

Lập trình trên môi trường Windows

Nguyễn Ánh

n.anh@hutech.edu.vn



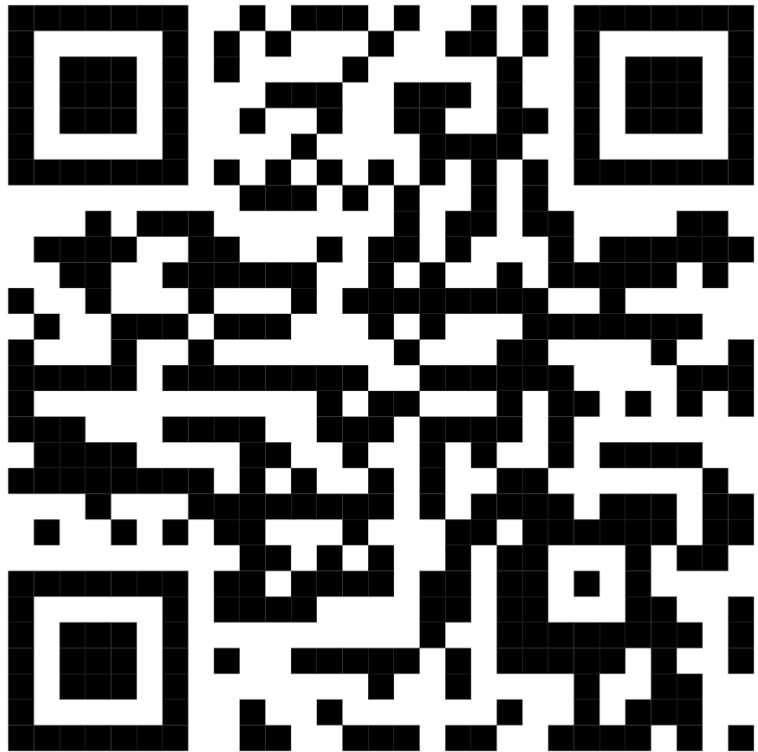
Giới thiệu CMP170

- Tên học phần: **Lập trình trên môi trường Windows**
- Số tín chỉ: 3
- Phân bổ thời gian:
 - Giờ giảng lý thuyết: 45 tiết

Đánh giá

- Điểm quá trình = Điểm chuyên cần + Điểm bài tập, kiểm tra
 - Điểm chuyên cần: 20%
 - Điểm bài tập, kiểm tra: 30%
- Đồ án học phần
 - Đánh giá kết quả: 50%
- **Điểm học phần** = Điểm quá trình + Thi cuối kỳ

Nhóm cho môn học

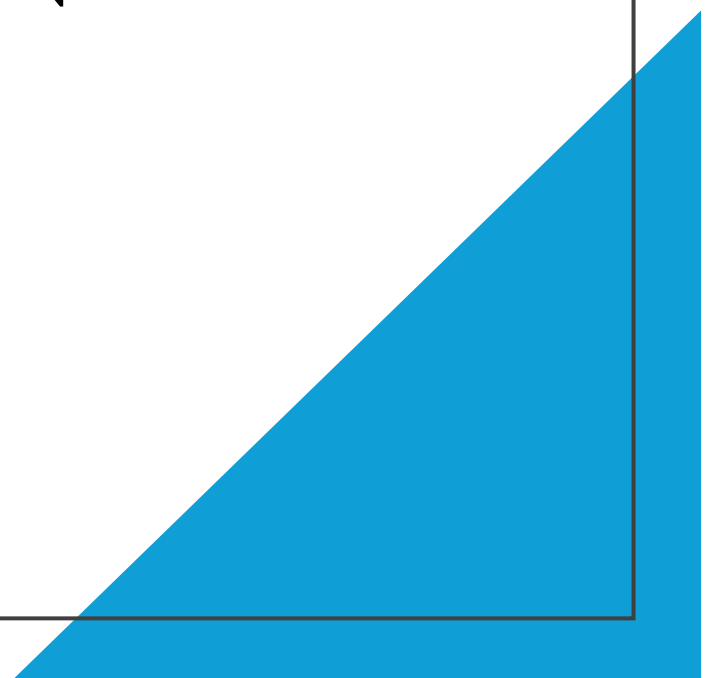


Zalo



Google Class

QUY TẮC VIẾT MÃ C# THEO CHUẨN MICROSOFT



Các quy tắc đặt tên biến

- Tên định danh (identifier) phải bắt đầu bằng một chữ cái (letter) hoặc một dấu gạch dưới (_).
- Sau ký tự đầu tiên, tên định danh có thể chứa các ký tự chữ cái Unicode, chữ số thập phân, ký tự nối Unicode.
- Sử dụng tiền tố @ (verbatim identifier). Bạn có thể sử dụng một từ khóa của C# làm tên định danh bằng cách đặt ký tự @ ở phía trước.

Ví dụ:

- `string _tenSanPham;`
- `double diemTrungBinh;`
- `string tênĐăngNhập;` (Không nên)
- `bool @while = true;` (Không nên)

Mục đích

- **Dễ Đọc (Readability):** Tên biến rõ ràng giúp người khác (và cả chính bạn trong tương lai) nhanh chóng hiểu được mục đích của biến, phương thức, hay lớp đó là gì mà không cần đọc toàn bộ logic bên trong.
- **Dễ Bảo Trì (Maintainability):** Khi cần sửa lỗi hay thêm tính năng mới, việc tìm kiếm và xác định đúng thành phần cần thay đổi sẽ nhanh hơn rất nhiều.
- **Tính Chuyên Nghiệp:** Viết mã theo chuẩn cho thấy bạn là một lập trình viên có kỷ luật và chuyên nghiệp.
- **Tránh Nhầm Lẫn:** Các quy tắc giúp phân biệt rõ ràng đâu là một lớp, đâu là một phương thức, đâu là một biến cục bộ.

Quy tắc Viết hoa: PascalCase và camelCase

- Có hai kiểu viết hoa chính thường được sử dụng:
- **PascalCase (hay UpperCamelCase)**: Viết hoa chữ cái đầu tiên của mỗi từ (kể cả từ đầu tiên).
- Ví dụ: FileManager, CalculateTotalPrice, CustomerData.
- **camelCase (hay lowerCamelCase)**: Viết thường chữ cái của từ đầu tiên, và viết hoa chữ cái đầu của các từ tiếp theo.
- Ví dụ: fileManager, calculateTotalPrice, customerData.

Quy ước đặt tên biến

- Đặt tên cho class, struct, enum, delegate và interface
- Quy tắc: Sử dụng **PascalCase**.
- Đối với interface: Sử dụng **PascalCase** và thêm tiền tố là một chữ I viết hoa ở đầu.

```
// Class và Struct
public class SinhVien
{
    //...
}

public struct Point
{
    //...
}

// Enum
public enum GioiTinh
{
    Nam,
    Nu,
    Khac
}

// Interface
public interface IQuanLyDuLieu
{
    void Them();
    void Xoa();
}
```

Đặt tên cho thành viên Public của Class (Public Members)

- Các phương thức (Method), thuộc tính (Property), sự kiện (Event)
- **Quy tắc:** Sử dụng **PascalCase**

```
public class SinhVien
{
    // Property
    public string HoTen { get; set; }

    // Method
    public double TinhDiemTrungBinh()
    {
        // ...
        return 0.0;
    }

    // Event
    public event EventHandler DiemThayDoi;
}
```

Đặt tên cho Biến (Variables)

- Quy tắc cho tham số phương thức (Method Parameters): Sử dụng **camelCase**.
- Quy tắc cho biến cục bộ (Local Variables): Sử dụng **camelCase**.

```
// Tham số dùng camelCase
public void CapNhatThongTin(string maSinhVien, string hoTenMoi)
{
    // Biến cục bộ dùng camelCase
    int soKyTu = hoTenMoi.Length;

    if (soKyTu > 5)
    {
        // ...
    }
}
```

**Đặt tên cho thành viên
Private hoặc Internal của
Class (Private/Internal
Fields)**

Quy tắc: Sử dụng dấu gạch dưới (_) ở đầu và theo sau là **camelCase**.

```
public class SinhVien
{
    // Private field
    private string _maSinhVien;

    // Constructor
    public SinhVien(string maSinhVien)
    {
        // _maSinhVien là field của class
        // maSinhVien là tham số của constructor
        _maSinhVien = maSinhVien;
    }
}
```

Đặt tên biến cho Controls

- **Quy ước chung:** [Tiền tố viết tắt] + [Mục đích theo PascalCase]
- **Tiền tố (prefix):** 3 chữ cái viết thường để nhận diện loại control.
- **Mục đích:** Mô tả chức năng của control đó, viết theo PascalCase.

Đặt tên biến cho Controls

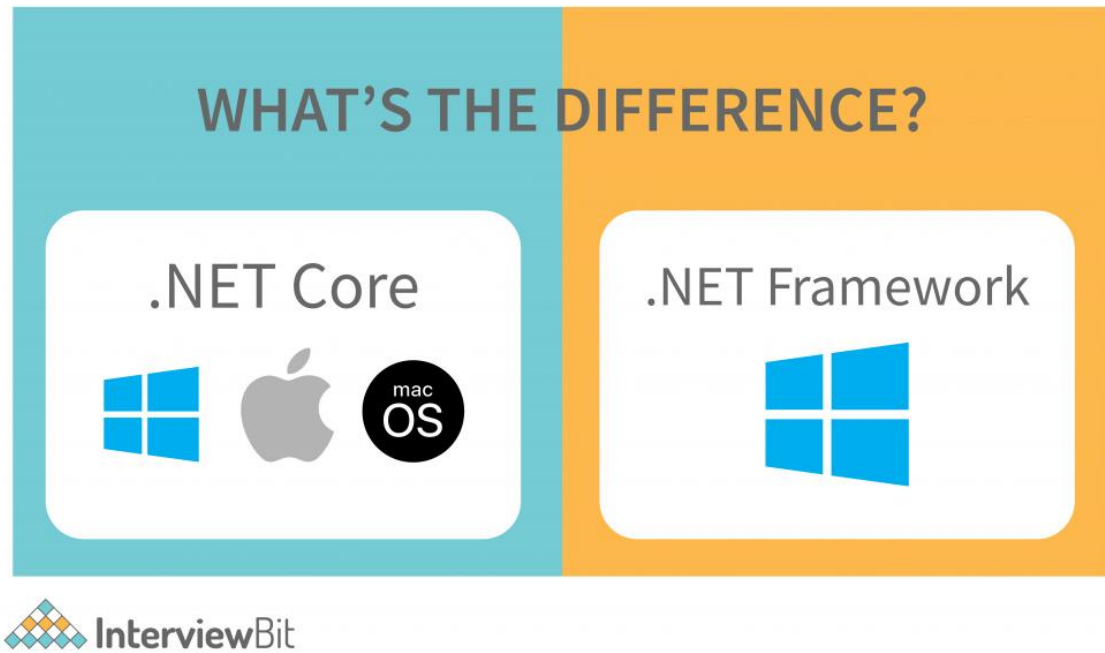
Control	Tiền tố	Ví dụ
Button	btn	btnLuu, btnThoat, btnXacNhan
TextBox	txt	txtHoTen, txtDiaChi, txtTimKiem
Label	lbl	lblTieuDe, lblThongBaoLoi
ComboBox	cbo	cboChonThanhPho, cboDanhSachLop
ListBox	lst	lstMonHoc
DataGridView	dgv	dgvDanhSachSinhVien
RadioButton	rad	radNam, radNu
CheckBox	chk	chkDongYDieuKhoan
MenuStrip	mnu	mnuFile, mnuHeThong
PictureBox	pic	picAnhDaiDien

Đặt tên biến cho Controls

Ví dụ trên Form Đăng nhập:

- TextBox Tên đăng nhập: txtTenDangNhap
- TextBox Mật khẩu: txtMatKhau
- Button Đăng nhập: btnDangNhap
- Button Thoát: btnThoat
- Label thông báo lỗi: lblThongBao

Nền tảng.NET



	.NET Framework	.NET (Core)
Phiên bản	Phiên bản cuối cùng: 4.8	Phiên bản mới nhất: .NET 9
Nền tảng:	Chỉ chạy trên Windows	Đa nền tảng (Windows, macOS, Linux).
Tình trạng	Vẫn được hỗ trợ (và lỗi bảo mật) vì là một phần của Windows. Không có tính năng mới.	Đang được phát triển tích cực, liên tục có tính năng và cải tiến hiệu năng mới.

Nền tảng.NET

<https://dotnet.microsoft.com/en-us/download>

Build it with .NET



Web

Build web apps and services for macOS, Windows, Linux, and Docker.



Mobile and desktop

Use a single codebase to build native apps for Windows, macOS, iOS, and Android.



Cloud

Build scalable and resilient cloud-native apps that run on all major cloud providers.



Artificial Intelligence and ML

Build smart apps with C#, OpenAI, and Azure.

Các công cụ viết mã nguồn

.NET Tools & Editors

Development tools for Windows, Linux, and macOS



Visual Studio

[Download Visual Studio →](#)



Visual Studio Code

[Download Visual Studio Code →](#)



JetBrains Rider

[Download JetBrains Rider →](#)

- <https://dotnet.microsoft.com/en-us/platform/tools>

Bộ công cụ phát triển

- <https://dotnet.microsoft.com/en-us/download>

Free. Cross-platform. Open source.

Download .NET

For Windows

.NET 9.0

Standard Term Support Recommended

[Download .NET SDK x64](#)

Version 9.0.6, released June 10, 2025

[All .NET 9.0 downloads](#) [.NET 9.0 runtime](#) [All .NET versions](#)

.NET 8.0

Long Term Support

[Download .NET SDK x64](#)

Version 8.0.17, released June 10, 2025

[All .NET 8.0 downloads](#) [.NET 8.0 runtime](#)

NET SDK (Software Development Kit)

- Là một siêu tập hợp bao gồm toàn bộ công cụ, thư viện và trình biên dịch cần thiết cho một nhà phát triển để xây dựng (build) và phát triển (develop) ứng dụng .NET từ mã nguồn.

Các thành phần cốt lõi:

- **Trình biên dịch Roslyn:** Chuyển hóa mã nguồn C# thành **Ngôn ngữ Trung gian (Intermediate Language - IL)**.
- **.NET CLI (Command-Line Interface):** Công cụ dotnet cho phép thực thi các tác vụ như dotnet build, dotnet publish, dotnet test...
- **Thư viện và Templates:** Toàn bộ thư viện nền tảng và các mẫu dự án (project templates).
- **Bao gồm cả .NET Runtime:** Để lập trình viên có thể chạy và gỡ lỗi (debug) ứng dụng ngay trong môi trường phát triển.
- **Đối tượng sử dụng:** Lập trình viên (**Developers**) và các hệ thống **Tích hợp/Triển khai Liên tục (CI/CD Pipelines)**.

NET Runtime - Môi trường Thực thi

- **Định nghĩa:** Là tập hợp các thành phần **tối thiểu** cần thiết để **thực thi (execute)** một ứng dụng .NET đã được biên dịch sẵn.
- **Các thành phần cốt lõi:**
- **Common Language Runtime (CLR):** Là thành phần chính của .NET, chịu trách nhiệm:
 - **Trình biên dịch Just-In-Time (JIT):** Dịch mã IL sang mã máy (native machine code) tại thời điểm chạy.
 - **Bộ thu dọn rác (Garbage Collector - GC):** Quản lý bộ nhớ tự động.
 - **Hệ thống quản lý kiểu (Type System) và cơ chế bảo mật (Security).**
- **Thư viện cốt lõi (Core Libraries):** Chỉ bao gồm những thư viện thiết yếu nhất cho việc vận hành ứng dụng.
- **Đối tượng sử dụng: Người dùng cuối (End-users) và các máy chủ sản phẩm (Production Servers).**

So sánh .NET SDK - Runtime

Tiêu chí	.NET SDK	.NET Runtime
Mục đích chính	Phát triển & Xây dựng (Develop & Build)	Thực thi & Vận hành (Execute & Run)
Đối tượng	Lập trình viên, Máy chủ Build	Người dùng cuối, Máy chủ Production
Thành phần	Trình biên dịch (Roslyn), CLI, Thư viện đầy đủ, bao gồm cả Runtime.	CLR (JIT, GC), Thư viện cốt lõi.
Kích thước	Lớn	Nhỏ gọn

Download .NET 9.0

Not what you're looking for? Visit the [downloads](#) page for more options.

^ 9.0.6 Security patch

[Release notes](#) Latest release date June 10, 2025

Build apps - SDK

SDK 9.0.301

OS	Installers	Binaries
Linux	Package manager instructions	Arm32 Arm32 Alpine Arm64 Arm64 Alpine x64 x64 Alpine
macOS	Arm64 x64	Arm64 x64
Windows	x64 x86 Arm64 winget instructions	x64 x86 Arm64
All	dotnet-install scripts	

Visual Studio support

Visual Studio 2022 (v17.14)

Included in

Visual Studio 17.14.5

Included runtimes

.NET Runtime 9.0.6
ASP.NET Core Runtime 9.0.6
.NET Desktop Runtime 9.0.6

Language support

C# 13.0
F# 9.0
Visual Basic 17.13

SDK 9.0.205

OS	Installers	Binaries
Linux	Package manager instructions	Arm32 Arm32 Alpine Arm64 Arm64 Alpine x64 x64 Alpine
macOS	Arm64 x64	Arm64 x64
Windows	x64 x86 Arm64 winget instructions	x64 x86 Arm64
All	dotnet-install scripts	

Included runtimes

.NET Runtime 9.0.6
ASP.NET Core Runtime 9.0.6
.NET Desktop Runtime 9.0.6

Language support

C# 13.0
F# 9.0
Visual Basic 17.13

Run apps - Runtime

ASP.NET Core Runtime 9.0.6

The ASP.NET Core Runtime enables you to run existing web/server applications. **On Windows, we recommend installing the Hosting Bundle, which includes the .NET Runtime and IIS support.**

IIS runtime support (ASP.NET Core Module v2)

19.0.25137.6

OS	Installers	Binaries
Linux	Package manager instructions	Arm32 Arm32 Alpine Arm64 Arm64 Alpine x64 x64 Alpine
macOS		Arm64 x64
Windows	x64 x86 Arm64 Hosting Bundle winget instructions	x64 x86 Arm64

.NET Desktop Runtime 9.0.6

The .NET Desktop Runtime enables you to run existing Windows desktop applications. **This release includes the .NET Runtime; you don't need to install it separately.**

OS	Installers	Binaries
Windows	x64 x86 Arm64 winget instructions	

.NET Runtime 9.0.6

The .NET Runtime contains just the components needed to run a console app. Typically, you'd also install either the ASP.NET Core Runtime or .NET Desktop Runtime.

OS	Installers	Binaries
Linux	Package manager instructions	Arm32 Arm32 Alpine Arm64 Arm64 Alpine x64 x64 Alpine
macOS	Arm64 x64	Arm64 x64
Windows	x64 x86 Arm64 winget instructions	x64 x86 Arm64
All	dotnet-install scripts	

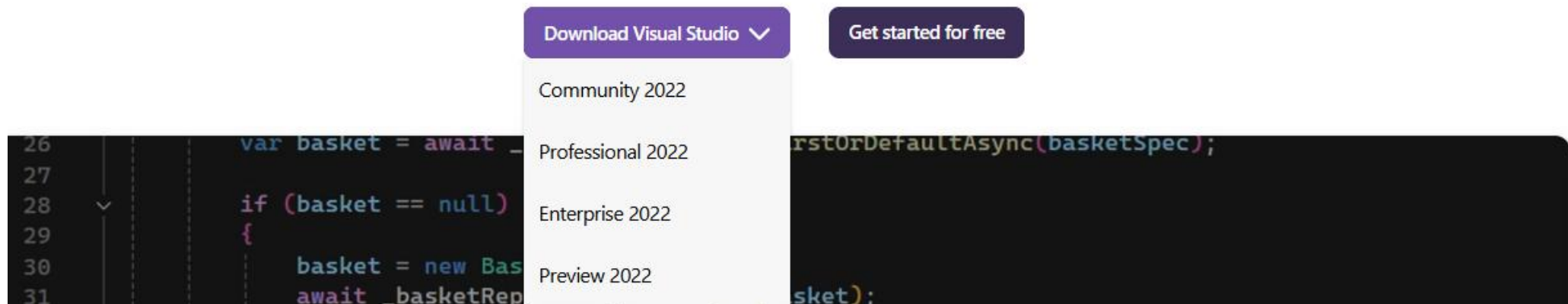
Phần mềm Visual Studio 2022

- <https://visualstudio.microsoft.com/vs/>

Powerful IDE, made smarter with AI

Agent mode – now available in GitHub Copilot Free

Experience AI-powered coding assistance that reasons through problems, coordinates next steps, applies changes, and iterates on errors.



Microsoft Visual Studio Community 2022

Visual Studio 2022 là gì?

- Là một **Môi trường Phát triển Tích hợp** (IDE - Integrated Development Environment) mạnh mẽ từ Microsoft.
- Cung cấp bộ công cụ toàn diện để viết mã, gỡ lỗi (debug), kiểm thử và triển khai ứng dụng.

Tại sao chọn phiên bản Community?

- **Hoàn toàn miễn phí** cho sinh viên, các nhà phát triển nguồn mở và cá nhân.
- **Đầy đủ tính năng** cần thiết cho việc học tập và phát triển hầu hết các loại ứng dụng, bao gồm cả Windows Forms.

Hướng dẫn cài đặt

Truy cập trang chủ và tải về

- Mở trình duyệt web và truy cập vào địa chỉ:
<https://visualstudio.microsoft.com/vs/>
- Tìm và nhấn vào nút **Download Visual Studio**.
- Một menu sẽ xổ xuống. Trong menu này, hãy chọn phiên bản **Community 2022**.
- *(Lưu ý: Phiên bản Professional hay Enterprise là các phiên bản trả phí).*
- Trình duyệt sẽ tự động tải về một file có tên tương tự như VisualStudioSetup.exe.

Workloads

- Là các gói công cụ được cài đặt sẵn cho một mục đích lập trình cụ thể (ví dụ: Lập trình Web, Lập trình Game, Lập trình Desktop,...).

Workload cần chọn cho môn học này:

- Để lập trình ứng dụng Windows Forms bằng C#, các em **BẮT BUỘC** phải tick chọn workload có tên là: **.NET desktop development**
- Workload này bao gồm mọi thứ chúng ta cần:
 - Ngôn ngữ C# và Visual Basic.
 - Framework để tạo ứng dụng Windows Forms, WPF và Console.
- Lưu ý: Chọn cài đặt thêm Github Copilot

Web & Cloud (4)



ASP.NET and web development

Build web applications using ASP.NET Core, ASP.NET, HTML/JavaScript, and Containers including Docker supp...



Azure development

Azure SDKs, tools, and projects for developing cloud apps and creating resources using .NET and .NET Framework...



Python development

Editing, debugging, interactive development and source control for Python.



Node.js development

Build scalable network applications using Node.js, an asynchronous event-driven JavaScript runtime.



Desktop & Mobile (5)



.NET Multi-platform App UI development

Build Android, iOS, Windows, and Mac apps from a single codebase using C# with .NET MAUI.



.NET desktop development

Build WPF, Windows Forms, and console applications using C#, Visual Basic, and F# with .NET and .NET Frame...



Desktop development with C++

Build modern C++ apps for Windows using tools of your



WinUI application development

Build applications for the Windows platform using WinUI



Installation details

✓ C# and Visual Basic

Optional

- ✓ Development tools for .NET
- ✓ .NET Framework 4.8 development tools
- ✓ Entity Framework 6 tools
- ✓ .NET profiling tools
- ✓ IntelliCode
- ✓ Just-In-Time debugger
- ✓ Live Share
- ✓ ML.NET Model Builder
- ✓ GitHub Copilot
- ✓ Blend for Visual Studio
- ☐ F# desktop language support
- ☐ PreEmptive Protection - Dotfuscator
- ☐ .NET Framework 4.6.2-4.7.1 development t...
- ☐ .NET Portable Library targeting pack
- ☐ Windows Communication Foundation
- ☐ SQL Server Express 2019 LocalDB
- ☐ MSIX Packaging Tools

Location

C:\Program Files\Microsoft Visual Studio\2022\Community [Change...](#)

Remove out-of-support components

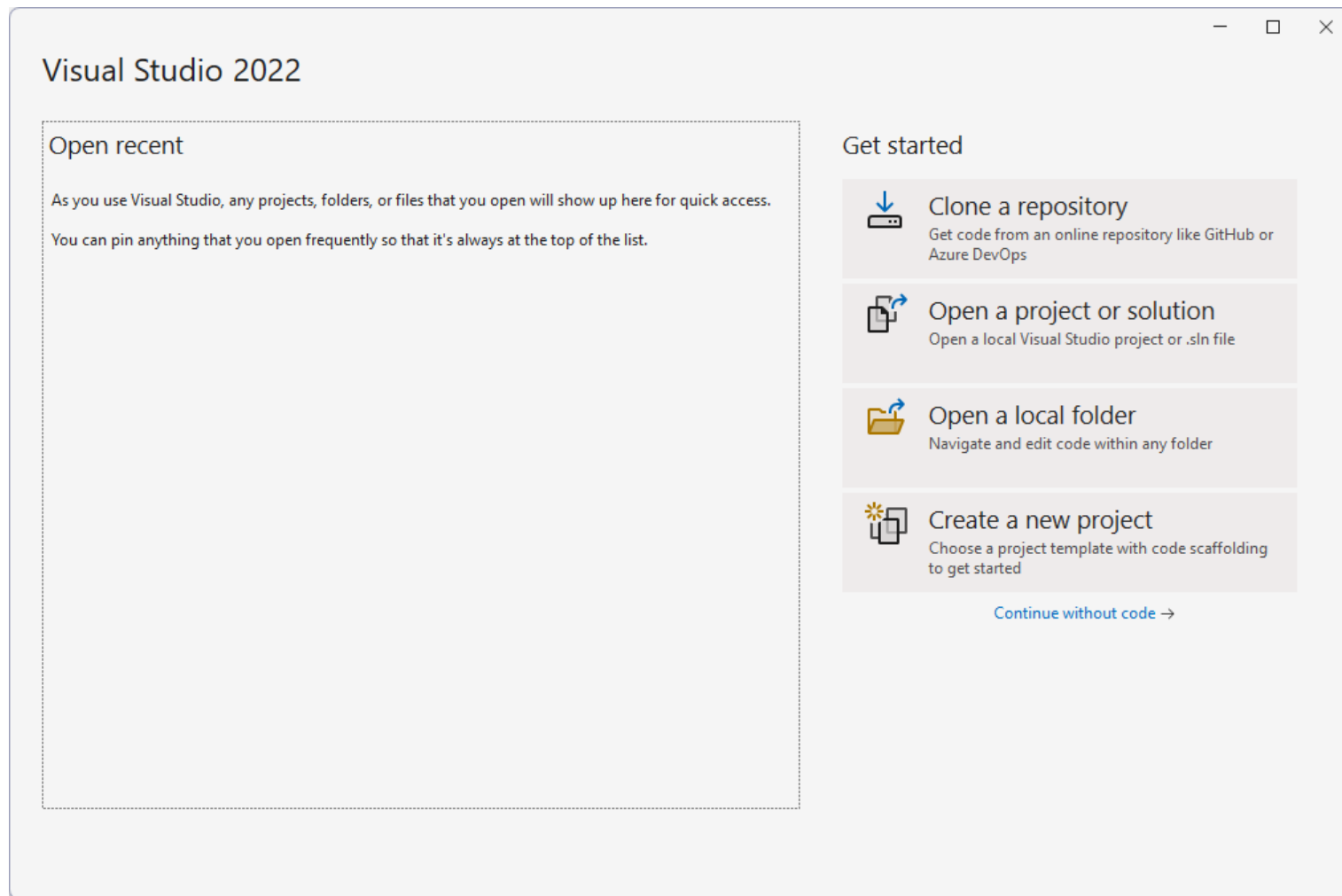
By continuing, you agree to the [license](#) for the product edition you selected. We also offer the ability to download other software. This software is licensed separately, as set out in the [3rd Party Notices](#) or in its accompanying license. By continuing, you also agree to those licenses.

Total space required 4.7 GB

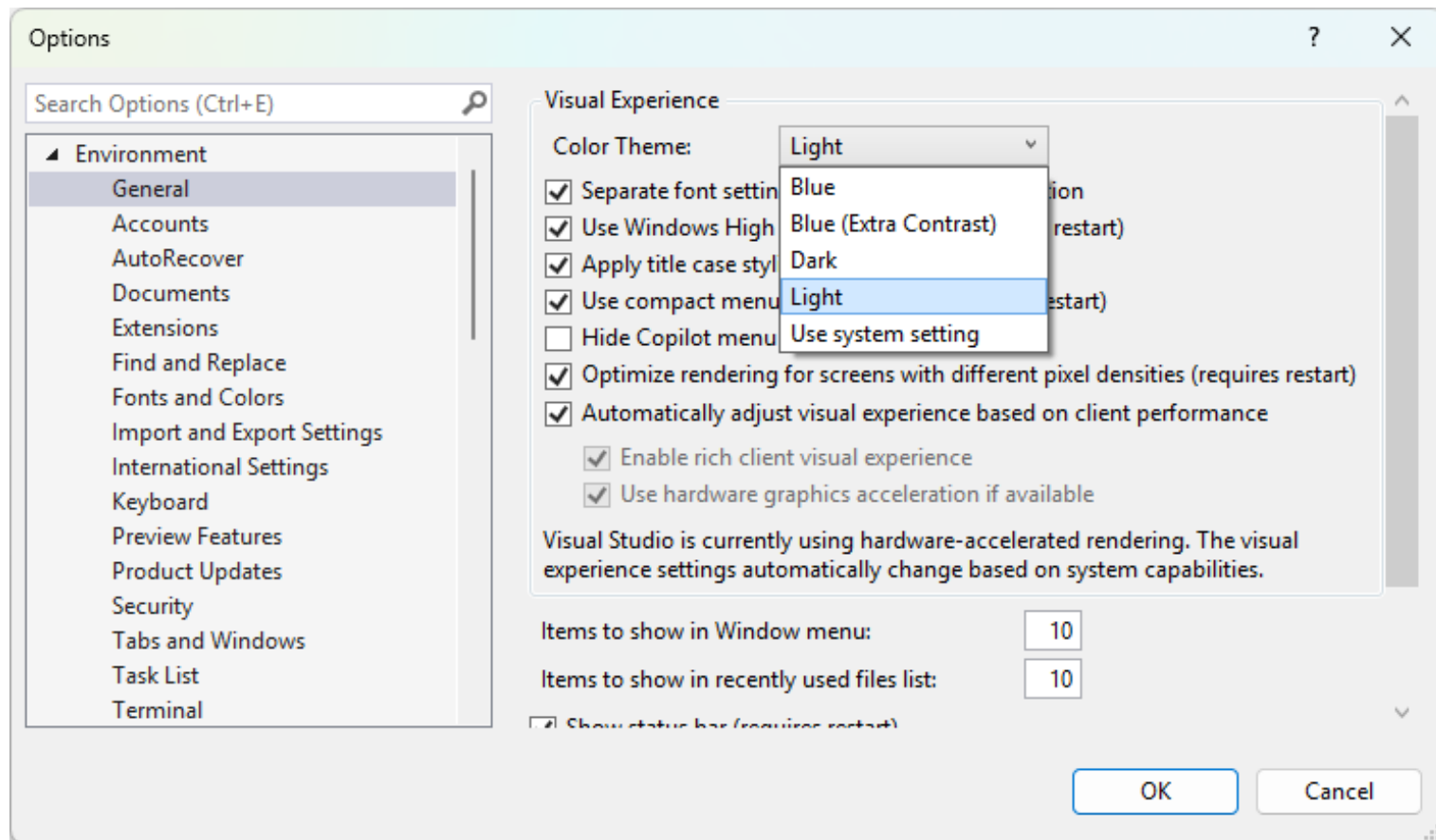
Install while downloading ▾

Install

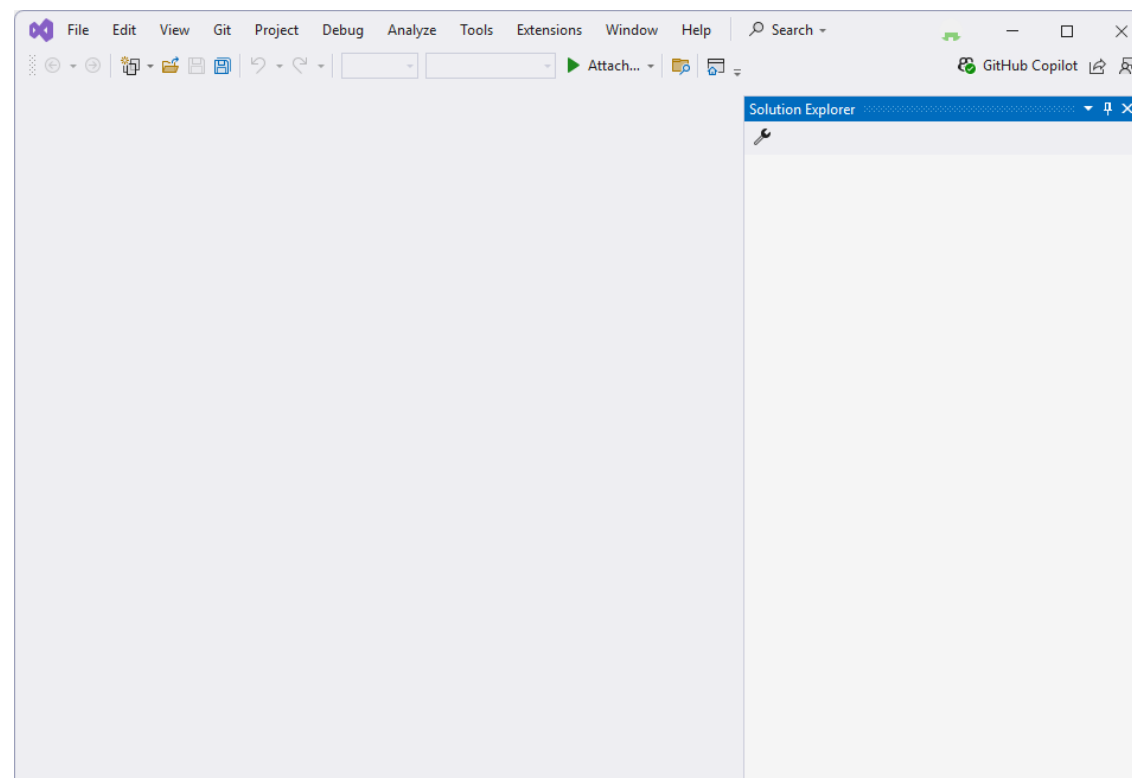
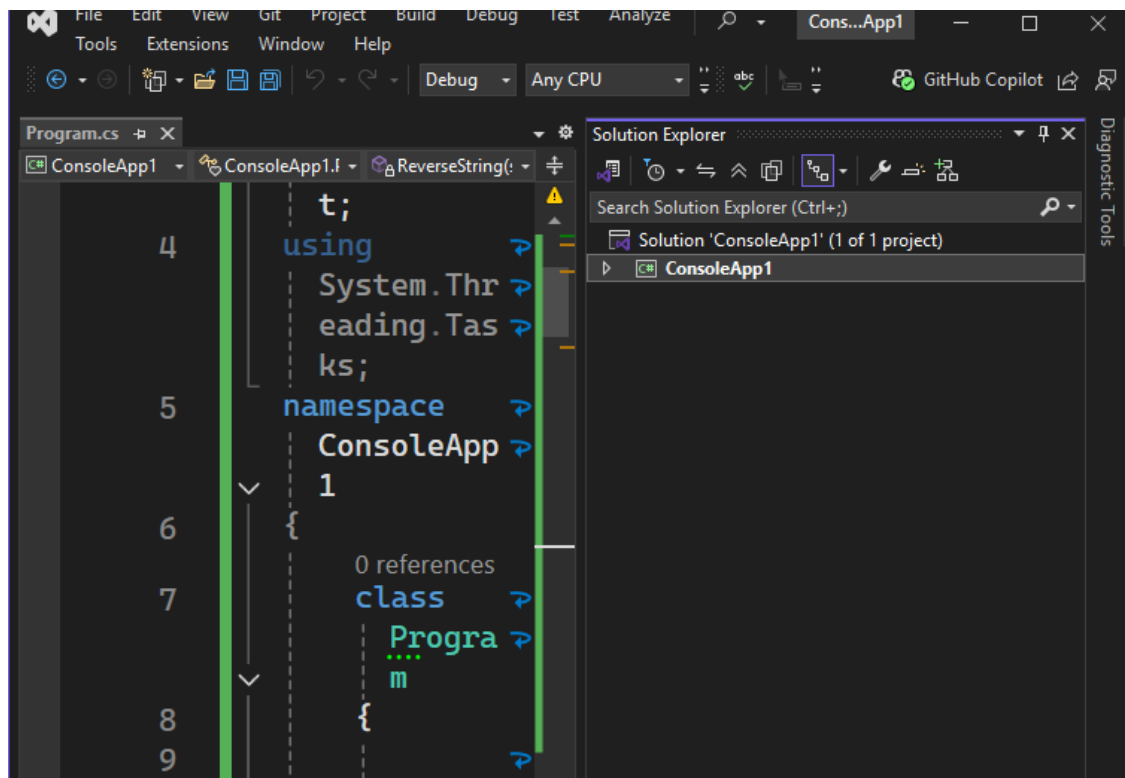
Màn hình giao diện phần mềm Visual Studio 2022



Thay đổi màu giao diện phần mềm



Dark - Light



Yêu cầu phần mềm cho môn học

Visual Studio 2022

.NET SDK: 9.0.301

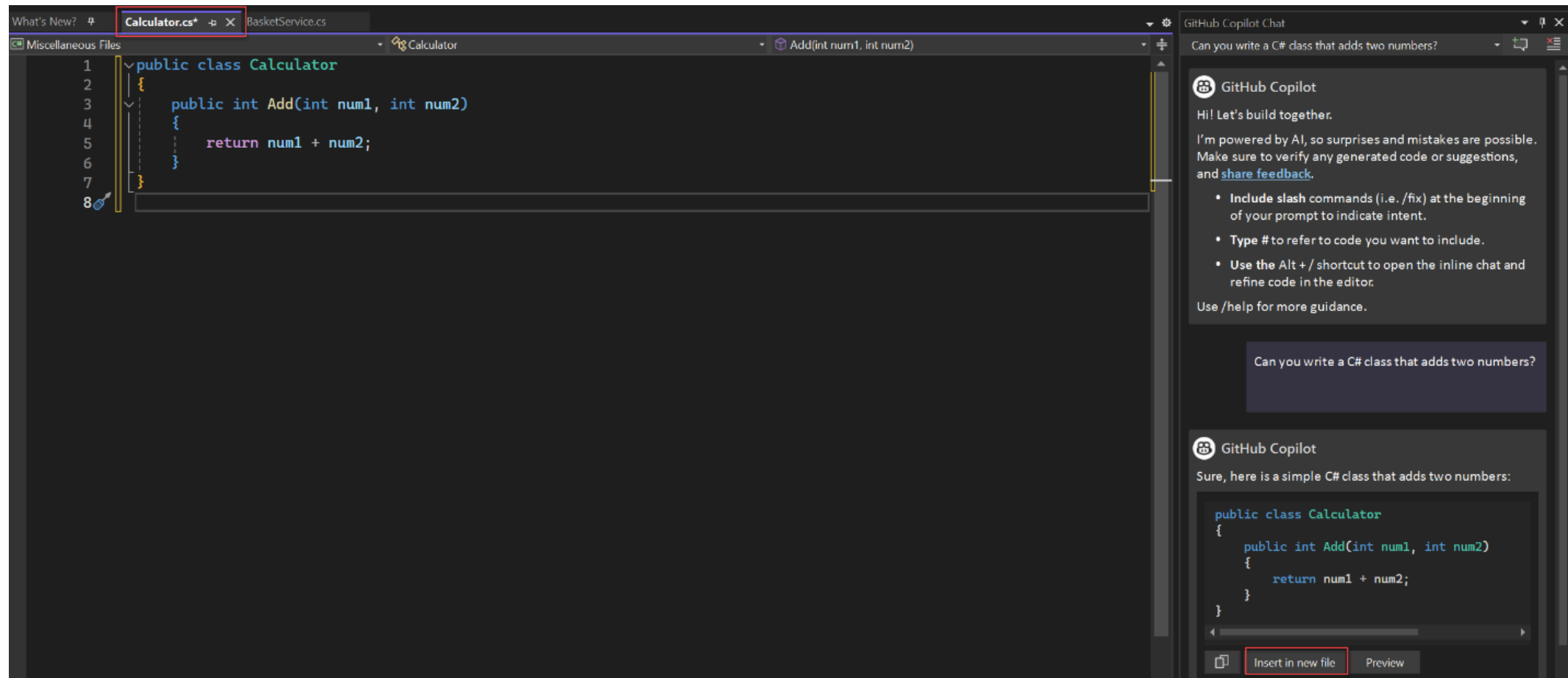
Github Copilot

Nâng cao hiệu suất và chất lượng công việc

- **Tăng Tốc Độ Viết Mã:** Hoàn thành nhanh chóng các đoạn mã lặp đi lặp lại hoặc các tác vụ phổ biến. Ví dụ: đọc file, kết nối CSDL,...
- **Tập Trung Vào Logic Phức Tạp:** Giúp bạn dành ít thời gian hơn cho cú pháp và nhiều thời gian hơn để giải quyết các bài toán kinh doanh cốt lõi.
- **Học Hỏi và Khám Phá:** Gợi ý các cách tiếp cận, các hàm hoặc thư viện mới mà bạn có thể chưa biết, giúp mở rộng kiến thức.
- **Giảm Thiểu Lỗi Cú Pháp:** Mã do Copilot tạo ra thường đúng về mặt cú pháp, giúp bạn tránh các lỗi cơ bản.

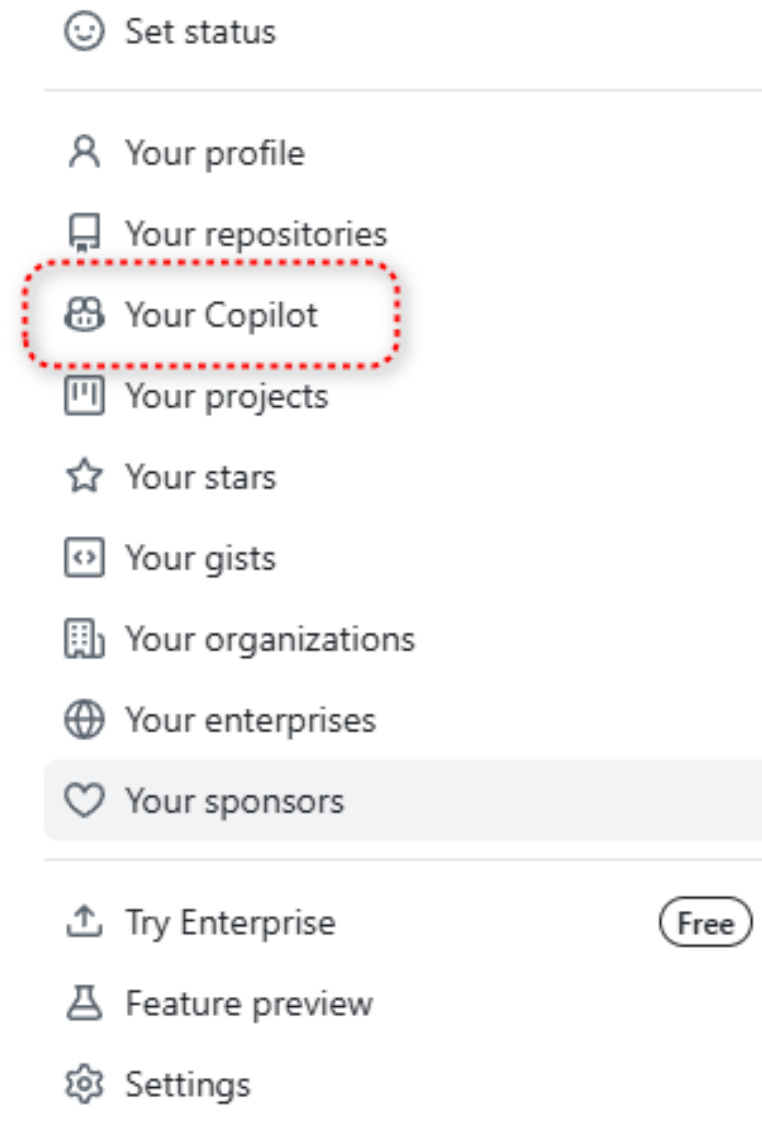
Copilot có thể làm gì cho bạn?

- **Gợi ý mã theo ngữ cảnh (Code Completion):** Dựa vào tên hàm hoặc biến bạn đang viết, Copilot sẽ đề xuất toàn bộ phần thân hàm.
- **Chuyển đổi Ngôn ngữ tự nhiên sang Mã (Natural Language to Code):**
 - Bạn viết một comment mô tả yêu cầu: // Hàm kiểm tra định dạng email
 - Copilot sẽ tự động tạo ra mã C# để thực hiện việc đó.
- **Viết Unit Test:** Giúp tạo các trường hợp kiểm thử (test cases) cho hàm của bạn, đảm bảo chất lượng mã nguồn.
- **Giải Thích Mã (Explain Code):** Bôi đen một đoạn mã phức tạp và yêu cầu Copilot giải thích cách nó hoạt động.
- **Hoàn thiện các đoạn mã phức tạp:** Dễ dàng tạo các lớp (class), các mẫu thiết kế (design patterns), hoặc các thuật toán phức tạp.



Kích hoạt Copilot

- Đăng nhập vào tài khoản GitHub của bạn.
- Truy cập vào **Settings** (Cài đặt) từ menu avatar của bạn.
- Trong menu bên trái, tìm và chọn mục **GitHub Copilot**.
- Làm theo hướng dẫn để **kích hoạt (enable)** dịch vụ. Bạn có thể cần đồng ý với các điều khoản và chính sách sử dụng.

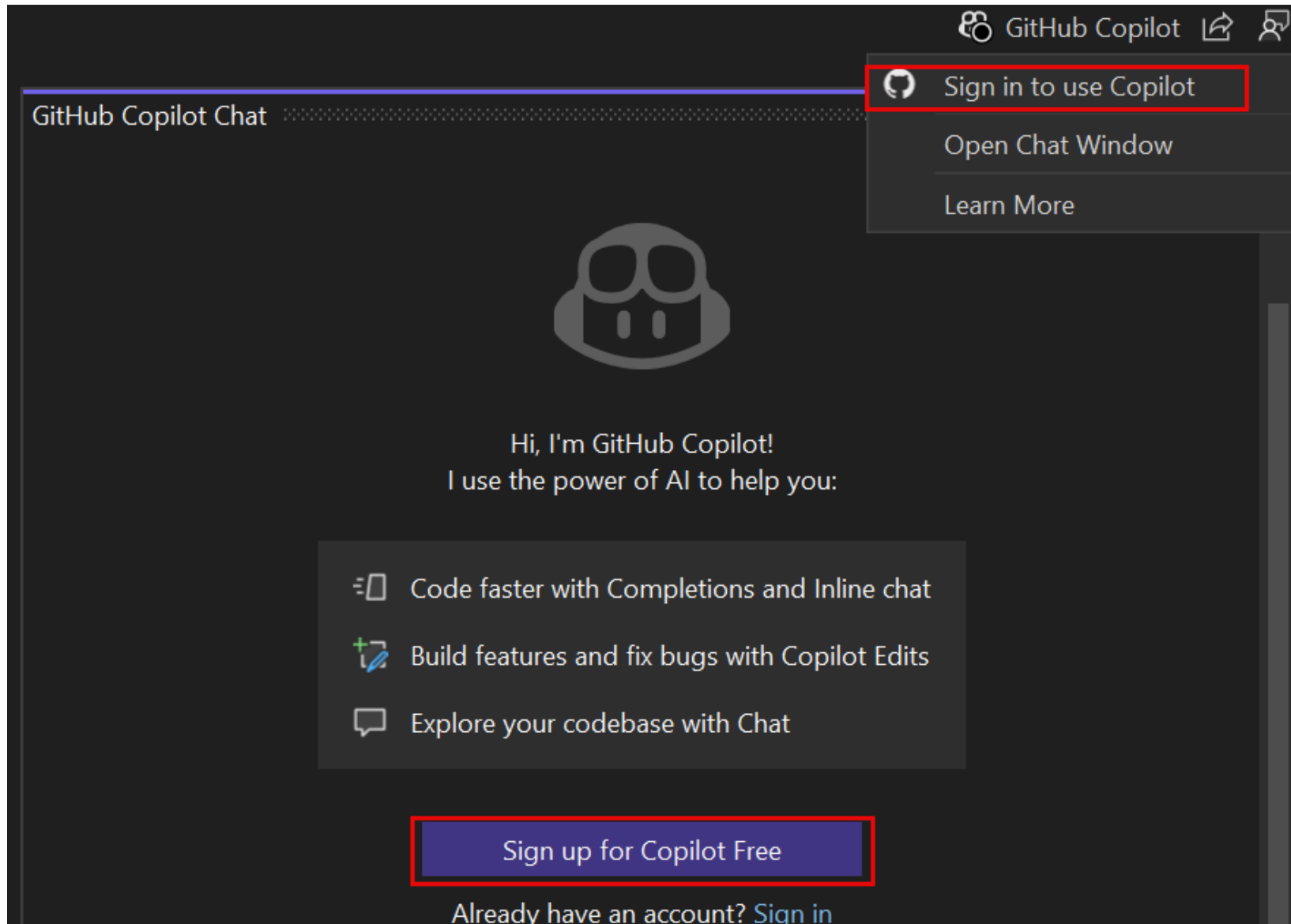


Cài đặt Github Copilot Vào Visual Studio 2022

- Các bước cài đặt Github Copilot Extensions trong Visual Studio:
- Mở **Visual Studio 2022**.
- Trên thanh menu, chọn **Extensions > Manage Extensions**.
- Trong cửa sổ Manage Extensions vừa mở:
 - Chọn mục **Online** ở thanh bên trái.
 - Sử dụng ô tìm kiếm ở góc trên bên phải, gõ: **GitHub Copilot**.
- Tìm extension có tên **GitHub Copilot** (thường là kết quả đầu tiên) và nhấn nút **Download**.

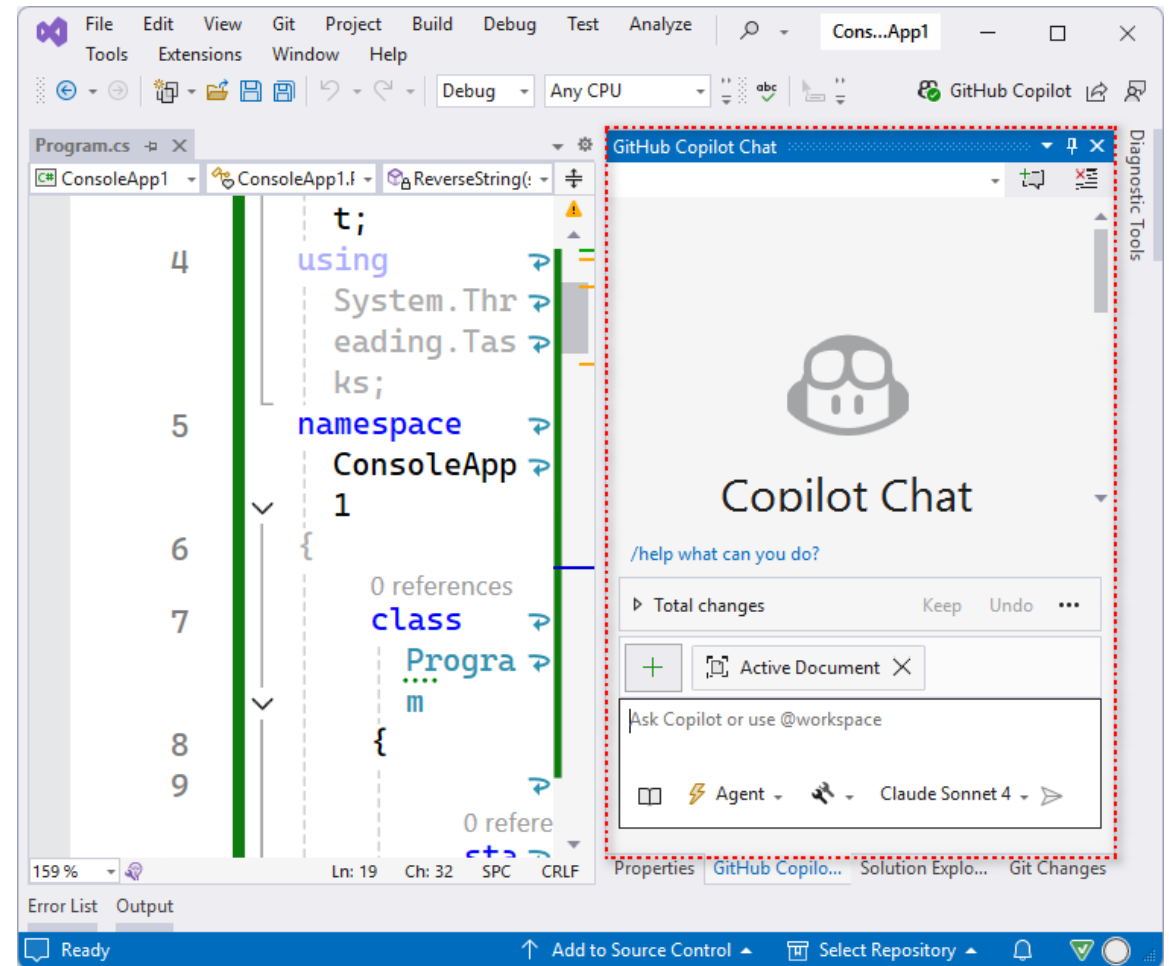
<https://learn.microsoft.com/en-us/visualstudio/ide/visual-studio-github-copilot-install-and-states?view=vs-2022>

<https://docs.github.com/en/education/about-github-education/github-education-for-students/apply-to-github-education-as-a-student>



Kết nối Visual Studio với tài khoản GitHub

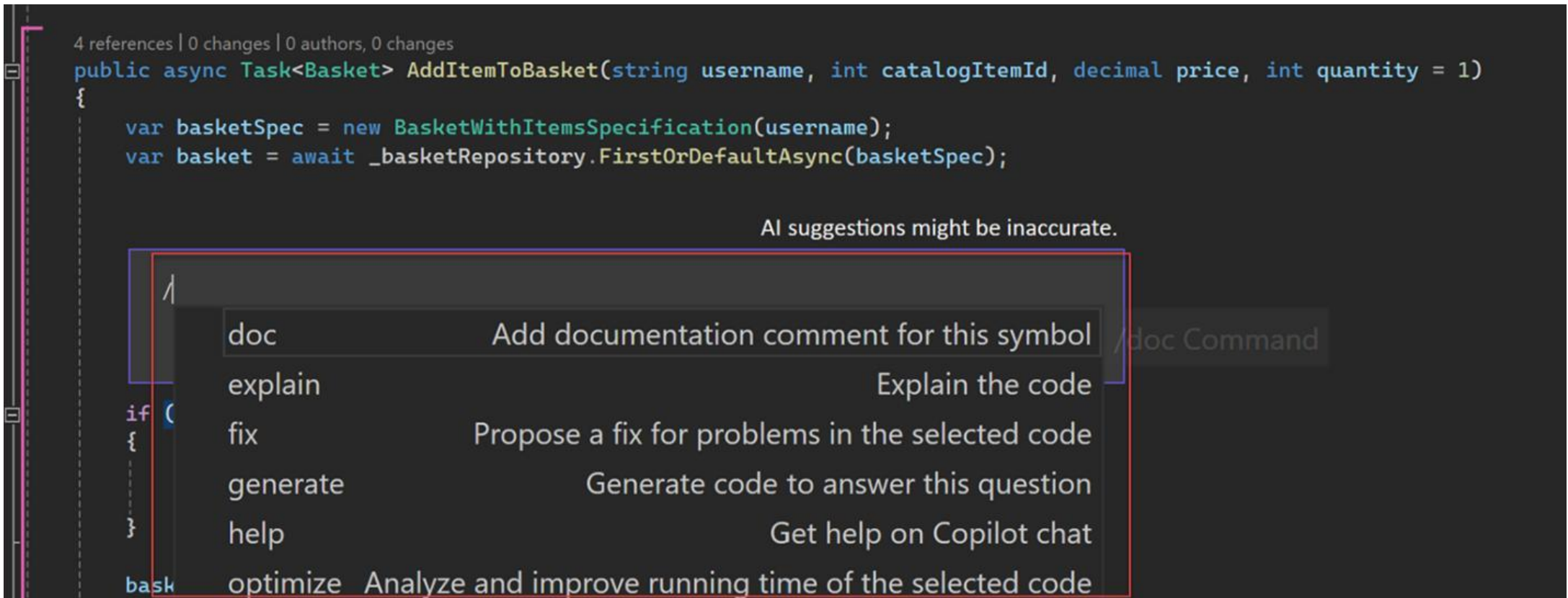
- **Mở lại Visual Studio 2022.**
- Một thông báo pop-up sẽ hiện ra yêu cầu bạn ủy quyền cho GitHub Copilot. Nhấn **OK**.
- Trình duyệt web của bạn sẽ mở ra một trang của GitHub, hiển thị một **mã xác thực (device code)**.
- Visual Studio cũng sẽ hiển thị một cửa sổ để bạn dán mã này vào.
- Sao chép mã từ trình duyệt, dán vào Visual Studio và làm theo các bước để **Authorize (Ủy quyền)**.



Các chức năng của Copilot

Command	Usage	Chat window	Inline chat
/doc	Add comments for specified or selected code. Examples: - <code>/doc DeleteBasketAsync method in BasketService.cs</code> - select desired code and enter <code>/doc</code>	Yes	Yes
/explain	Get code explanations. Examples: - <code>/explain the AddItemToBasket method in BasketService.cs</code> - select desired code and enter <code>/explain</code>	Yes	Yes
/fix	Propose a fix for problems in the selected code. Examples: - <code>/fix the SetQuantities method in BasketService.cs</code> - select desired code and enter <code>/fix</code>	Yes	Yes
/generate	Generate code to answer specified question. Example: <code>/generate code to add two numbers in Calculator.cs</code>	Yes	Yes
/help	Get help on using Copilot Chat. Example: <code>/help</code>	Yes	Yes
/optimize	Analyze and improve running time of the selected code. Examples: - <code>/optimize the AddItemToBasket method in BasketService.cs</code> - select desired code and enter <code>/optimize</code>	Yes	Yes
/tests	Create unit tests for the selected code. Example: select desired code and enter <code>/tests using XUnit Framework</code>	Yes	Yes

Sử dụng Copilot trong VS 2022



Sinh mã nguồn

```
Ask Copilot or use @workspace Claude Sonnet 4 ▶ | ✕

// Hàm để đảo ngược một chuỗi
static string ReverseString(string input)
{
    if (string.IsNullOrEmpty(input))
        return input;

    char[] charArray = input.ToCharArray();
    Array.Reverse(charArray);
    return new string(charArray);
}
}
```

Tối ưu mã nguồn

1 reference | Anh Nguyen, Less than 5 minutes ago | 1 author, 1 change

```
static int[] FindIntersection(int[] array1, int[] a
{
    var intersection = new List<int>();

    for (int i = 0; i < array1.Length; i++)
    {
        for (int j = 0; j < array2.Length; j++)
        {
            if (array1[i] == array2[j])
            {
                intersection.Add(array1[i]);
            }
        }
    }

    return intersection.ToArray();
}
```

File Edit View Git Project Build Debug Test Analyze Tools Extensions Window Help Search Github_Copilot

Program.cs

0 references

```
static void Main()
{
    // Create two lists of arrays not duplicating elements
    List<int> list1 = [3, 1, 4, 5, 9, 2, 6];
    List<int> list2 = [2, 7, 1, 8, 0, 21];

    // Sort both lists
    list1.Sort();
    list2.Sort();

    Console.WriteLine($"Sorted List 1: [{string.Join(", ", list1)}]");
    Console.WriteLine($"Sorted List 2: [{string.Join(", ", list2)}]");

    // Find intersection
    var intersection: int[] = FindIntersection(list1.ToArray(), list2.ToArray());

    Console.WriteLine($"Intersection: [{string.Join(", ", intersection)}]");
    Console.ReadLine();
}

1 reference
static int[] FindIntersection(int[] array1, int[] array2)
{
    var intersection = new List<int>();

    for (int i = 0; i < array1.Length; i++)
    {
        for (int j = 0; j < array2.Length; j++)
        {
            if (array1[i] == array2[j])
            {
                intersection.Add(array1[i]);
            }
        }
    }

    return intersection.ToArray();
}
```

GitHub Copilot Chat

Optimize the FindIntersection method

1. If first element is smaller, advance first pointer
2. If second element is smaller, advance second pointer
3. Continue until one array is exhausted
4. If first element is smaller, advance first pointer
5. If second element is smaller, advance second pointer
6. Continue until one array is exhausted

ResponseType: Code File: Program.cs Content:

Active Document

```
static int[] FindIntersection(int[] array1, int[] array2)
{
    var intersection = new List<int>();
    int i = 0, j = 0;

    while (i < array1.Length && j < array2.Length)
    {
        if (array1[i] == array2[j])
        {
            intersection.Add(array1[i]);
            i++;
            j++;
        }
        else if (array1[i] < array2[j])
        {
            i++;
        }
        else
        {
            j++;
        }
    }

    return intersection.ToArray();
}
```

Total changes

Active Document

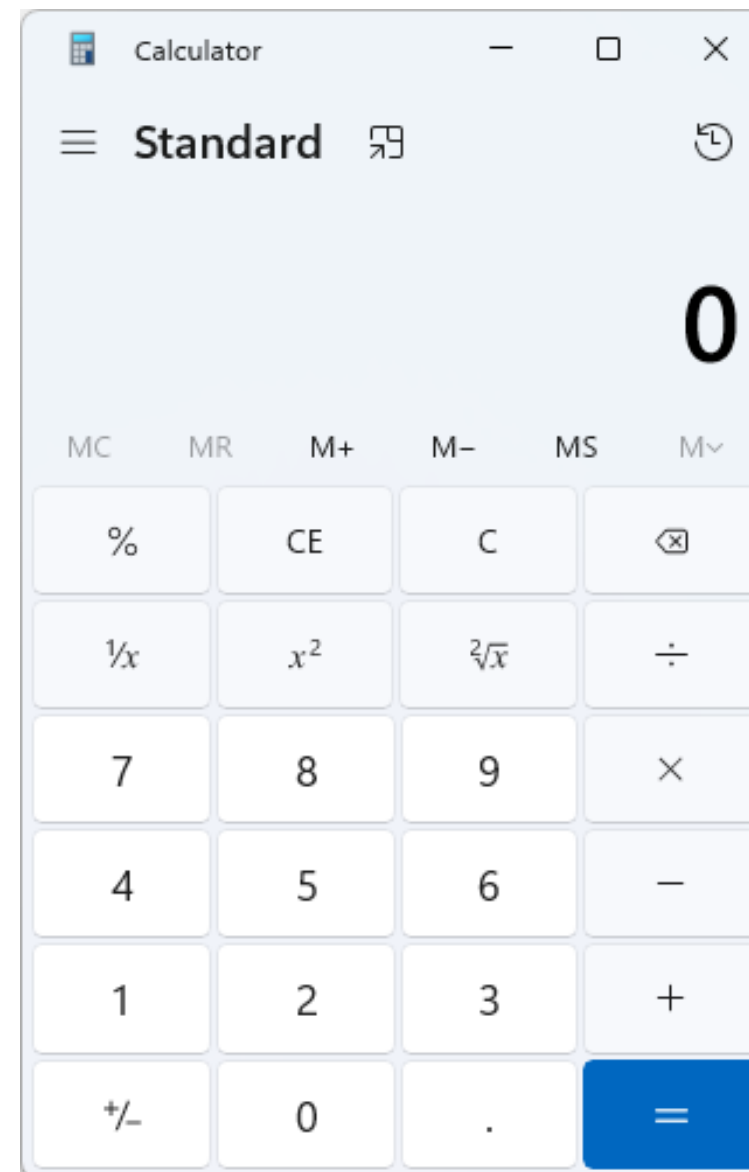
or use @workspace

Agent Claude Sonnet 4

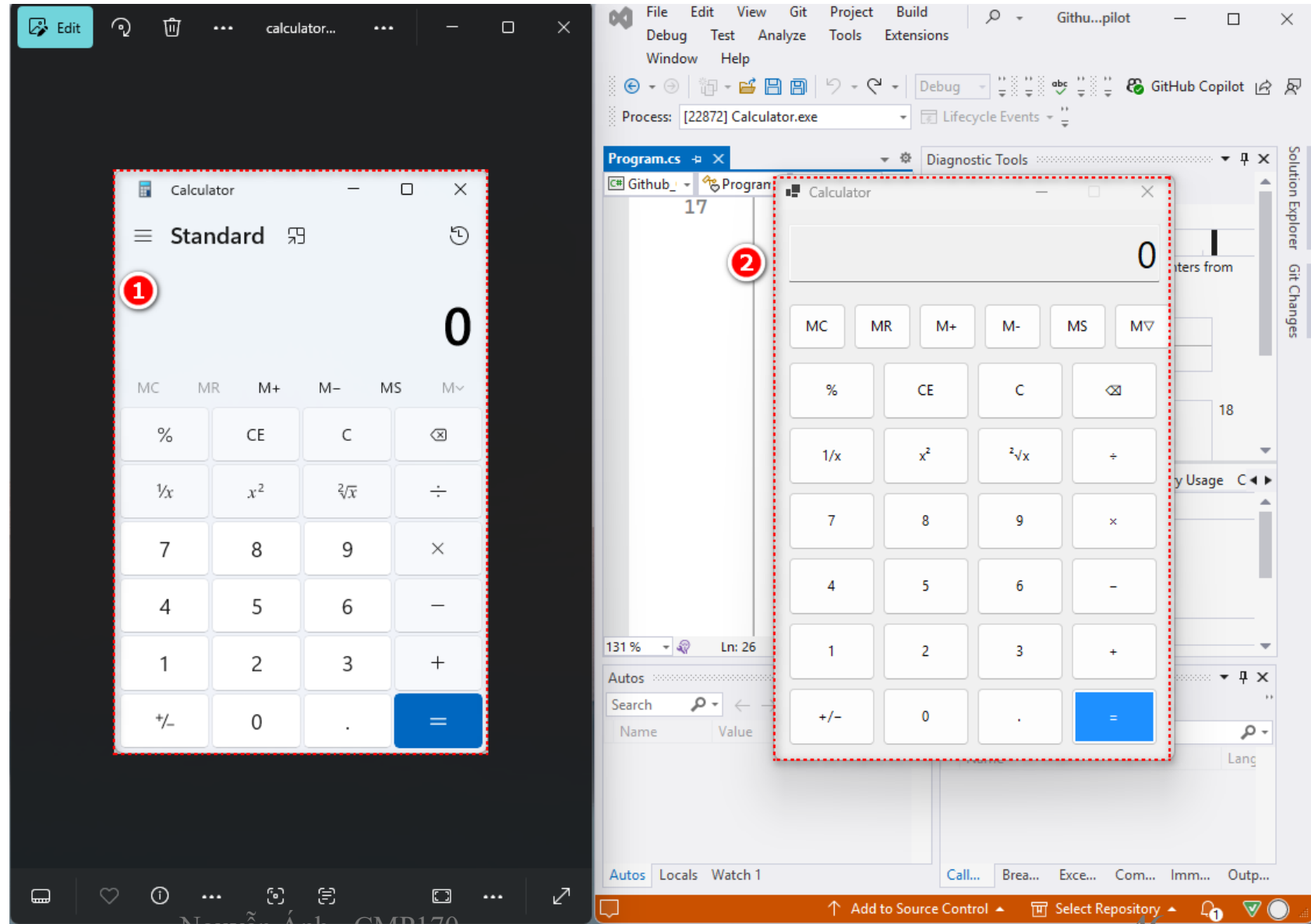
Properties GitHub Copilot Chat Solution Explorer Git Changes

Tối ưu mã nguồn

Tạo giao diện và chức năng từ hình ảnh



Tạo giao diện và chức năng từ hình ảnh

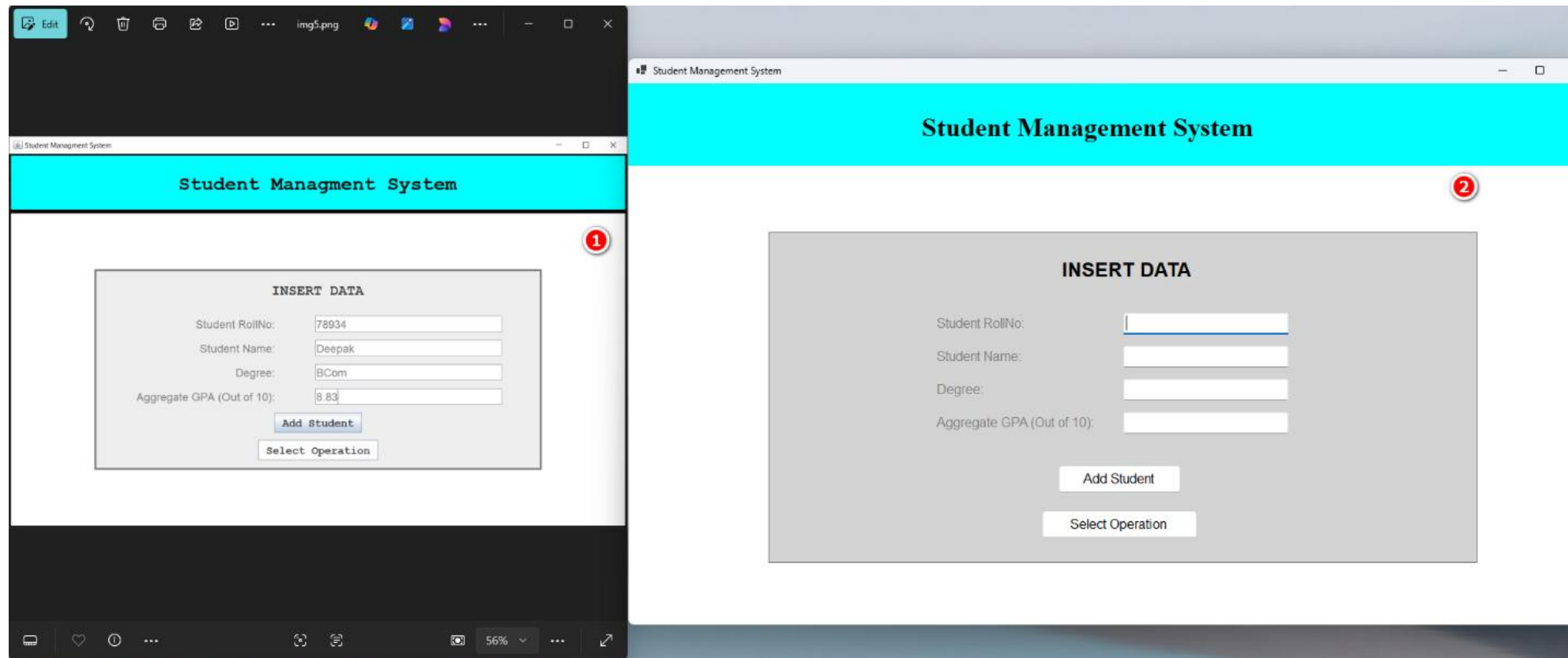


Tạo ứng dụng từ hình ảnh

The screenshot displays a web browser window titled "Student Managment System". The page has a bright cyan header bar with the text "Student Managment System" in a black, monospaced font. Below the header, the main content area is white and contains a light gray rectangular box. Inside this box, the title "INSERT DATA" is centered at the top. Below the title, there are four input fields arranged vertically, each with a label to its left: "Student RollNo:" with the value "78934", "Student Name:" with the value "Deepak", "Degree:" with the value "BCom", and "Aggregate GPA (Out of 10):" with the value "8.83". Below these fields are two buttons: a blue button labeled "Add Student" and a white button labeled "Select Operation".

INSERT DATA	
Student RollNo:	<input type="text" value="78934"/>
Student Name:	<input type="text" value="Deepak"/>
Degree:	<input type="text" value="BCom"/>
Aggregate GPA (Out of 10):	<input type="text" value="8.83"/>
<input type="button" value="Add Student"/>	
<input type="button" value="Select Operation"/>	

Tạo ứng dụng từ hình ảnh



Hướng dẫn nộp bài tập

Tạo Fork từ Repository mẫu theo đường dẫn:

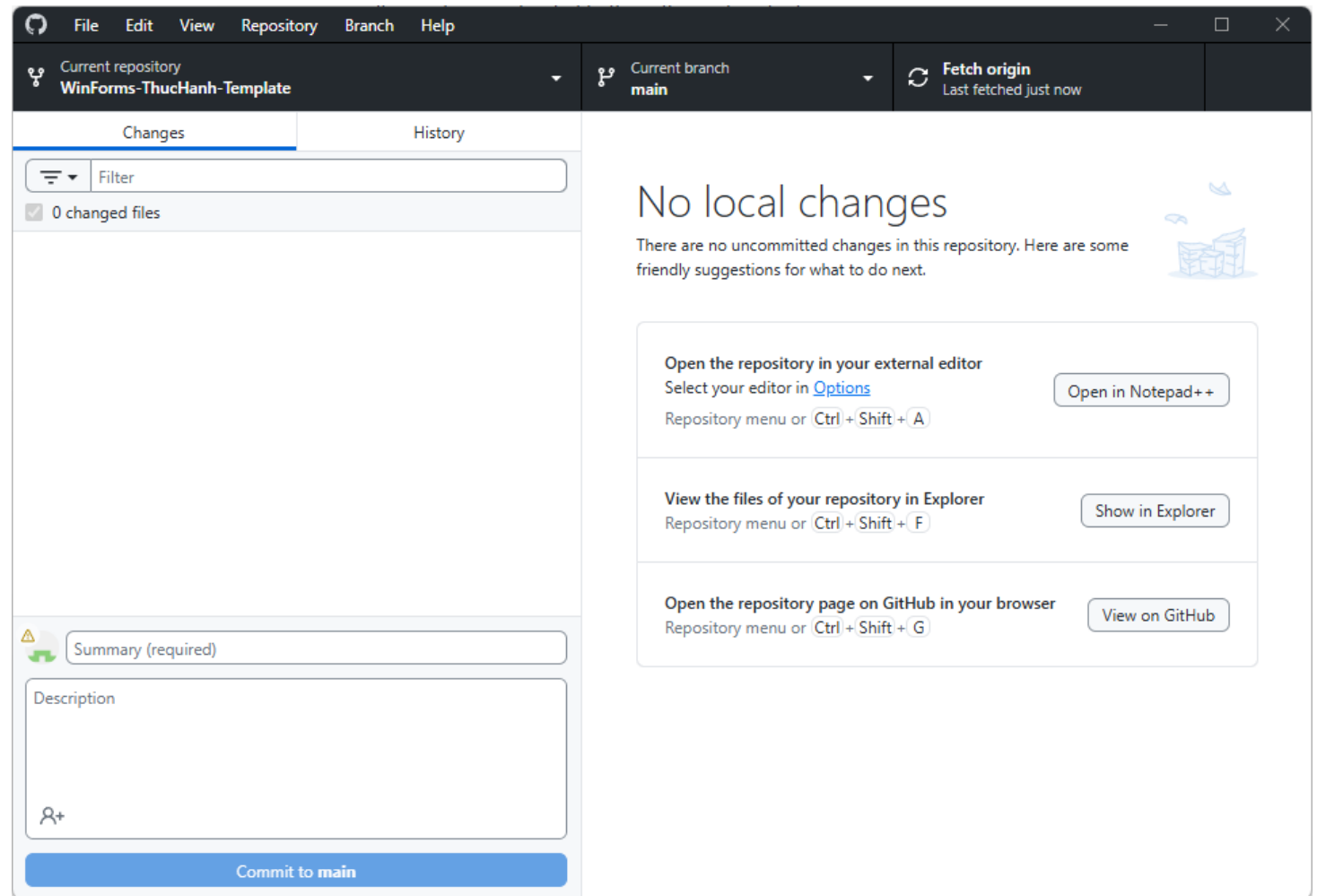
- <https://github.com/nguyenanhvn9/WinForms-Template>



Hướng dẫn nộp bài tập

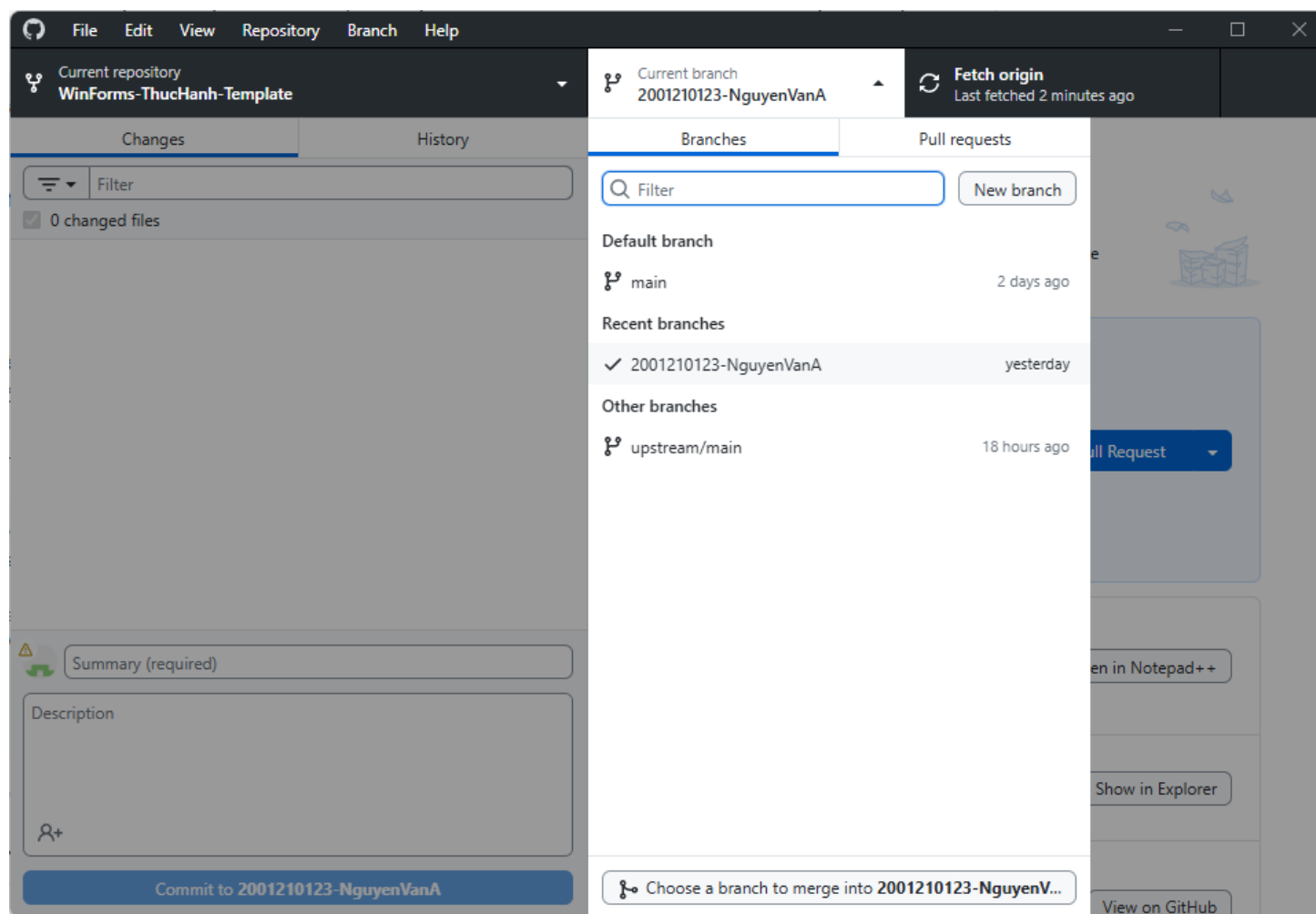
Clone Repository cá nhân về máy tính

1. Truy cập trang repository bạn vừa fork (repository cá nhân).
2. Nhấn vào nút màu xanh lá < > Code và sao chép đường link HTTPS.



Hướng dẫn nộp bài tập

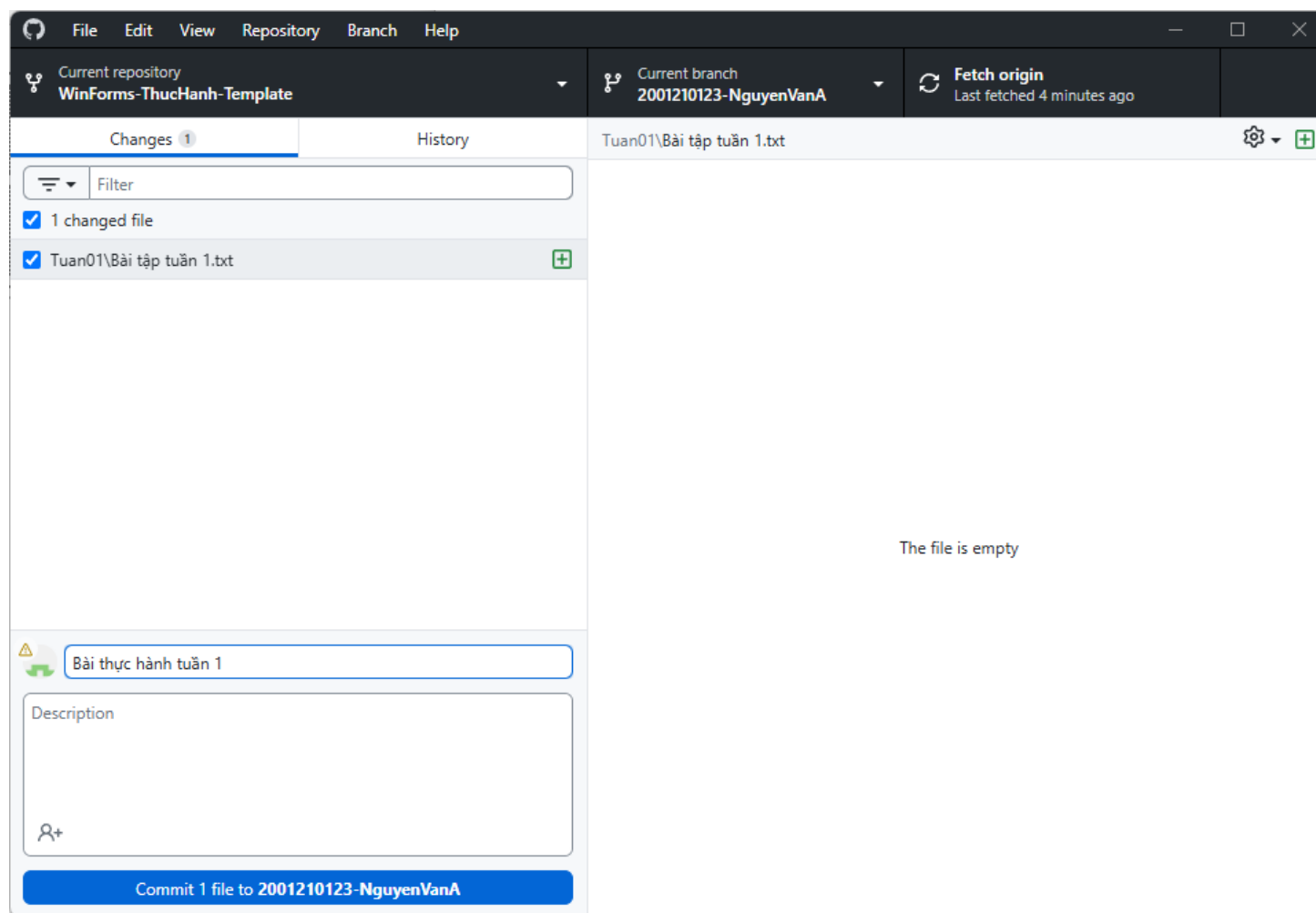
- **Tạo Branch (Nhánh) làm việc cá nhân**
- Tạo branch mới với tên theo định dạng: **[MSSV]-[HoTen]**



Hướng dẫn nộp bài tập

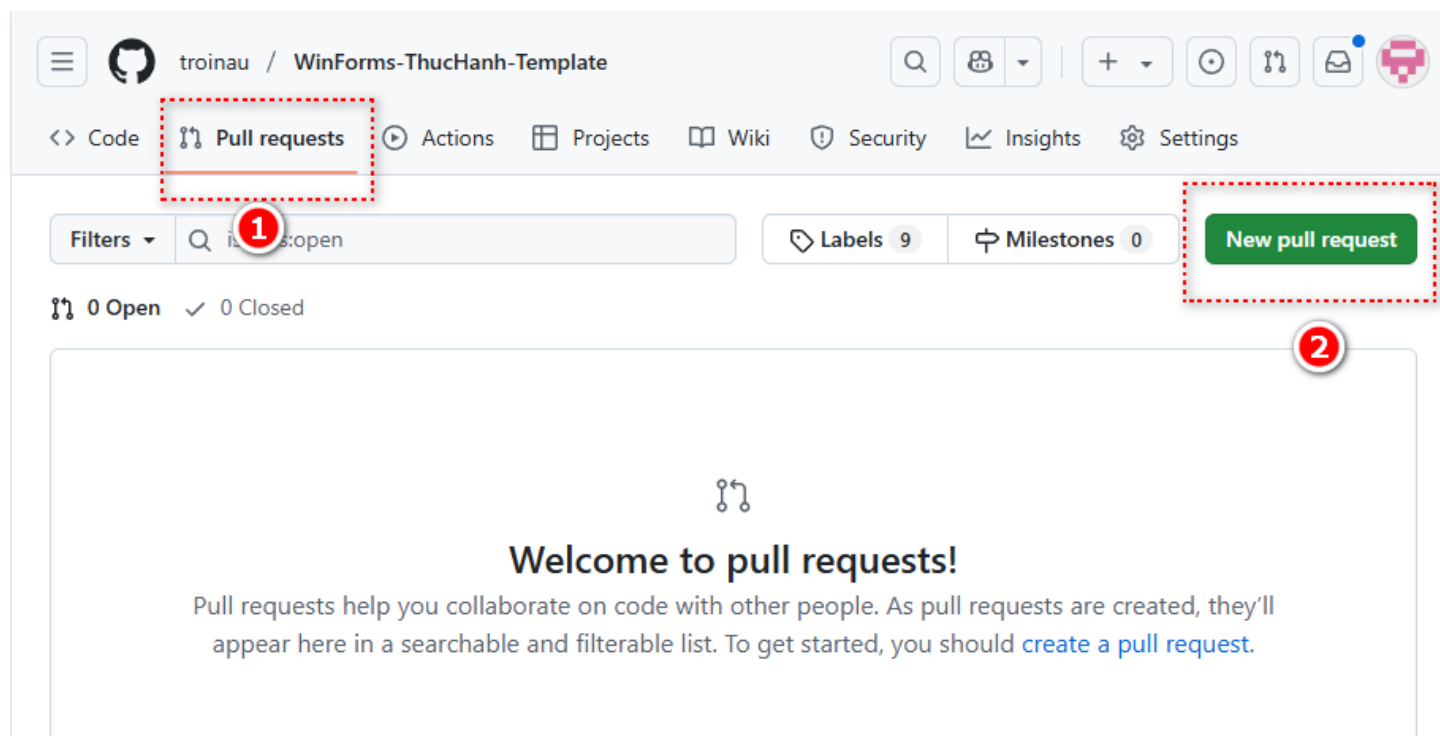
Commit kết quả

- Commit kết quả bài lên
lên Repository cá nhân



Hướng dẫn nộp bài tập

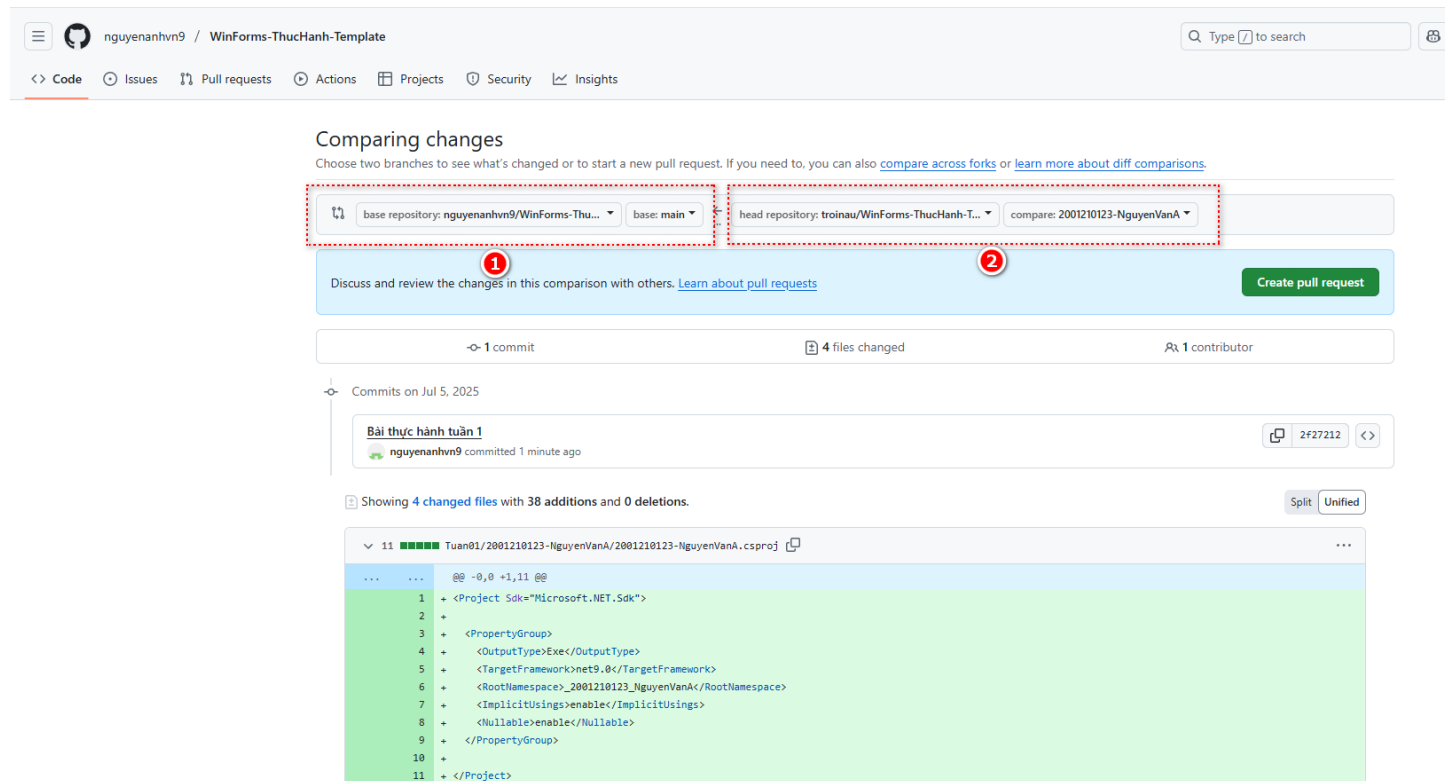
- Tạo một Pull Request



Hướng dẫn nộp bài tập

Chọn thông tin cho Pull Request

- Chọn chính xác base repository và head repository.

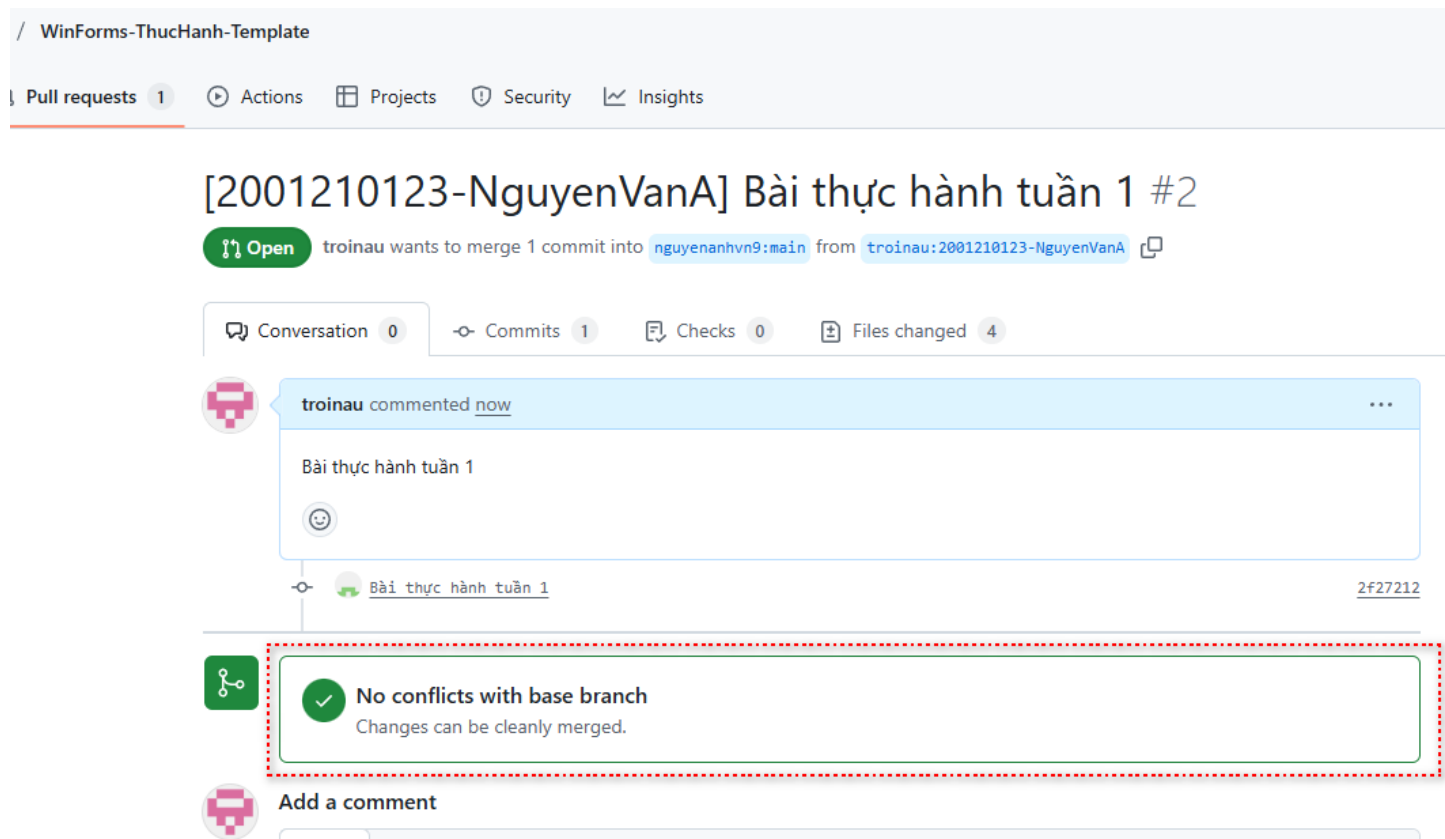


Đặt tên tiêu đề theo định dạng [Mã sinh viên-Họ và tên] Bài thực hành tuần X (X là thứ tự của tuần)

Remember, contributions to this repository should follow our [GitHub Community Guidelines](#).

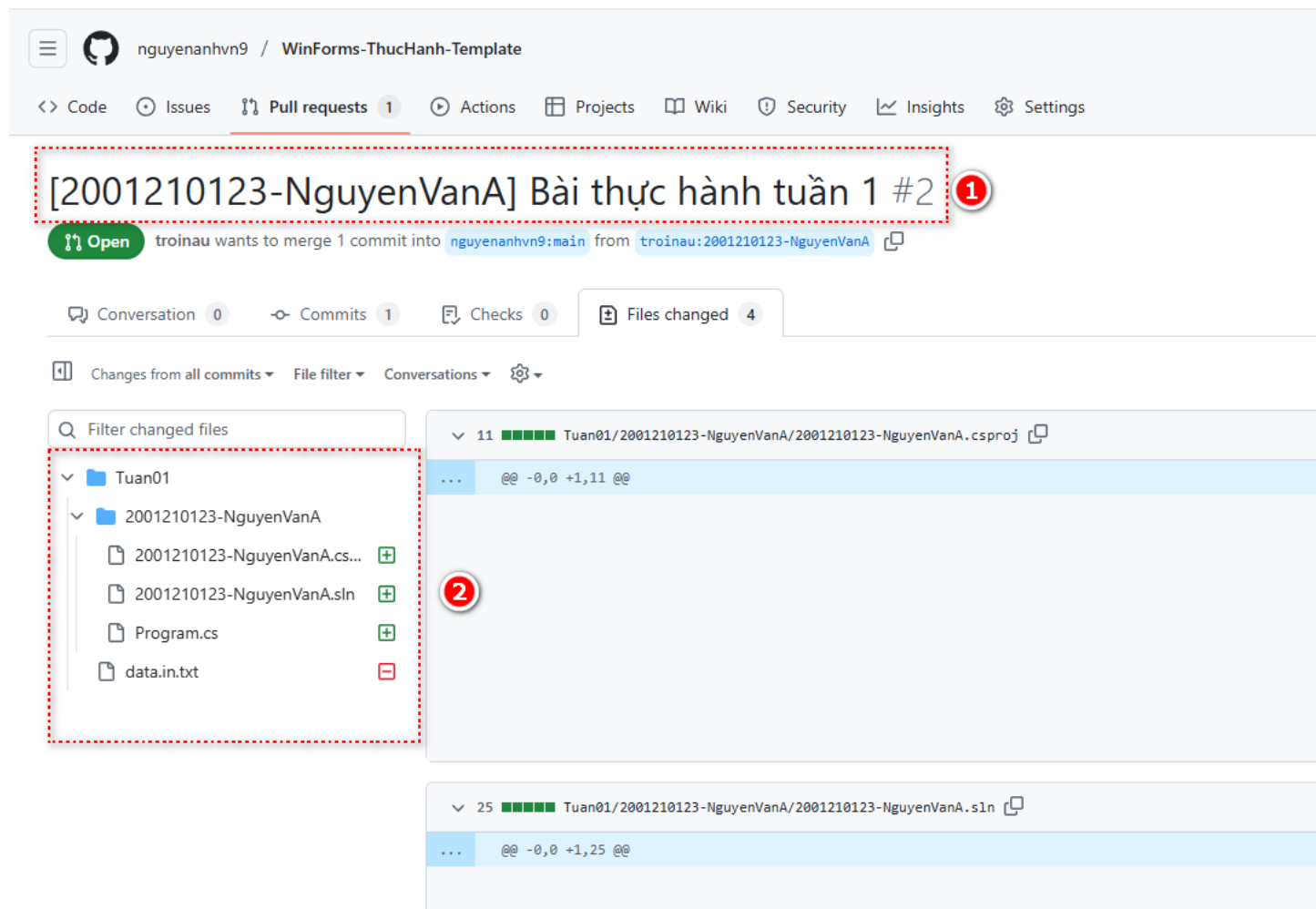
Hướng dẫn nộp bài tập

- Kết quả sau khi đã tạo Pull Request



Hướng dẫn nộp bài tập

- Kết quả ở trên Github Repository của giảng viên



Các Kiểu Dữ Liệu (Data Types)

- Trong C#, các kiểu dữ liệu được chia thành hai loại chính dựa trên cách chúng lưu trữ giá trị trong bộ nhớ:
- **Kiểu Tham trị (Value Types):**
 - Chứa trực tiếp dữ liệu của nó.
 - Được lưu trữ trên vùng nhớ **Stack**.
 - Khi gán, giá trị sẽ được **sao chép**.
 - Ví dụ: int, double, bool, char, struct, enum.
- **Kiểu Tham chiếu (Reference Types):**
 - Chứa một tham chiếu (địa chỉ) đến vùng nhớ chứa dữ liệu thực tế.
 - Dữ liệu được lưu trữ trên vùng nhớ **Heap**.
 - Khi gán, chỉ có **tham chiếu được sao chép**, cả hai biến cùng trỏ đến một đối tượng.
 - Ví dụ: string, object, class, interface, delegate, array.

Các Kiểu Dữ Liệu Thông Dụng

Kiểu dữ liệu	Mô tả	Kích thước	Ví dụ khai báo
int	Số nguyên	4 bytes	int soLuong = 100;
double	Số thực dấu phẩy động chính xác kép	8 bytes	double diemTB = 8.5;
decimal	Số thập phân chính xác cao (dùng cho tài chính)	16 bytes	decimal luong = 1500.99m;
bool	Logic (Đúng/Sai)	1 byte	bool daKichHoat = true;
char	Một ký tự Unicode	2 bytes	char kyTu = 'A';
string	Chuỗi ký tự (tham chiếu)	Thay đổi	string hoTen = "Nguyễn Văn A";

Cấu Trúc Một Chương Trình C# Console



```
// 1. Khai báo không gian tên được sử dụng
using System;

// 2. Định nghĩa không gian tên cho dự án
namespace HelloWorldApp
{
    // 3. Định nghĩa lớp (class) chính
    class Program
    {
        // 4. Phương thức Main - Điểm khởi chạy của
        static void Main(string[] args)
        {
            // 5. Câu lệnh thực thi
            Console.WriteLine("Hello, World!");
        }
    }
}
```

Phân tích chương trình “Hello, World!”

- **using:** Chỉ thị cho trình biên dịch biết chúng ta muốn sử dụng các lớp từ một không gian tên (namespace) nào đó. System là namespace chứa các lớp cơ bản như Console.
- **namespace:** "Họ" của các lớp, dùng để tổ chức và tránh trùng tên.
- **class:** "Bản thiết kế" cho một đối tượng, bao gồm các thuộc tính (dữ liệu) và phương thức (hành vi).
- **Main:** Phương thức đặc biệt, là **điểm vào (entry point)** của mọi ứng dụng C#. Chương trình sẽ bắt đầu thực thi từ đây.
- **Câu lệnh:** Các chỉ thị cụ thể mà chương trình sẽ thực hiện.

Tính Đóng Gói (Encapsulation)

- **Định nghĩa:** Là cơ chế ràng buộc dữ liệu (thuộc tính) và các phương thức thao tác với dữ liệu đó vào cùng một thể thống nhất (lớp), đồng thời che giấu thông tin bên trong khỏi sự truy cập trực tiếp từ bên ngoài.

Mục đích:

- **Bảo vệ dữ liệu:** Ngăn chặn việc thay đổi dữ liệu một cách không hợp lệ.
- **Tăng tính linh hoạt:** Có thể thay đổi logic bên trong mà không ảnh hưởng đến các lớp khác.
- **Giảm độ phức tạp:** Chỉ đưa ra những gì cần thiết.

Private Fields Và Public Properties

Tính đóng gói:

Private field (Trường dữ liệu riêng tư):

- Là biến thành viên của lớp, được khai báo với từ khóa private.
- Chỉ có thể được truy cập từ bên trong lớp đó.
- Dùng để lưu trữ trạng thái nội tại của đối tượng.
- **Quy ước đặt tên:** Bắt đầu bằng dấu gạch dưới và dùng kiểu `_camelCase`.

Public property (Thuộc tính công khai):

- Là một "cổng giao tiếp" công khai, cho phép bên ngoài đọc (get) hoặc ghi (set) giá trị cho private field một cách có kiểm soát.
- **get accessor:** Trả về giá trị của private field.
- **set accessor:** Gán giá trị mới cho private field. Từ khóa `value` đại diện cho giá trị được gán vào.

Ví Dụ: Lớp Sinhvien

```
public class SinhVien
{
    // 1. Private field
    private string _maSoSinhVien;
    private double _diemTrungBinh;

    // 2. Public property cho maSoSinhVien (chỉ cho phép đọc)
    public string MaSoSinhVien
    {
        get { return _maSoSinhVien; }
        // Không có 'set' -> thuộc tính này là read-only
    }

    // 3. Public property cho diemTrungBinh (có kiểm soát logic)
    public double DiemTrungBinh
    {
        get { return _diemTrungBinh; }
        set
        {
            // Logic kiểm tra: điểm phải nằm trong khoảng từ 0 đến 10
            if (value >= 0 && value <= 10)
            {
                _diemTrungBinh = value;
            }
            else
            {
                // Có thể ném ra một ngoại lệ hoặc gán giá trị mặc định
                throw new ArgumentOutOfRangeException("Điểm trung bình
không hợp lệ.");
            }
        }
    }

    // 4. Auto-Implemented Property (cú pháp rút gọn)
    // Trình biên dịch sẽ tự động tạo một private field ẩn
    public string HoTen { get; set; }

    // Constructor
    public SinhVien(string mssv)
    {
        _maSoSinhVien = mssv;
    }
}
```

Các Cấu Trúc Dữ Liệu (Collections)

Collection	Đặc điểm	Trường hợp sử dụng
List<T>	Danh sách có thứ tự, cho phép phần tử trùng lặp. Truy cập bằng chỉ số (index).	Cần một danh sách động, truy cập thường xuyên theo vị trí. Đây là collection phổ biến nhất.
Dictionary<TKey, TValue>	Tập hợp các cặp Key-Value. Key phải là duy nhất. Truy cập rất nhanh thông qua Key.	Lưu trữ dữ liệu cần tra cứu nhanh. Ví dụ: danh bạ (Key: SĐT, Value: Tên).
HashSet<T>	Tập hợp không có thứ tự, không cho phép phần tử trùng lặp. Tối ưu cho việc kiểm tra sự tồn tại của một phần tử.	Cần đảm bảo các phần tử là duy nhất.
Array	Mảng có kích thước cố định, có thứ tự. Hiệu năng cao.	Khi biết trước chính xác số lượng phần tử và không thay đổi.

Ví Dụ Về Collections



```
// 1. List<T> - Quản lý danh sách sinh viên
List<SinhVien> danhSachLop = new List<SinhVien>();
danhSachLop.Add(new SinhVien("SV001"));
danhSachLop.Add(new SinhVien("SV002"));
Console.WriteLine($"Số lớp: {danhSachLop.Count}");

// 2. Dictionary<TKey, TValue> - Tra cứu sinh viên theo MSSV
Dictionary<string, SinhVien> tuDienSV = new Dictionary<string, SinhVien>();
tuDienSV.Add("SV001", danhSachLop[0]);
tuDienSV.Add("SV002", danhSachLop[1]);

SinhVien sv = tuDienSV["SV001"]; // Truy cập rất nhanh

// 3. HashSet<T> - Lưu danh sách các mã môn học duy nhất
HashSet<string> maMonHoc = new HashSet<string>();
maMonHoc.Add("CS101");
maMonHoc.Add("MA102");
maMonHoc.Add("CS101"); // Thao tác này sẽ bị bỏ qua vì đã tồn tại
Console.WriteLine($"Số môn học: {maMonHoc.Count}"); // Kết quả là 2
```

Nhập/Xuất Dữ Liệu (Input/Output)

- **Xuất:** `Console.WriteLine()` (in ra màn hình và xuống dòng), `Console.Write()` (in và không xuống dòng).
- **Nhập:** `Console.ReadLine()` đọc một dòng văn bản từ người dùng (luôn trả về kiểu string).

Ép kiểu (Parsing)

Vì ReadLine() trả về string, cần chuyển đổi sang kiểu số để tính toán



```
Console.Write("Nhập tuổi của bạn: ");  
string input = Console.ReadLine();  
int tuoi = int.Parse(input); // Chuyển string sang int
```

Xử lý Ngoại lệ (Exception Handling)

Điều gì xảy ra nếu người dùng nhập "abc" thay vì số?
Chương trình sẽ bị "crash". try-catch dùng để bắt và xử lý các lỗi này.



```
try
{
    int tuoi = int.Parse(Console.ReadLine());
    Console.WriteLine($"Năm sau bạn sẽ {tuoi + 1} tuổi.");
}
catch (FormatException)
{
    Console.WriteLine("Lỗi: Vui lòng nhập một con số hợp lệ.");
}
```

