AI driven development

Github Copilot

What is GitHub Copilot?

- Tool to write easier and faster code
- Powered by GPT-3
  - Generative Pre-trained Transformer 3 (GPT-3)
  - Large language model released by OpenAI in 2020

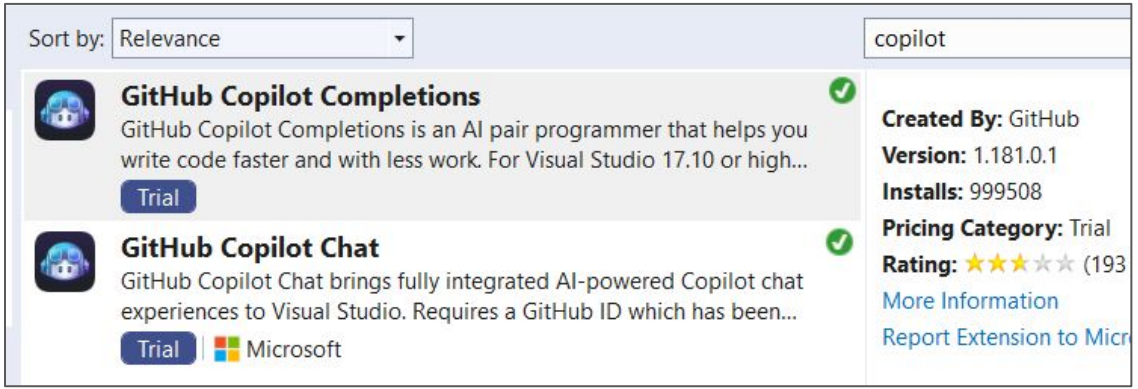
General idea

1. Write a description of what you need
2. Copilot will create for you

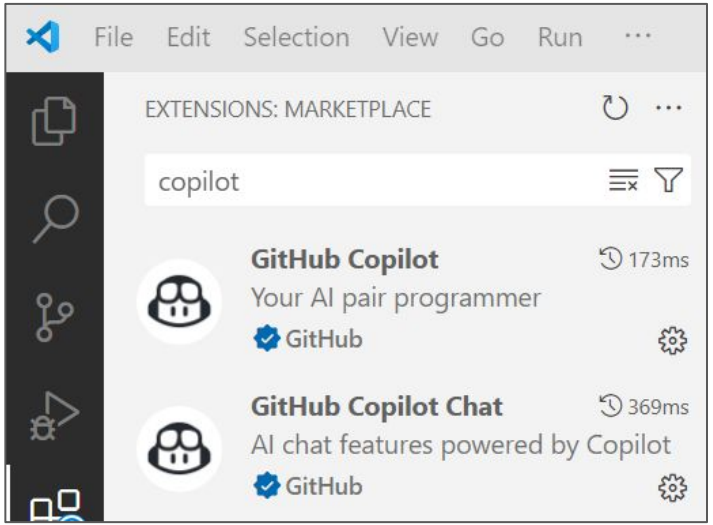
Get Copilot as a student

1. Go to https://education.github.com/benefits
2. Use your student email to register
3. Still need to wait 2-3 days to be activated

Extensions to install



Visual Studio



VS Code

- ❑ Login using your Github account

Ask Copilot for help instructions

tdquang7

/help what can you do

GitHub Copilot

You can ask me general programming questions, or chat with the following participants which have specialized expertise and can perform actions:

- `@editor` - undefined
- `@workspace` - Ask about your workspace
  - `/explain` - Explain how the code in your active editor works
  - `/tests` - Generate unit tests for the selected code
  - `/fix` - Propose a fix for the problems in the selected code
  - `/new` - Scaffold code for a new workspace
  - `/newNotebook` - Create a new Jupyter Notebook
- `@vscode` - Ask questions about VS Code
  - `/search` - Generate query parameters for workspace search
  - `/api` - Ask about VS Code extension development
- `@terminal` - Ask how to do something in the terminal
  - `/explain` - Explain something in the terminal

You can also help me understand your question by using the following variables to give me extra context:

- `#selection` - The current selection in the active editor
- `#editor` - The visible source code in the active editor
- `#terminalLastCommand` - The active terminal's last run command
- `#terminalSelection` - The active terminal's selection

To have a great conversation, ask me questions as if I was a real programmer:

- **Show me the code** you want to talk about by having the files open and selecting the most important lines.
- **Make refinements** by asking me follow-up questions, adding clarifications, providing errors, etc.
- **Review my suggested code** and tell me about issues or improvements, so I can iterate on it.

Code generation

Two ways to use Copilot to generate code

1. Manually write description in comment
2. Automatically suggest code as you write

Generate a class Fraction

- ❏ Write comment: “//class Fraction” > Enter

Code explanation

Ask Copilot to explain the code for you /explain

GitHub Copilot

The selected code defines a Fraction class in C# that represents a mathematical fraction. Here's a breakdown of its components:

- **Fields**: The class has two private fields, numerator and denominator, which store the numerator and denominator of the fraction, respectively.
- **Properties**: The Numerator and Denominator properties provide public access to the numerator and denominator fields. They allow the fields to be read and modified from outside the class.
- **Constructors**: The class has two constructors. The parameterless constructor initializes a new Fraction instance with a numerator of 0 and a denominator of 1. The second constructor takes two parameters and initializes a new Fraction instance with the specified numerator and denominator. It also includes a check to prevent the denominator from being set to zero, which is not allowed in mathematics.
- **Methods**: The class includes several methods for working with fractions:

- **Methods**: The class includes several methods for working with fractions:

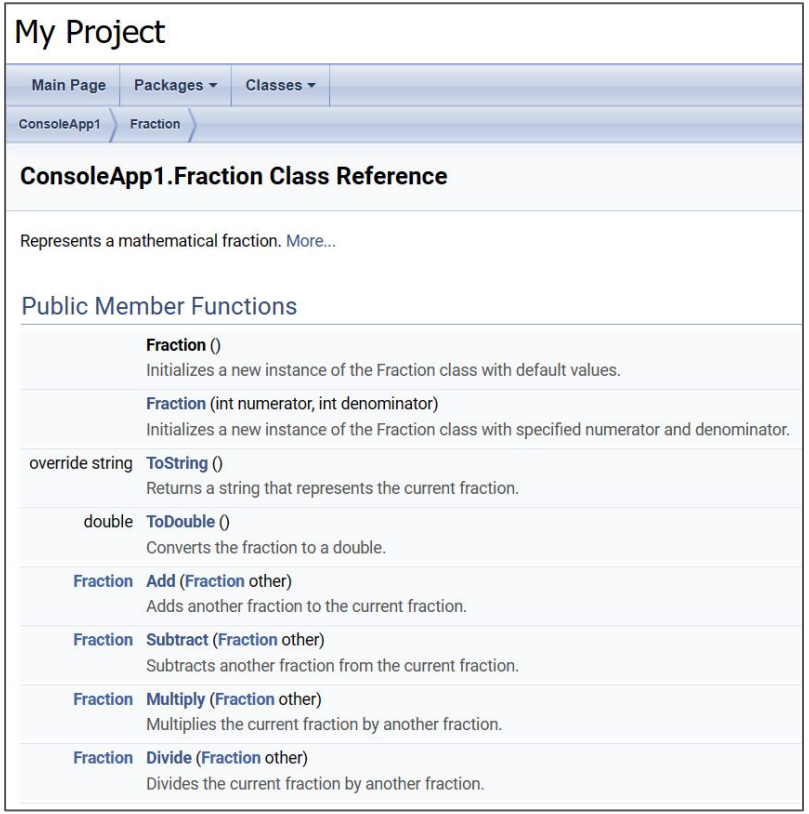
  - ToString: This method overrides the base ToString method and returns a string representation of the fraction in the format "numerator/denominator".
  - ToDouble: This method converts the fraction to a double by dividing the numerator by the denominator.
  - Add: This method adds another Fraction to the current Fraction and returns the result as a new Fraction object.
  - Subtract: This method subtracts another Fraction from the current Fraction and returns the result as a new Fraction object.
  - Multiply: This method multiplies the current Fraction by another Fraction and returns the result as a new Fraction object.
  - Divide: This method divides the current Fraction by another Fraction and returns the result as a new Fraction object.

Each of the arithmetic methods (Add, Subtract, Multiply, Divide) creates a new Fraction object to hold the result. This is because fractions are immutable in mathematics - once a fraction is created, its value doesn't change. Instead, operations on fractions return

Create documentation

Ask Copilot to generate documentation: /doc

- ❑ Preview then accept
- ❑ Use **doxygen** to create documentation


My Project

Main Page | Packages | Classes

ConsoleApp1 › Fraction

ConsoleApp1.Fraction Class Reference

Represents a mathematical fraction. More...

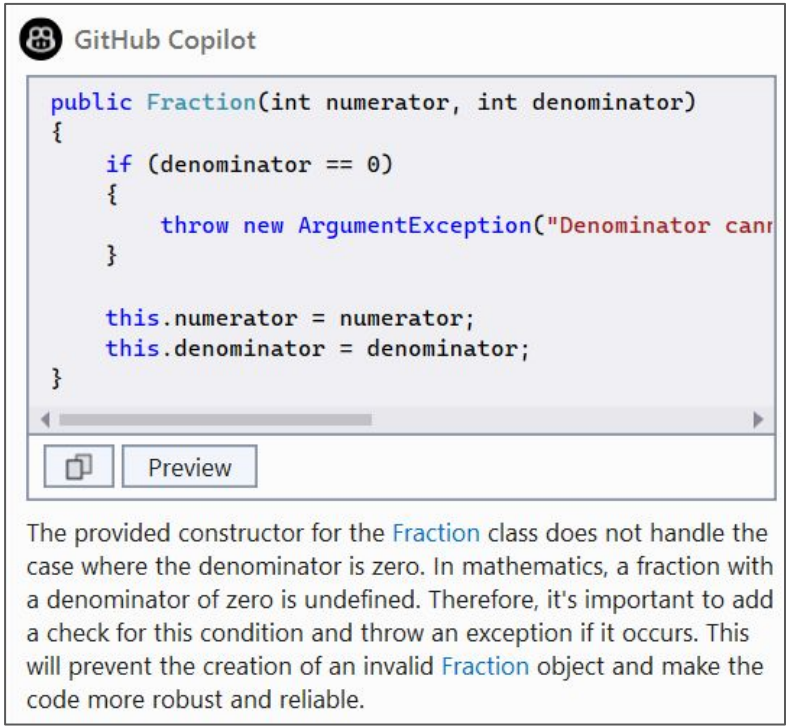
Public Member Functions

	Fraction () Initializes a new instance of the Fraction class with default values.
	Fraction (int numerator, int denominator) Initializes a new instance of the Fraction class with specified numerator and denominator.
override string	**ToString** () Returns a string that represents the current fraction.
double	**ToDouble** () Converts the fraction to a double.
Fraction	**Add** (**Fraction** other) Adds another fraction to the current fraction.
Fraction	**Subtract** (**Fraction** other) Subtracts another fraction from the current fraction.
Fraction	**Multiply** (**Fraction** other) Multiplies the current fraction by another fraction.
Fraction	**Divide** (**Fraction** other) Divides the current fraction by another fraction.

Code improvement

Ask Copilot to improve your code /optimize

```
public Fraction(int numerator, int denominator)
{
    this.numerator = numerator;
    this.denominator = denominator;
}
```



- To apply the change, click **Preview > Accept**

Other example of /optimize

```
public Fraction Add(Fraction other)
{
    int newNumerator = numerator * other.denominator + o
    int newDenominator = denominator * other.denominator
    return new Fraction(newNumerator, newDenominator);
}
```

GitHub Copilot

```
public Fraction Add(Fraction other)
{
    checked
    {
        int newNumerator = numerator * other.denomina
        int newDenominator = denominator * other.deno
        return new Fraction(newNumerator, newDenomina
    }
}
```

Preview

The Add method performs arithmetic operations that could potentially result in an overflow. If the values of the fractions are large enough, the multiplication and addition operations could exceed the maximum value that an int can hold, leading to incorrect results due to overflow.

To prevent this, I've added the checked keyword, which forces the runtime to throw an OverflowException if an overflow occurs. This makes the code safer by preventing silent overflow errors.

Please note that using checked can slightly decrease performance due to the additional overflow checks. However, in most cases, the

What if we want to improve more?

“Improve the selected function to return a reduced fraction using greatest common divisor”

GitHub Copilot

```
public Fraction Add(Fraction other)
{
    checked
    {
        int newNumerator = numerator * other.denomina
        int newDenominator = denominator * other.deno

        // Calculate the greatest common divisor (GCD
        int gcd = GCD(newNumerator, newDenominator);

        // Divide the numerator and the denominator b
        newNumerator /= gcd;
        newDenominator /= gcd;

        return new Fraction(newNumerator, newDenomina
    }
}
```

```
// Method to calculate the greatest c
private static int GCD(int a, int b)
{
    while (b != 0)
    {
        int temp = b;
        b = a % b;
        a = temp;
    }
    return a;
}
```

Let’s ask Copilot to add a Parse function

```
// Generate Parse function
0 references
public static Fraction Parse(string s)
{
    string[] parts = s.Split('/');
    if (parts.Length != 2)
    {
        throw new FormatException("Input string was n
    }

    int numerator = int.Parse(parts[0]);
    int denominator = int.Parse(parts[1]);
    return new Fraction(numerator, denominator);
}
```

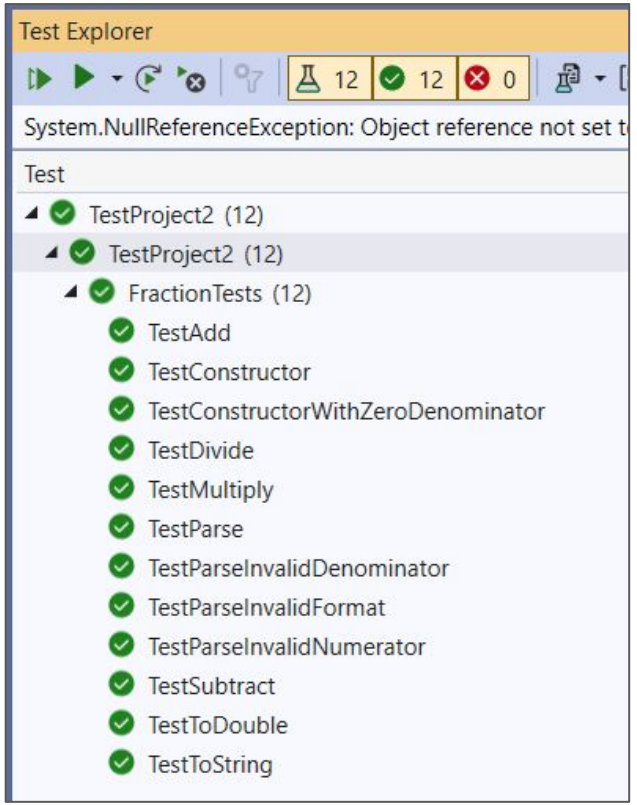
How about

- Explain the code for us?
- Adding comments to create documentation?
- Could we improve / optimize the code more?
- How can our teammate be sure that our code is good?
  - => Unit test

Test generation

Generate unit tests with /tests

1. Create MSTest Test project
2. Copy&Paste the generated code
3. Add project reference



Problem? Naming convention not good

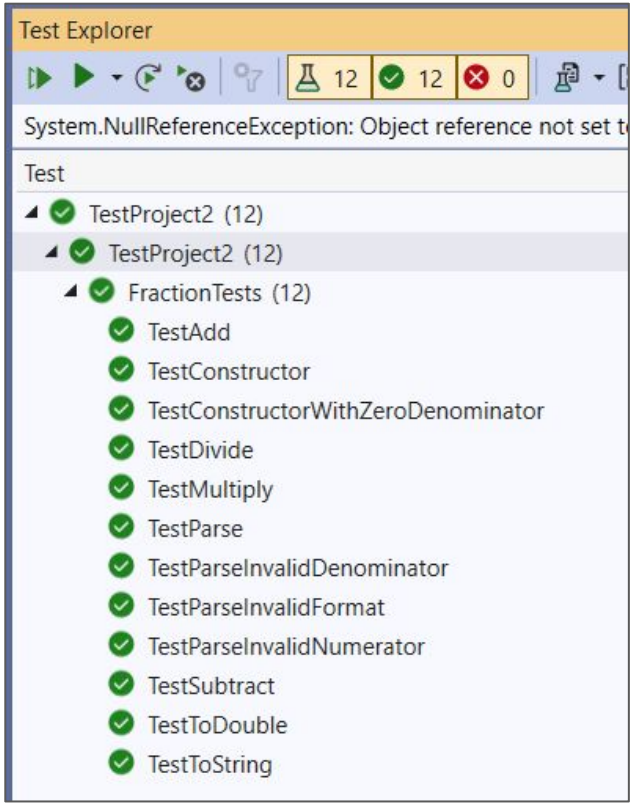
Asking Copilot to fix it

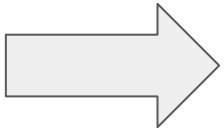
- Function naming convention should be:

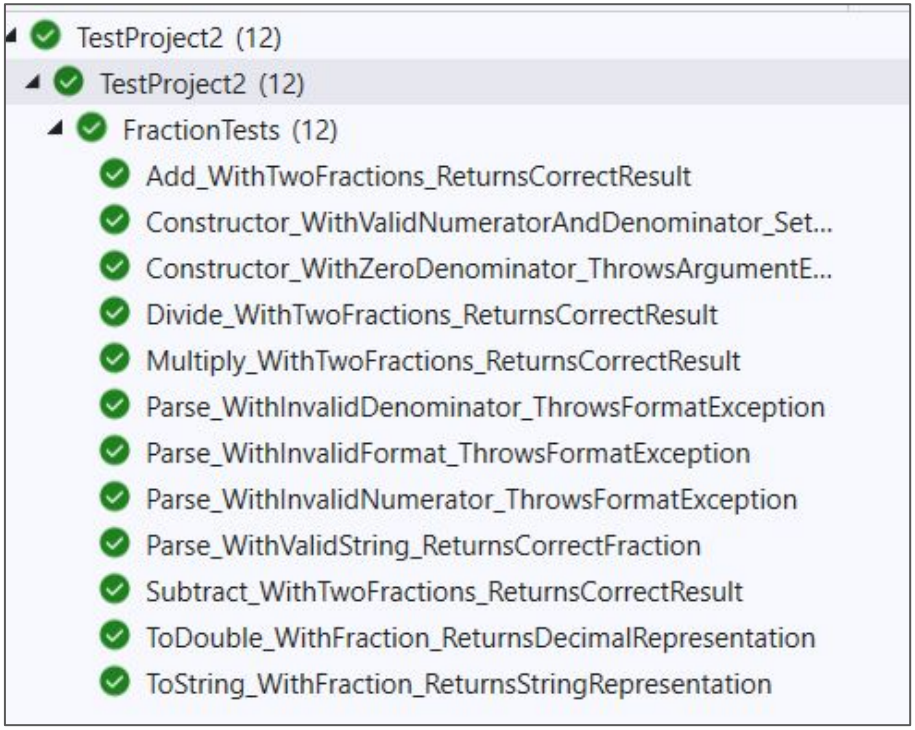
UnitOfWork_StateUnderTest_ExpectedBehavior

- Prompt: “**Improve the name of all the functions using the naming standard UnitOfWork_StateUnderTest_ExpectedBehavior**”

Better result







Fixing bugs

In the main function

// Generate randomly 20 fractions into a list

// Sort the list of fractions

// Print out all the fractions

```
Unhandled exception. System.InvalidOperationException: Failed to compare two elements in the array.
 ---> System.ArgumentException: At least one object must implement IComparable.
   at System.Collections.Comparer.Compare(Object a, Object b)
   at System.Collections.Generic.ArraySortHelper`1.SwapIfGreater(Span`1 keys, Comparison`1 comparer, Int32 i, Int32 j)
   at System.Collections.Generic.ArraySortHelper`1.PickPivotAndPartition(Span`1 keys, Comparison`1 comparer)
   at System.Collections.Generic.ArraySortHelper`1.IntroSort(Span`1 keys, Int32 depthLimit, Comparison`1 comparer)
   at System.Collections.Generic.ArraySortHelper`1.IntrospectiveSort(Span`1 keys, Comparison`1 comparer)
   at System.Collections.Generic.ArraySortHelper`1.Sort(Span`1 keys, IComparer`1 comparer)
   --- End of inner exception stack trace ---
   at System.Collections.Generic.ArraySortHelper`1.Sort(Span`1 keys, IComparer`1 comparer)
   at System.Array.Sort[T](T[] array, Int32 index, Int32 length, IComparer`1 comparer)
   at System.Collections.Generic.List`1.Sort(Int32 index, Int32 count, IComparer`1 comparer)
   at ConsoleApp1.Program.Main(String[] args) in C:\Users\tdqua\source\repos\ConsoleApp1\ConsoleApp1\Program.cs:line 19
```

Ask Copilot to explain the error

GitHub Copilot

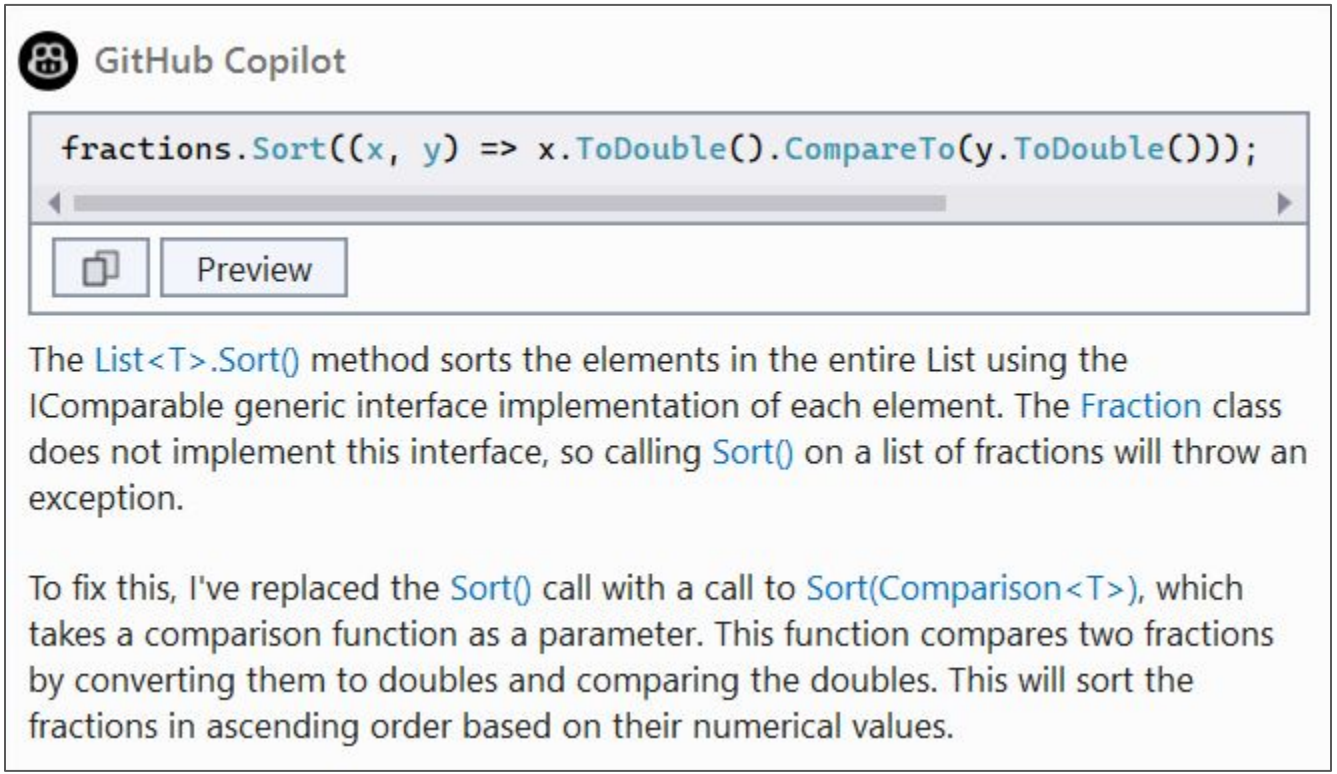
This error occurs when you try to sort a collection of objects that do not implement the IComparable interface. The IComparable interface defines a method CompareTo that two objects can use to compare themselves to each other. The .NET sorting methods require this interface to determine how to order the objects.

In your case, the error message "At least one object must implement IComparable" suggests that you're trying to sort an array or list of objects that don't implement IComparable. The sorting algorithm doesn't know how to compare these objects, so it throws an InvalidOperationException.

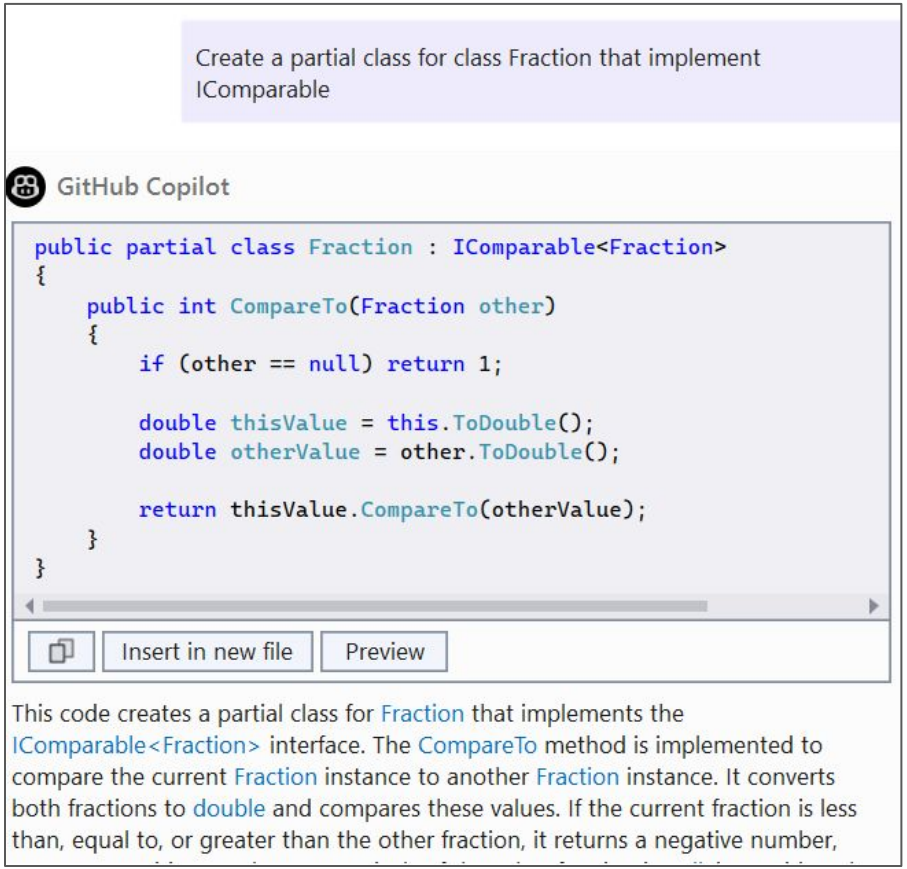
To fix this error, you have two options:

1. Make the class of the objects you're trying to sort implement the IComparable or IComparable<T> interface, and provide a CompareTo method that defines how to compare two objects of this class.
2. If you can't or don't want to modify the class, you can provide a custom IComparer or Comparison<T> when you call the sort method. This is a function that takes two objects and returns a value indicating which one is greater.

Ask Copilot to fix the error /fix



Alternative to improve



Further improvement

Improve this code, print the fraction in string format and decimal format too

GitHub Copilot

```
foreach (Fraction fraction in fractions)
{
    Console.WriteLine($"{fraction} ({fraction.ToDouble():F2})");
}
```

Preview

This code will print each fraction in both string format (e.g., "3/4") and decimal format (e.g., "0.75"). The :F2 format specifier in fraction.ToDouble():F2 indicates that the decimal should be formatted as a fixed-point number with two digits after the decimal point.

```
2/7 (0.29)
2/7 (0.29)
1/3 (0.33)
4/7 (0.57)
3/5 (0.60)
7/9 (0.78)
6/6 (1.00)
7/7 (1.00)
9/7 (1.29)
8/6 (1.33)
9/6 (1.50)
6/4 (1.50)
5/3 (1.67)
4/2 (2.00)
8/3 (2.67)
8/3 (2.67)
6/2 (3.00)
8/2 (4.00)
9/2 (4.50)
9/2 (4.50)
```

Some other tips

Other things

```
// Create query that finds all fractions that are greater than 0.5
var query = from fraction in fractions
            where fraction.ToDouble() > 0.5
            select fraction;
```

```
// Print out all the fractions but show only numerator
foreach (Fraction fraction in query)
{
    Console.WriteLine(fraction.Numerator);
}
```

Translate your comment

```
// Dịch sang tiếng Anh: "Tìm các phân số có giá trị lớn nhất và nhỏ nhất"
// Find the fractions with the largest and smallest values
```

Quickly prepared sample data

```
// Create a list of 15 students
List<Student> students = new List<Student>
{
    new Student { ID = "S001", Name = "Alice", GPA = 3.5, Email = ""},
    new Student { ID = "S002", Name = "Bob", GPA = 3.2, Email = ""},
    new Student { ID = "S003", Name = "Charlie", GPA = 3.7, Email = ""},
    new Student { ID = "S004", Name = "David", GPA = 3.1, Email = ""},
    new Student { ID = "S005", Name = "Eve", GPA = 3.6, Email = ""},
};
```

Quick reminder of regular expression

```
// Validate email
var emailPattern = @"^([\w\.\-]+)@([\w\-]+)((\.(\w){2,3})+)$";
```

explain the regular expression "@"^([\w.-]+)@([\w-]+)((.(\w){2,3})+)$";"

GitHub Copilot

The regular expression @"^([\w\.\-]+)@([\w\-]+)((\.(\w){2,3})+)$" is used to validate email addresses. Here's a breakdown of what each part does:

- ^ : This asserts the start of a line. The email address must start here.
- ([\w\.\-]+) : This matches one or more word characters (which include letters, numbers, and underscores), periods, or hyphens. This part corresponds to the local part of the email address (before the @ symbol).
- @ : This matches the @ symbol exactly.
- ([\w\-]+) : This matches one or more word characters or hyphens. This part corresponds to the domain name of the email address (after the @ symbol and before the dot).
- ((\.(\w){2,3})+) : This matches one or more occurrences of a period followed by two or three word characters. This part corresponds to the domain extension (like .com, .org, .net).
- $: This asserts the end of a line. The email address must end here.

So, this regular expression will match any string that starts with one or more word characters, periods, or hyphens, followed by an @ symbol, followed by one or more word characters or hyphens, followed by one or more occurrences of a period and two or three word characters, and ends there. This corresponds to the general structure of an email address.

Converting to another language

- Try translate & explain
- For example
  - Write a function to calculate the sum of an integer array in **C#**.
  - Convert this function into **python**

Database schema & data creation

- Generate the script to create the tables and foreign keys:

Table: **User**, Fields and Types: id auto, firstName text(50), lastName text(50), email text(50), password text(20)
Table: **Project**, Fields and Types: id auto, name text(50)
Table: **ExpenseType**, Fields and Types: id auto, name text(50)
Table: **Expense**, Fields and Types: id auto, description text(100), idExpenseType numeric, idProject numeric, idUser numeric, date date

- Generate the script to fill with sample data those tables: 5 Users, 3 Projects, 4 ExpenseTypes
- Generate 30 Expenses with random combination for users, projects and expenseTypes, with dates between November 15 and December 15, 2022

What about models / entities from the database?

- Prompt: Create the C# classes to map those tables

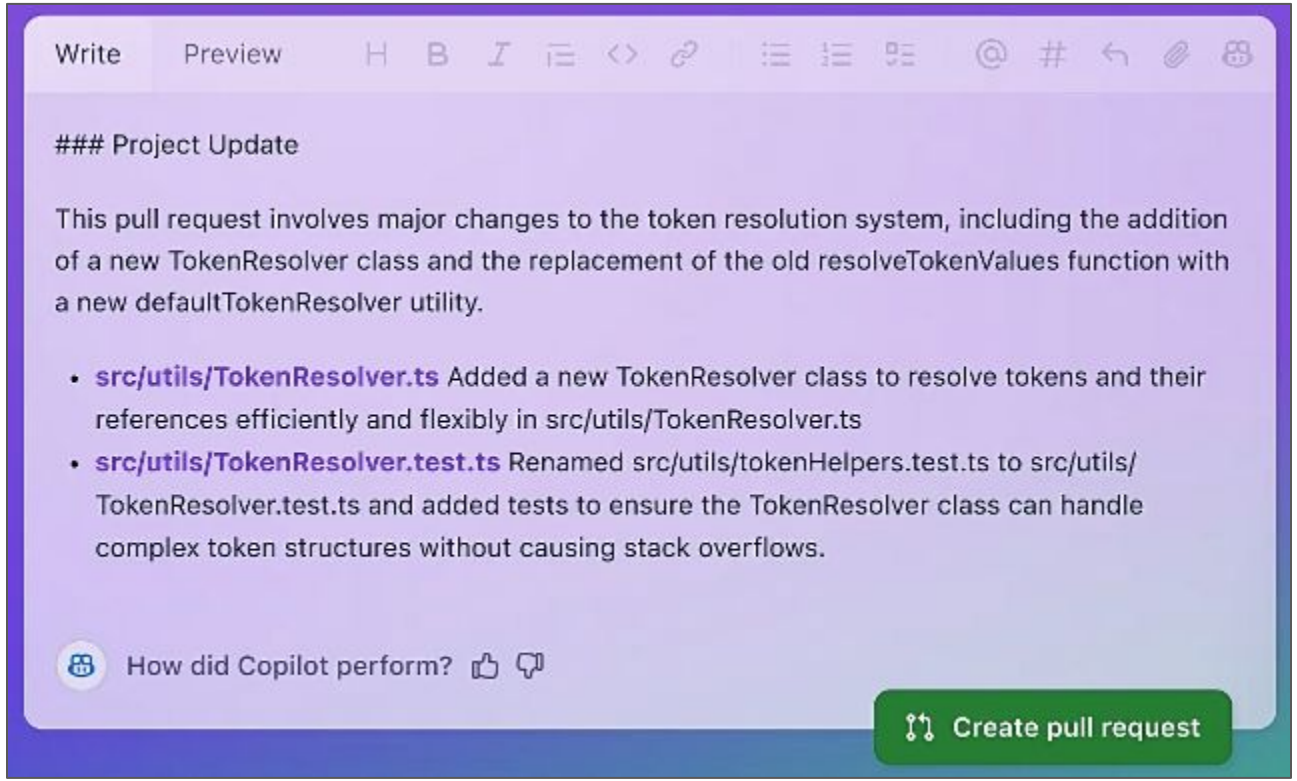
```
public class Expense
{
    public int Id { get; set; }
    public string Description { get; set; }
    public int IdExpenseType { get; set; }
    public int IdProject { get; set; }
    public int IdUser { get; set; }
    public DateTime Date { get; set; }

    // Navigation properties
    public User User { get; set; }
    public Project Project { get; set; }
    public ExpenseType ExpenseType { get; set; }
}
```

What should we notice?

- The AI is aware of the flow of the conversation and previous generated code.
- It can continue to work on the chat history
- Context window

Pull request suggestions



IDX + Gemini

- Activate: **Ctrl + Shift + Space**
- Inline code help: **Ctrl + I**

[Get code help with Gemini | Project IDX | Google for Developers]

Some pros & cons

Benefits & Drawbacks (1)

Benefits	Drawbacks
Increased efficiency and productivity AI-driven development tools can automate repetitive and time-consuming tasks, such as code generation and testing, allowing developers to focus on more complex and higher-value work.	**High initial costs** Implementing AI-driven development tools and processes can be costly in terms of both time and money. However, subscription models such as Copilot can solve the issue for developers.

Benefits & Drawbacks (2)

Benefits	Drawbacks
Improved code quality AI-driven development tools can analyze code and identify potential bugs or errors, helping to improve the overall quality of the code.	**Limited capabilities** AI-driven development tools are still in the early stages of development and may not be able to handle all aspects of the software development process.

Benefits & Drawbacks (3)

Benefits	Drawbacks
Enhanced scalability AI-driven development tools can help identify and address scalability issues early on in the development process, allowing for the creation of more robust and scalable software.	**Dependence on data** AI-driven development tools rely on data to function, which can be a problem if the data is not accurate or of high quality.

Benefits & Drawbacks (4)

Benefits	Drawbacks
Faster time to market By automating certain tasks, AI-driven development can help to speed up the overall development process, allowing for faster deployment and time-to-market.	**Lack of transparency** Some of the decisions made by AI-driven development tools can be difficult to understand or explain, which can be a problem if stakeholders need to understand the reasoning behind certain decisions.

Benefits & Drawbacks (5)

Benefits	Drawbacks
Better collaboration AI-driven development tools can help bridge communication gaps between developers, stakeholders, and customers, leading to better collaboration and more efficient decision-making	**Job Losses** As AI-driven development automates repetitive and time-consuming tasks, some less skilled jobs may become redundant, leading to job losses

Further reading

What’s more

- 10 Best AI Chatbots for Coding in 2025?
- Strategy for prompt engineering
  https://microsoft.github.io/prompt-engineering

Related - Better Prompts

- Awesome prompts

https://github.com/f/awesome-chatgpt-prompts

Gemini prompting guide 101