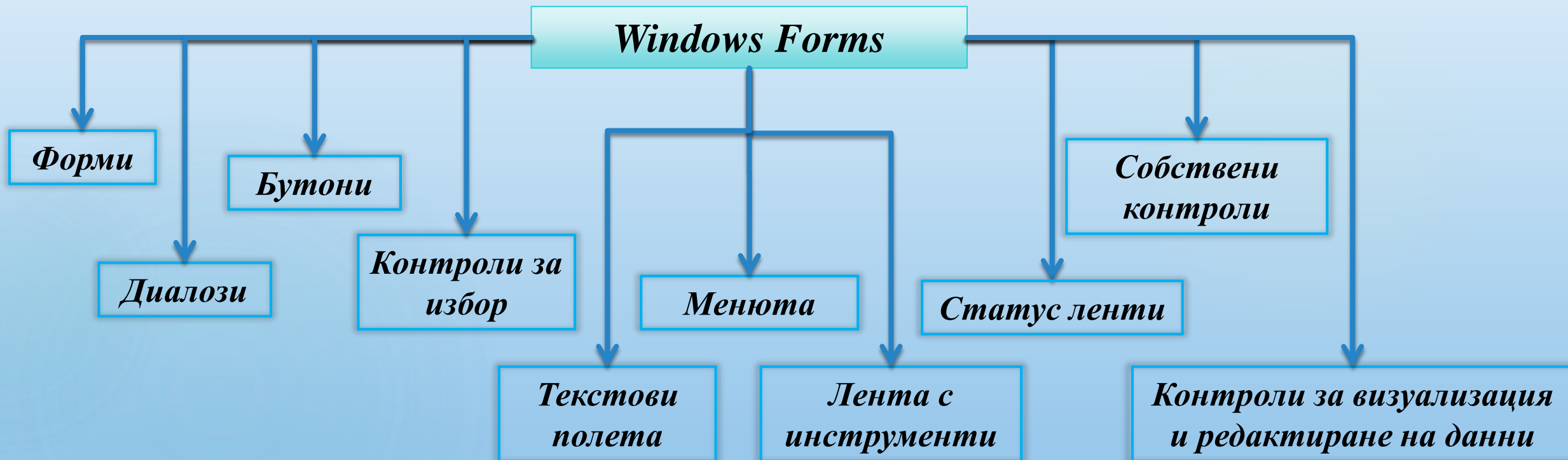
Windows Forms е типична компонентно-ориентирана библиотека за създаване на GUI, която предоставя възможност с малко писане на програмен код да се създава гъвкав графичен потребителски интерфейс.

Windows Forms позволява създаването на формите и другите елементи от графичния интерфейс на приложенията да се извършва визуално и интуитивно чрез подходящи редактори, като например Windows Forms Designer във Visual Studio .NET.

I. ИЗГРАЖДАНЕ НА ГРАФИЧЕН ПОТРЕБИТЕЛСКИ ИНТЕРФЕЙС С WINDOWS FORMS

3

Контролите в Windows Forms



I. ИЗГРАЖДАНЕ НА ГРАФИЧЕН ПОТРЕБИТЕЛСКИ ИНТЕРФЕЙС С WINDOWS FORMS



Библиотеките на .NET за изграждане на GUI

Средствата на .NET Framework за изграждане на графичен потребителски интерфейс са дефинирани в пространствата от имена **System.Drawing** и **System.Windows.Forms**, които са реализирани съответно в асемблитата **System.Drawing.dll** и **System.Windows.Forms.dll**.

System.Windows.Forms

Design

System.Drawing

Drawing2D

Printing

Imaging

Text

System.Windows.Forms - съдържа класове, които поддържат конфигурирането на компонентите и дефинират поведението на Windows Forms контролите по време на дизайн.

System.Drawing – позволява работа с повърхности, точки, линии, четки, моливи, геометрични фигури, картинки, текст и шрифтове и др.

II. ПРОГРАМНИ КОМПОНЕНТИ



В софтуерното инженерство **компонентите са преизползваеми (reusable) програмни единици** (класове), **които решават специфична задача**. Всеки компонент има **ясно дефиниран интерфейс, който описва неговите свойства, методи и събития**. Компонентите се използват като части от други компоненти или програми – те са градивните елементи на софтуера.

Компоненти и контейнери

В .NET Framework са дефинирани два вида преизползваеми обекти:

- **компоненти**
- **контейнери**

Компонентите са функционални единици, които решават някаква задача, а контейнерите са обекти, които съдържат списък от компоненти.

II. ПРОГРАМНИ КОМПОНЕНТИ



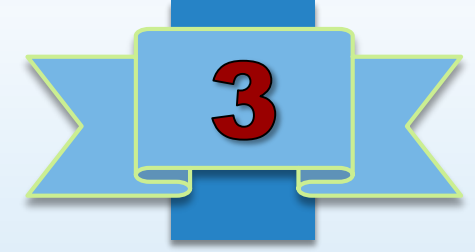
Контроли и контейнер-контроли

Контролите в Windows Forms са всички компоненти, които са видими за потребителя (имат графично изображение). Те биват два вида:

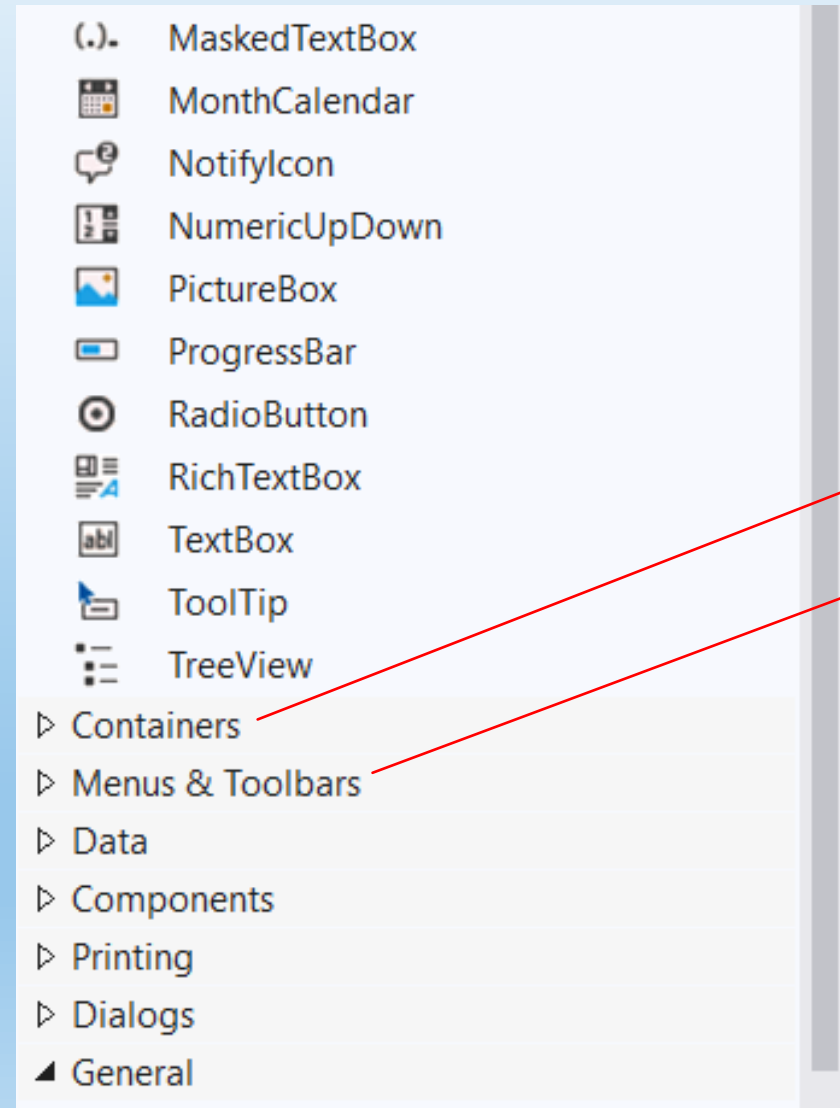
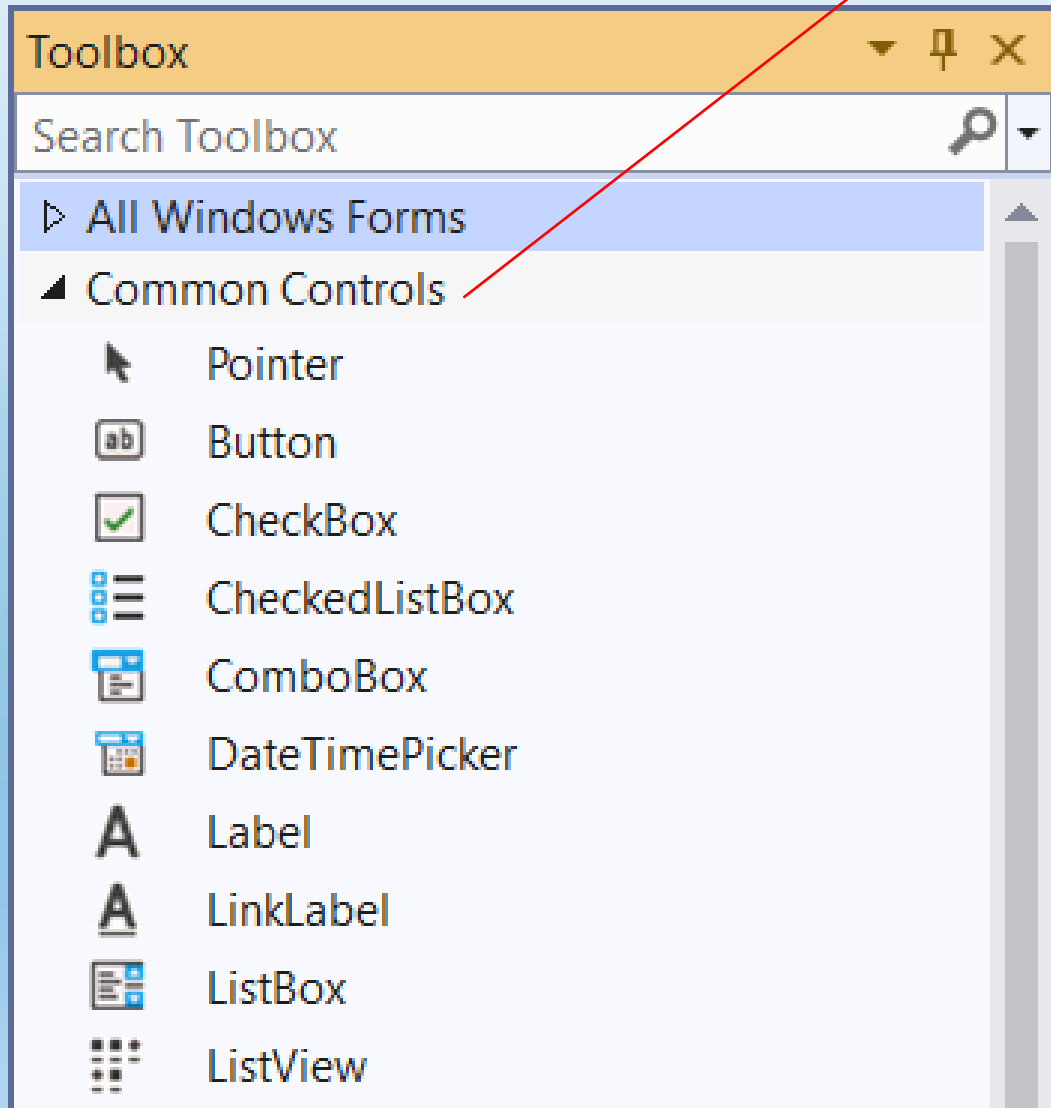
- **контейнер-контроли** (форми, диалози, панели и т.н.)
- **контроли** (бутони, текстови полета, етикети, списъчни контроли и т.н.)

Контейнерите са предназначени да съдържат в себе си други контроли (включително и други контейнер-контроли), докато контролите са предназначени да се съдържат в контейнерите.

II. ПРОГРАМНИ КОМПОНЕНТИ



Основни контролни единици



Контейнери

Менюта и toolbar инструменти

III. ПРОГРАМЕН МОДЕЛ НА WINDOWS FORMS

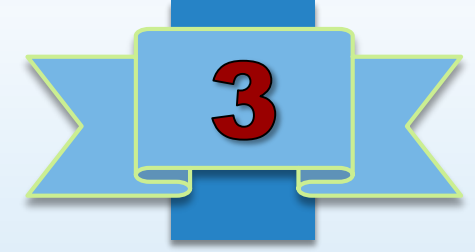


Програмният модел на библиотеката Windows Forms дефинира класовете за работа с форми, диалози и контроли, събитията на контролите, жизнения цикъл на приложенията, модела на пречертаване на контролите, модела на получаване и обработка на събитията и модела на управление на фокуса.

1. Форми - Windows Forms предлага стандартни класове за работа с форми (**това са прозорците и диалозите** в GUI приложенията). Формите могат да бъдат **модални и немодални (по една или по много активни едновременно)**. Формите са контейнер-контроли и могат да съдържат други контроли, например етикети, текстови полета, бутони и т.н. Базов клас за всички форми е класът `System.Windows.Forms.Form`.

2. Контроли - Контролите в Windows Forms **са текстовите полета, етикетите, бутоните, списъците, дърветата, таблиците, менютата, лентите с инструменти, статус лентите и много други**. Windows Forms дефинира базови класове за контролите и класове-наследници за всяка контрола. **Базов клас** за всички контроли е класът `System.Windows.Forms.Control`. Пример за контрола е например бутонът (класът `System.Windows.Forms.Button`).

III. ПРОГРАМЕН МОДЕЛ НА WINDOWS FORMS



3. Събития - Всички контроли от Windows Forms дефинират събития, които програмистът може да прихваща. Например контролата **Button** дефинира събитието **Click**, което се активира при натискане на бутона. Събитията в Windows Forms управляват взаимодействието между програмата и контролите и между самите контроли.

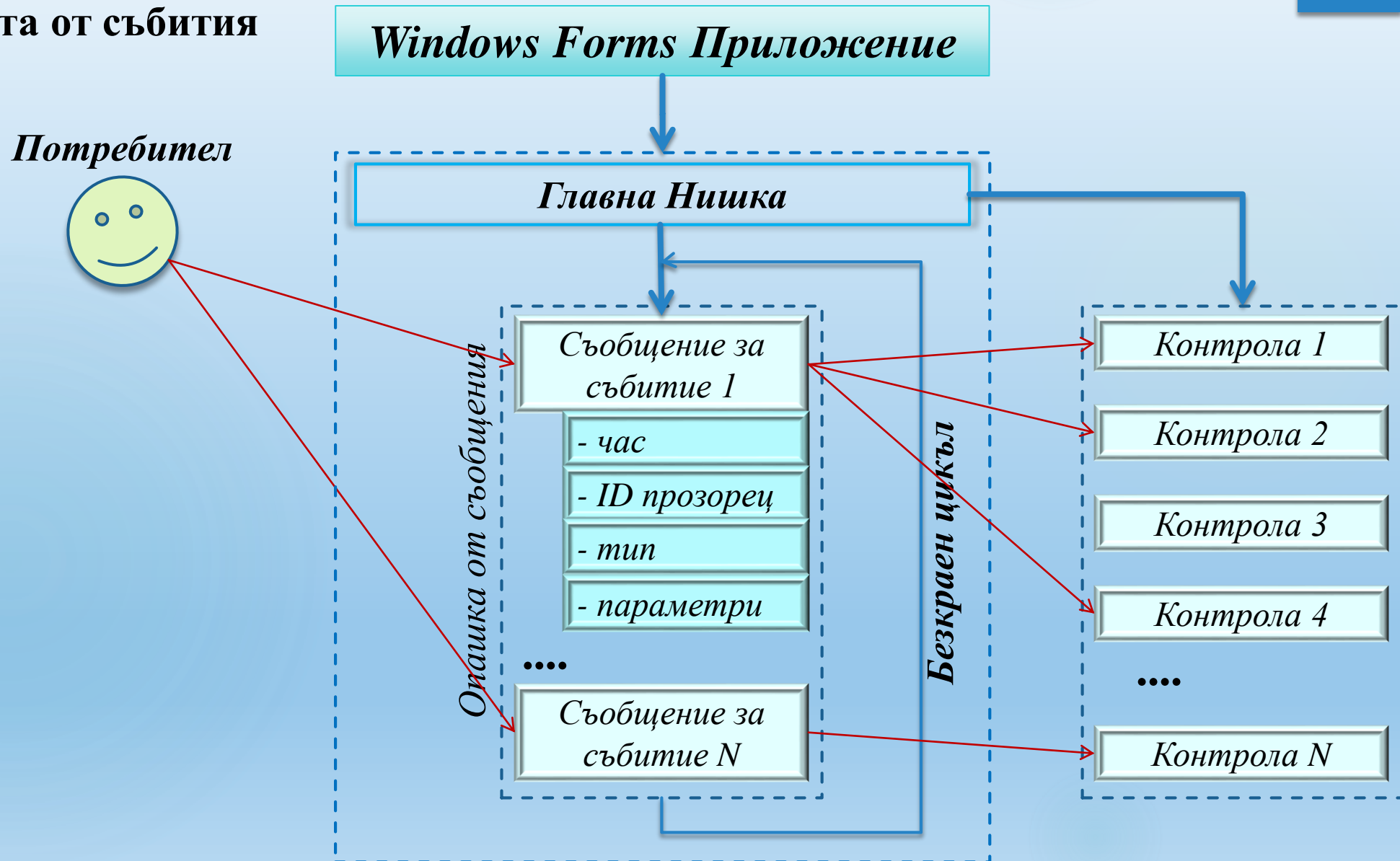
4. Жизнен цикъл на Windows Forms приложенията - Жизненият цикъл на GUI приложенията е базиран на съобщения. **Графичната среда на операционната система прихваща всички потребителски действия (напр. движението на мишката, натискането на клавиши от клавиатурата и т.н.) и ги натрупва в специална опашка.** След това всяко съобщение се предава към приложението, за което се отнася и по-точно към нишката (thread) от приложението, за която се отнася.

- Нишки и многозадачност - В многозадачните операционни системи (каквито са например Windows и Linux) е възможно едно приложение да изпълнява няколко задачи паралелно, като използва няколко нишки (threads) в рамките на процеса, в който работи програмата.

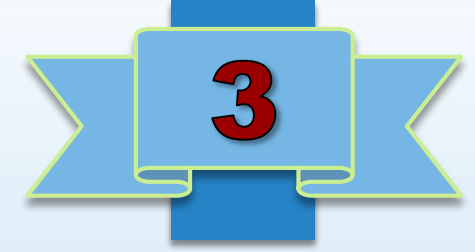
III. ПРОГРАМЕН МОДЕЛ НА WINDOWS FORMS



- Опашката от събития



III. ПРОГРАМЕН МОДЕЛ НА WINDOWS FORMS



- Всяка нишка от всяко приложение си има своя собствена опашка, в която постъпват съобщенията за всички събития, идващи от потребителя или от други източници.
- Всяко съобщение носи информация за събитието, което е настъпило – часът на настъпване, идентификатор на прозорец, за който се отнася събитието, тип на събитието, параметри на събитието (напр. номер на натиснатия клавиш при събитие от клавиатурата или позиция на курсора при събитие от мишката).
- Главната нишка на всяко Windows Forms приложение извършва една единствена задача: в безкраен цикъл обработва опашката от съобщения за приложението и предава постъпилите съобщения на контролата, за която са предназначени.
- В Windows Forms приложенията винаги имат точно една нишка, която обработва всички съобщения, идващи от графичните контроли, и това е главната нишка на приложението.

III. ПРОГРАМЕН МОДЕЛ НА WINDOWS FORMS



Само главната нишка трябва да взаимодейства с опашката от събития

Много е важно, когато разработваме Windows Forms приложения, да се съобразяваме със следното правило:



Графичният потребителски интерфейс на приложението трябва да се управлява само и единствено от неговата главна нишка.

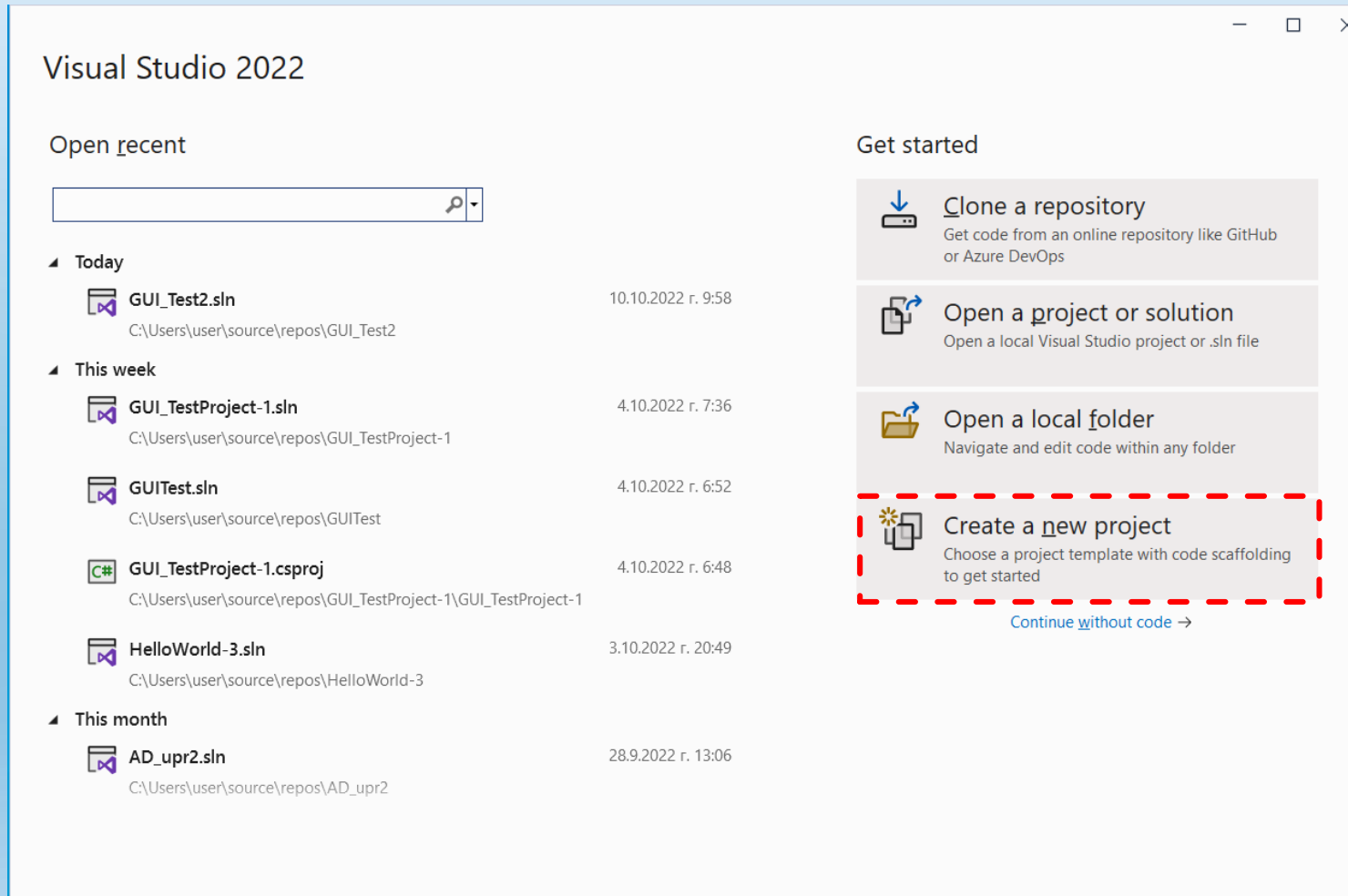
Прекратяване на Windows Forms приложение

При затваряне на главната форма на Windows Forms приложение, към нея се изпраща съобщение за затваряне. Формата се затваря в момента, в който получи съобщението и го обработи. В резултат на затварянето на формата се прекратява цикълът, в който главната нишка на приложението обработва пристигащите за нея съобщения и приложението приключва изпълнението си.

IV. СЪЗДАВАНЕ НА WINDOWS FORMS ПРОЕКТ

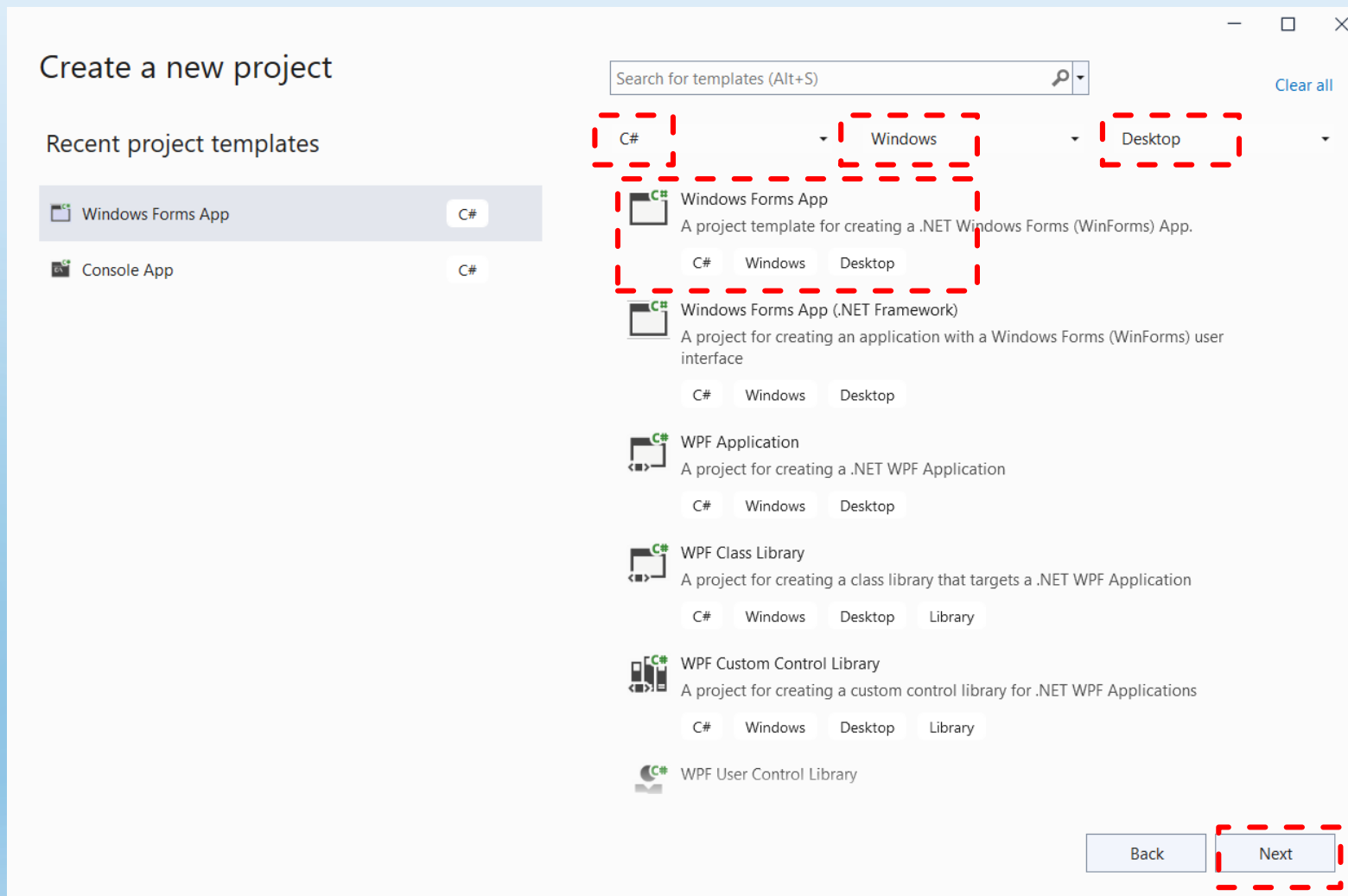


1. Създаване на нов Windows Forms проект



IV. СЪЗДАВАНЕ НА WINDOWS FORMS ПРОЕКТ

3



IV. СЪЗДАВАНЕ НА WINDOWS FORMS ПРОЕКТ

3

Configure your new project

Windows Forms App C# Windows Desktop

Project name

GUI_TEST_Project_3

Location

C:\Users\user\source\repos

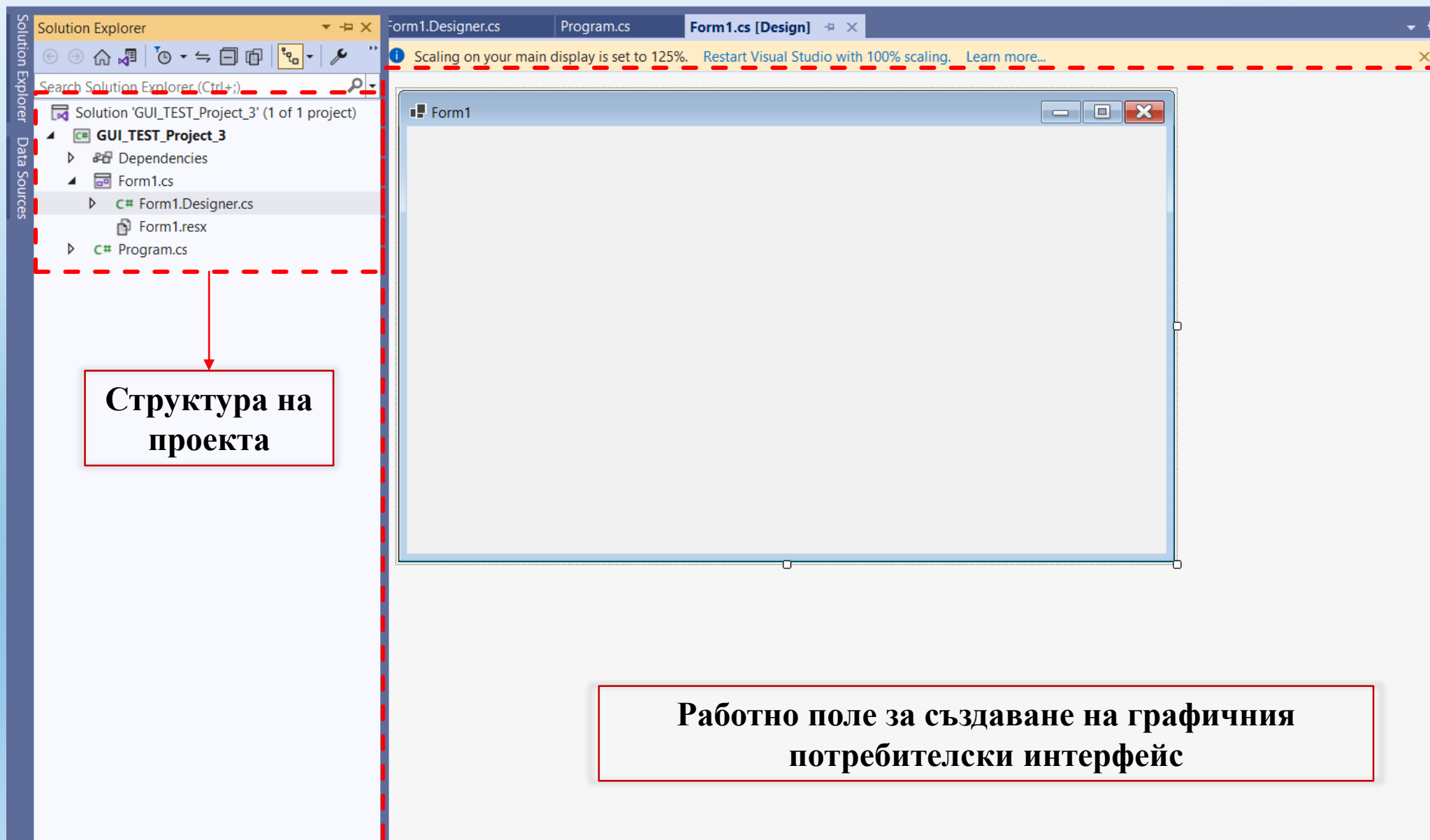
Solution name ⓘ

GUI_TEST_Project_3

☐ Place solution and project in the same directory

IV. СЪЗДАВАНЕ НА WINDOWS FORMS ПРОЕКТ

3



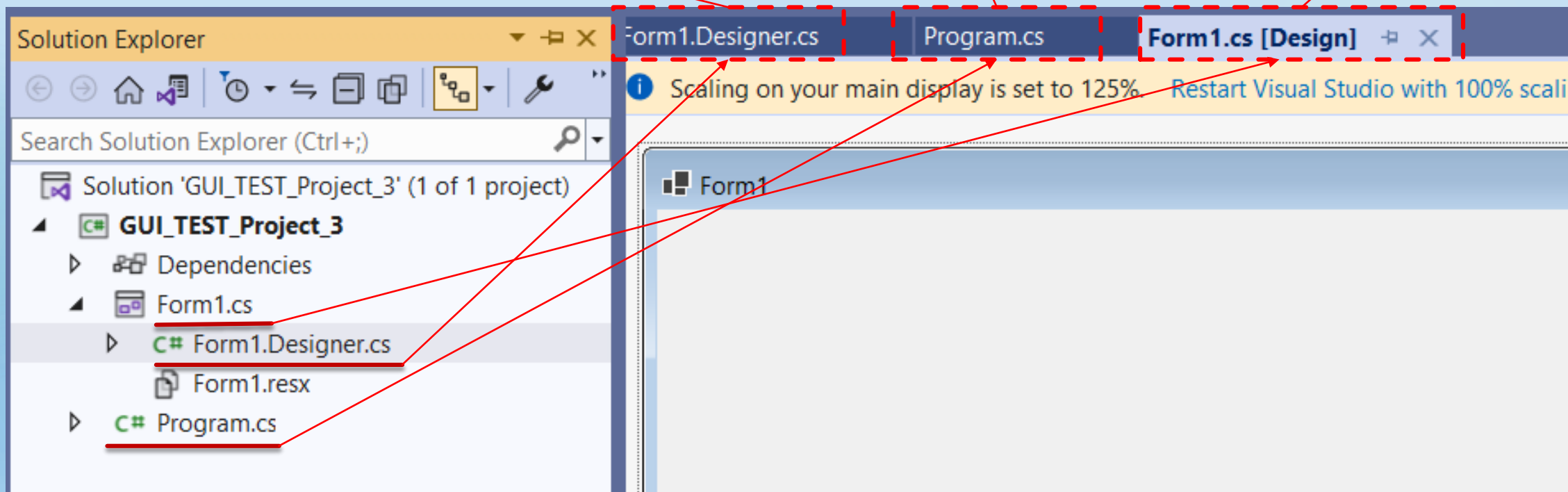
IV. СЪЗДАВАНЕ НА WINDOWS FORMS ПРОЕКТ

3

Файл с код описващ всеки
един от графичните
компоненти

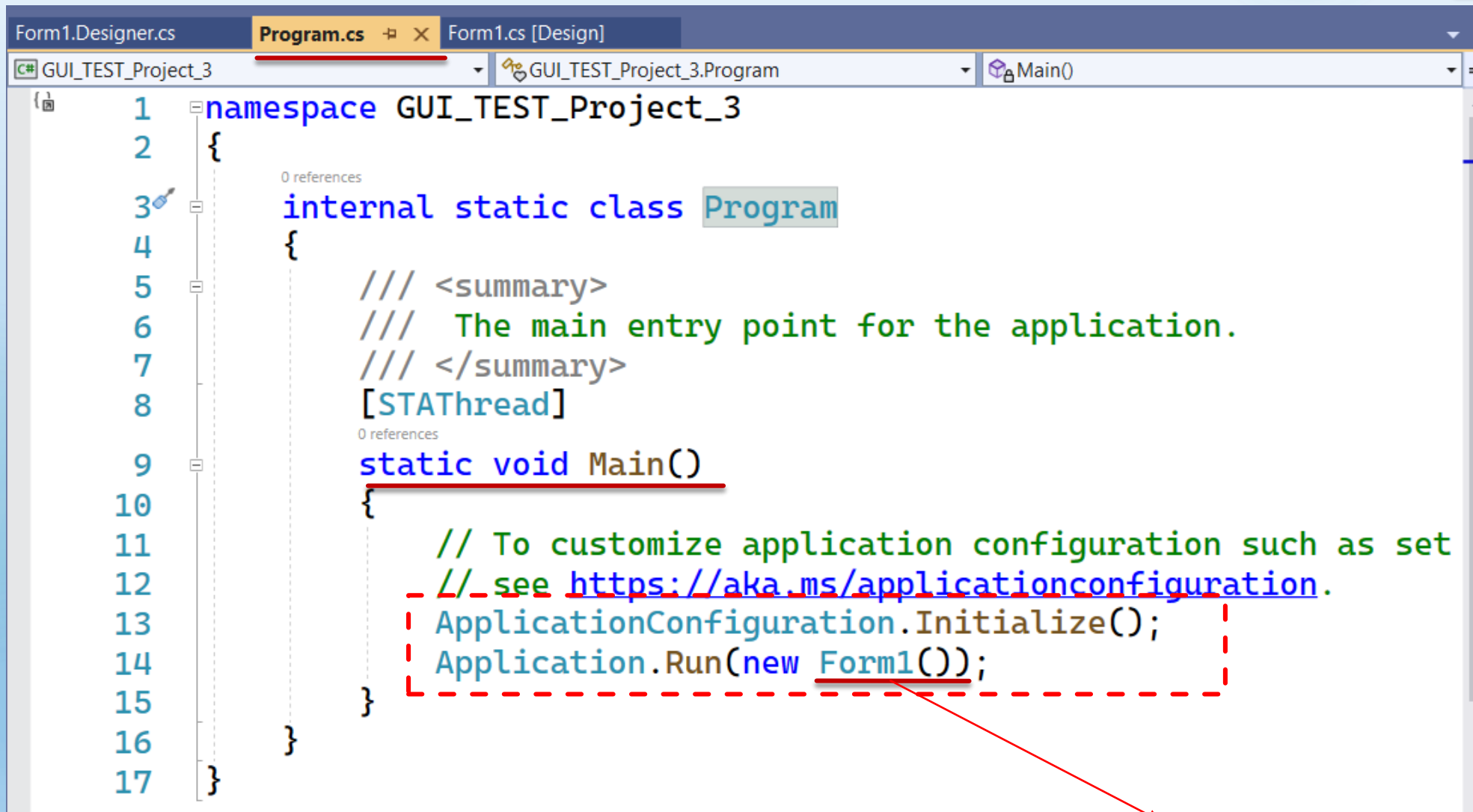
Файл съдържащ
Main() метода

Прозорец за
изграждане на дизайна



IV. СЪЗДАВАНЕ НА WINDOWS FORMS ПРОЕКТ

3

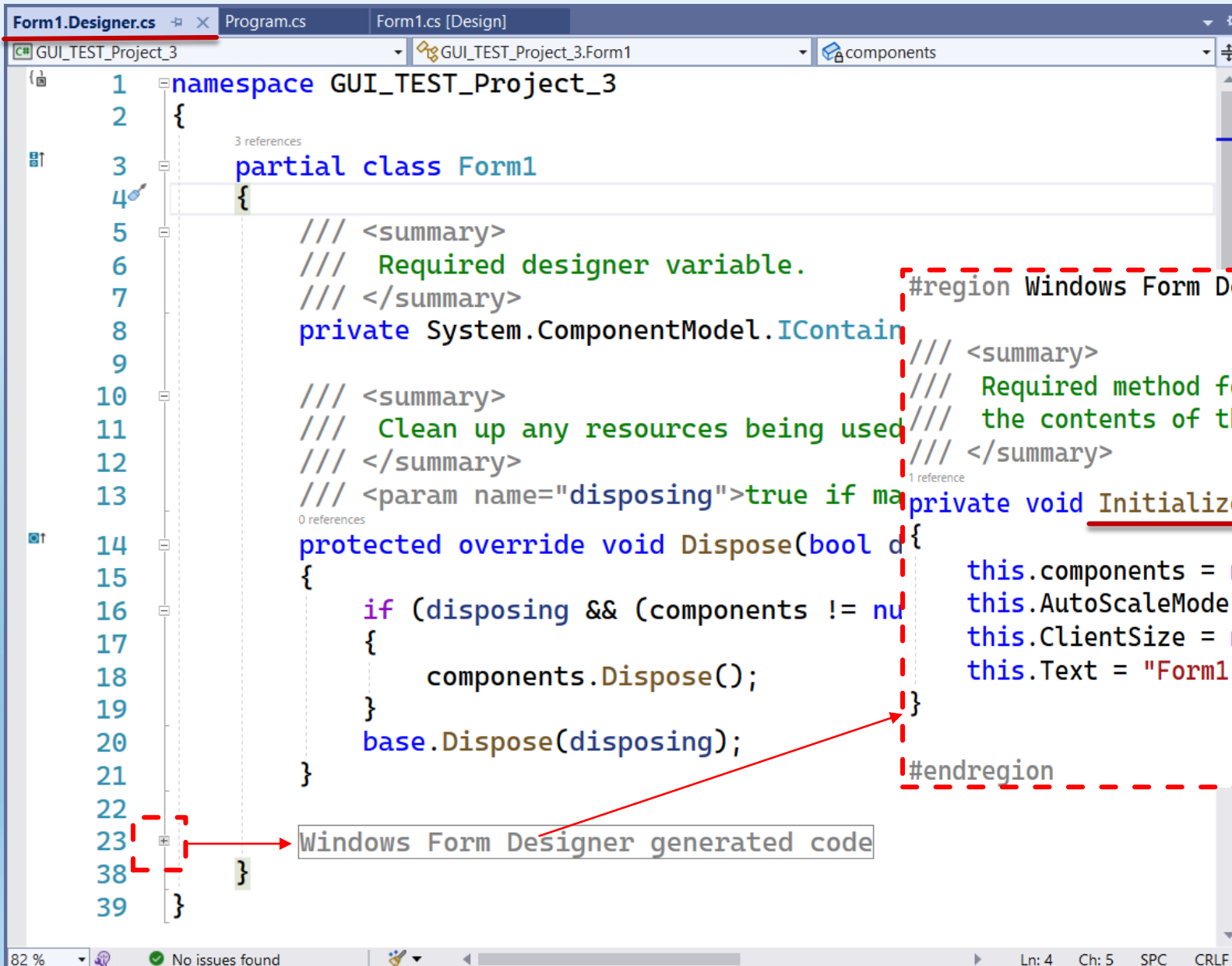


```
1 namespace GUI_TEST_Project_3
2 {
3     internal static class Program
4     {
5         /// <summary>
6         /// The main entry point for the application.
7         /// </summary>
8         [STAThread]
9         static void Main()
10        {
11            // To customize application configuration such as set
12            // see https://aka.ms/applicationconfiguration.
13            ApplicationConfiguration.Initialize();
14            Application.Run(new Form1());
15        }
16    }
17 }
```

Стартиране на класа Form1, съдържащ програмния код за графичните компоненти

IV. СЪЗДАВАНЕ НА WINDOWS FORMS ПРОЕКТ

3



```
Form1.Designer.cs Program.cs Form1.cs [Design]
GUI_TEST_Project_3 GUI_TEST_Project_3.Form1 components
1 namespace GUI_TEST_Project_3
2 {
3     3 references
4     partial class Form1
5     {
6         /// <summary>
7         /// Required designer variable.
8         /// </summary>
9         private System.ComponentModel.IContainer components = null;
10
11         /// <summary>
12         /// Clean up any resources being used.
13         /// </summary>
14         /// <param name="disposing">true if managed objects should be disposed; otherwise, false;
15         /// </param>
16         protected override void Dispose(bool disposing)
17         {
18             if (disposing && (components != null))
19             {
20                 components.Dispose();
21             }
22             base.Dispose(disposing);
23         }
24
25         #region Windows Form Designer generated code
26
27         /// <summary>
28         /// Required method for Designer support - do not modify
29         /// the contents of this method with the code editor.
30         /// </summary>
31         private void InitializeComponent()
32         {
33             this.components = new System.ComponentModel.Container();
34             this.AutoScaleMode = System.Windows.Forms.AutoScaleMode.Font;
35             this.ClientSize = new System.Drawing.Size(800, 450);
36             this.Text = "Form1";
37         }
38
39         #endregion
40     }
41 }
```

Инициализация на графичните
компоненти

IV. СЪЗДАВАНЕ НА WINDOWS FORMS ПРОЕКТ

3

