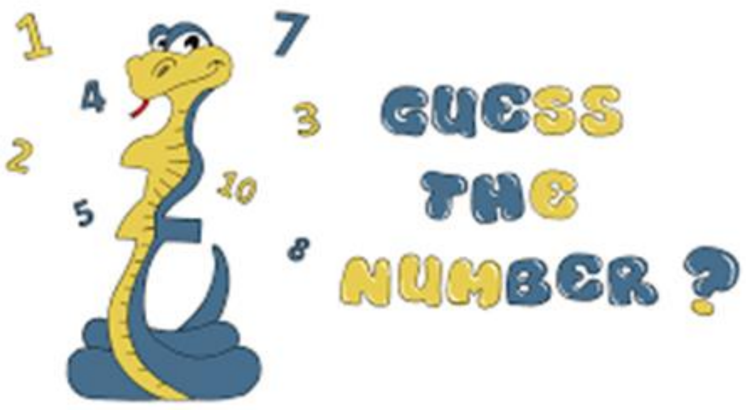
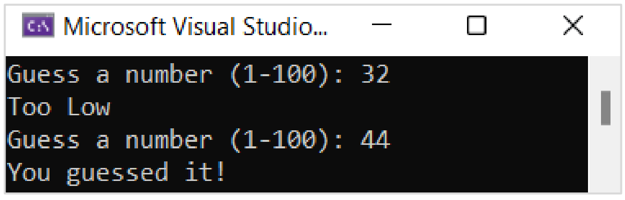
Practical Project: Guess A Number

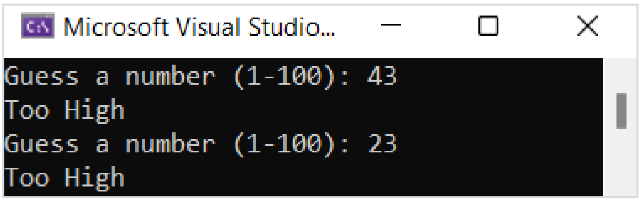
This is additional practical project and **it is not mandatory and it is not included in the final score**. The main purpose is to use gained knowledge in different type of problems and to improve your portfolio and GitHub skills.



Today we will make the console game "**`Guess A Number`**". "**`Guess A Number`**" is a game, in which your opponent "**the computer**" chooses a **random** number between **1 and 100** and your task is to **guess** this number. After each number you enter, the computer will give you a **hint** of whether the number is **greater** or **less** than the number you selected until you guess the **correct** number:



```
Guess a number (1-100): 32
Too Low
Guess a number (1-100): 44
You guessed it!
```

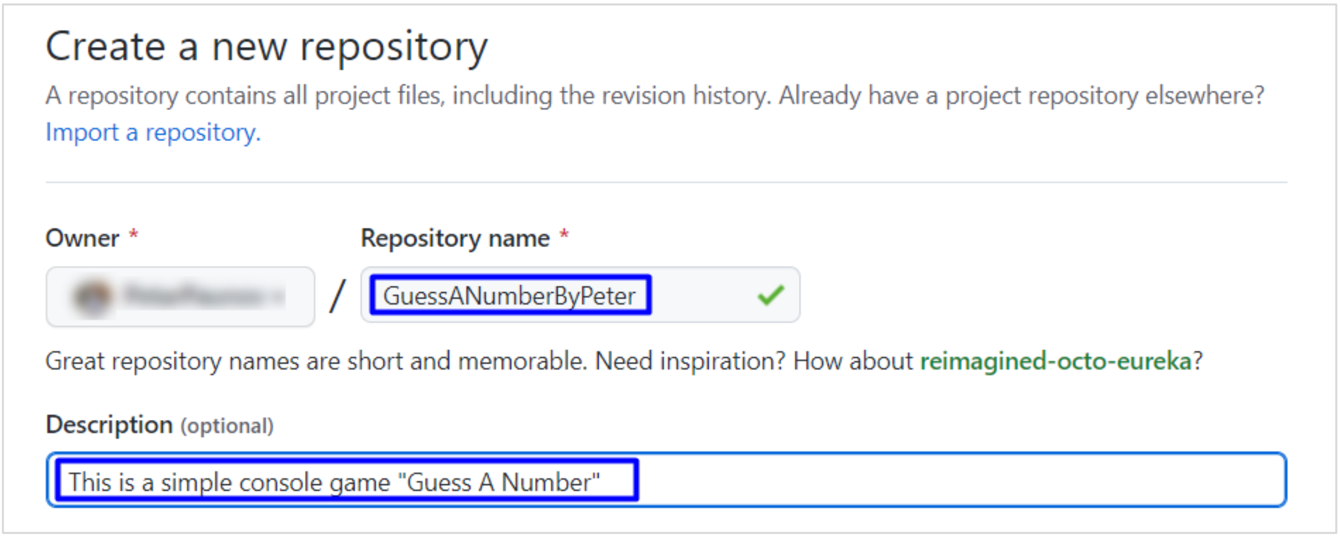


```
Guess a number (1-100): 43
Too High
Guess a number (1-100): 23
Too High
```

1. Create GitHub Repository

We already have a **`GitHub`** account created, so we're moving directly to creating a new **repository**.

Create a **new repository** from: https://github.com/new. Choose a **meaningful name**, e. g. "**`GuessANumberByUsername`**" add a **short description** and make your repo **public**:



Create a new repository

A repository contains all project files, including the revision history. Already have a project repository elsewhere? Import a repository.

Owner * / Repository name *

GuessANumberByPeter

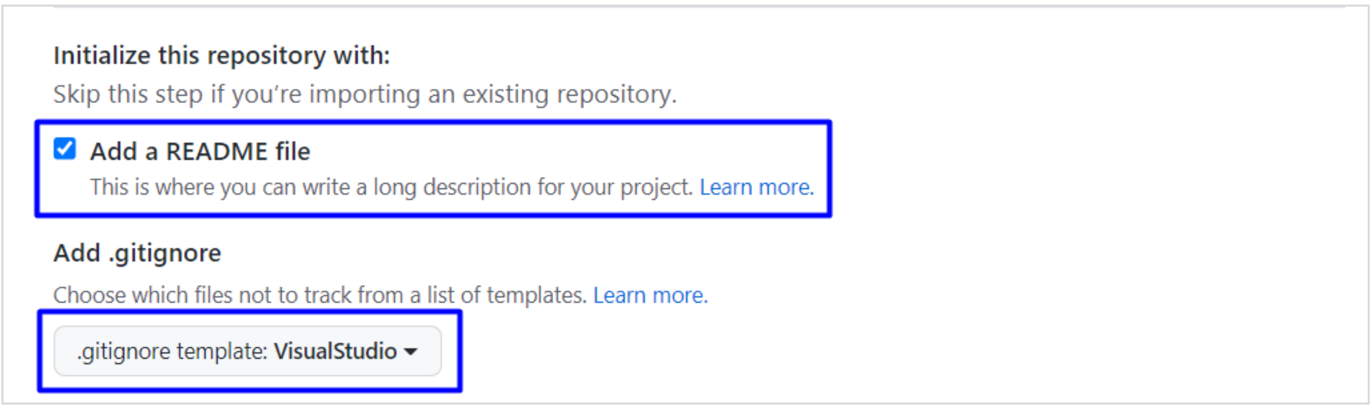
Great repository names are short and memorable. Need inspiration? How about reimagined-octo-eureka?

Description (optional)

This is a simple console game "Guess A Number"

⚠	Please choose **your own original and unique name** for your project! Your GitHub profile should be **unique**, not the same as your colleagues'. You can follow this tutorial, but you can also **make changes** and **implement your project different** from your colleagues.

Also, **add a `README.md`** file and **`.gitignore` for Visual Studio**, as shown below:



Initialize this repository with:

Skip this step if you're importing an existing repository.

☑ Add a README file

This is where you can write a long description for your project. Learn more.

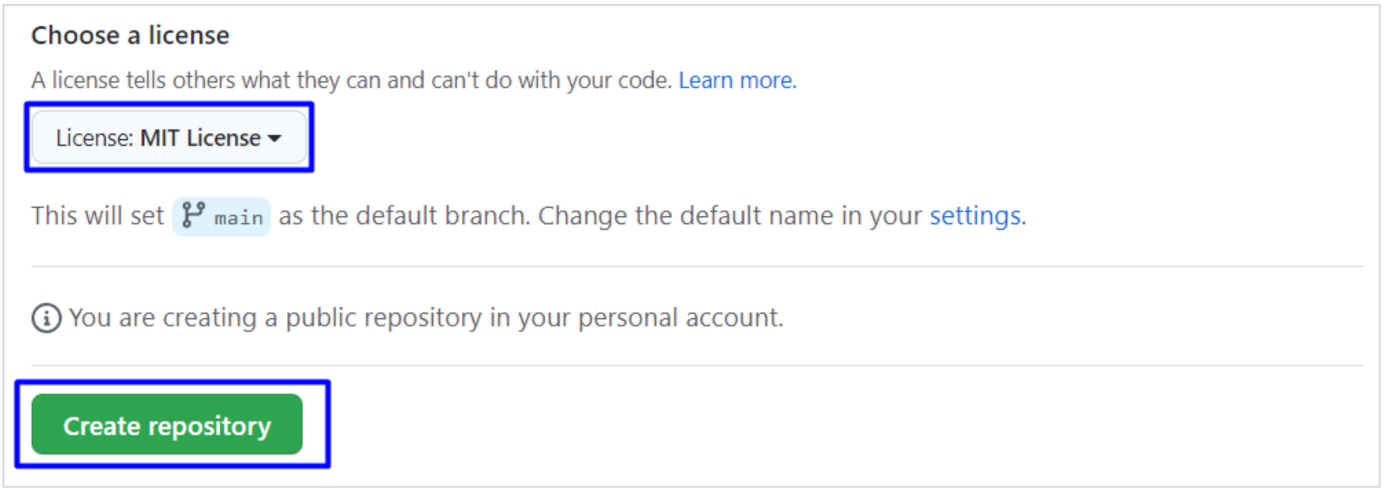
Add .gitignore

Choose which files not to track from a list of templates. Learn more.

.gitignore template: VisualStudio

In Git projects the **`.gitignore` file** specifies which files from your repo are not part of the source code and should be ignored (not uploaded in the GitHub repo). Typically in GitHub, we upload in the repo **only the source code** and we don't upload the compiled binaries and temp files.

Finally, **change the license** to "**`MIT`**", which is the most widely used open-source license, and click on the **`[Create]` button** to **create your repository**:



Choose a license

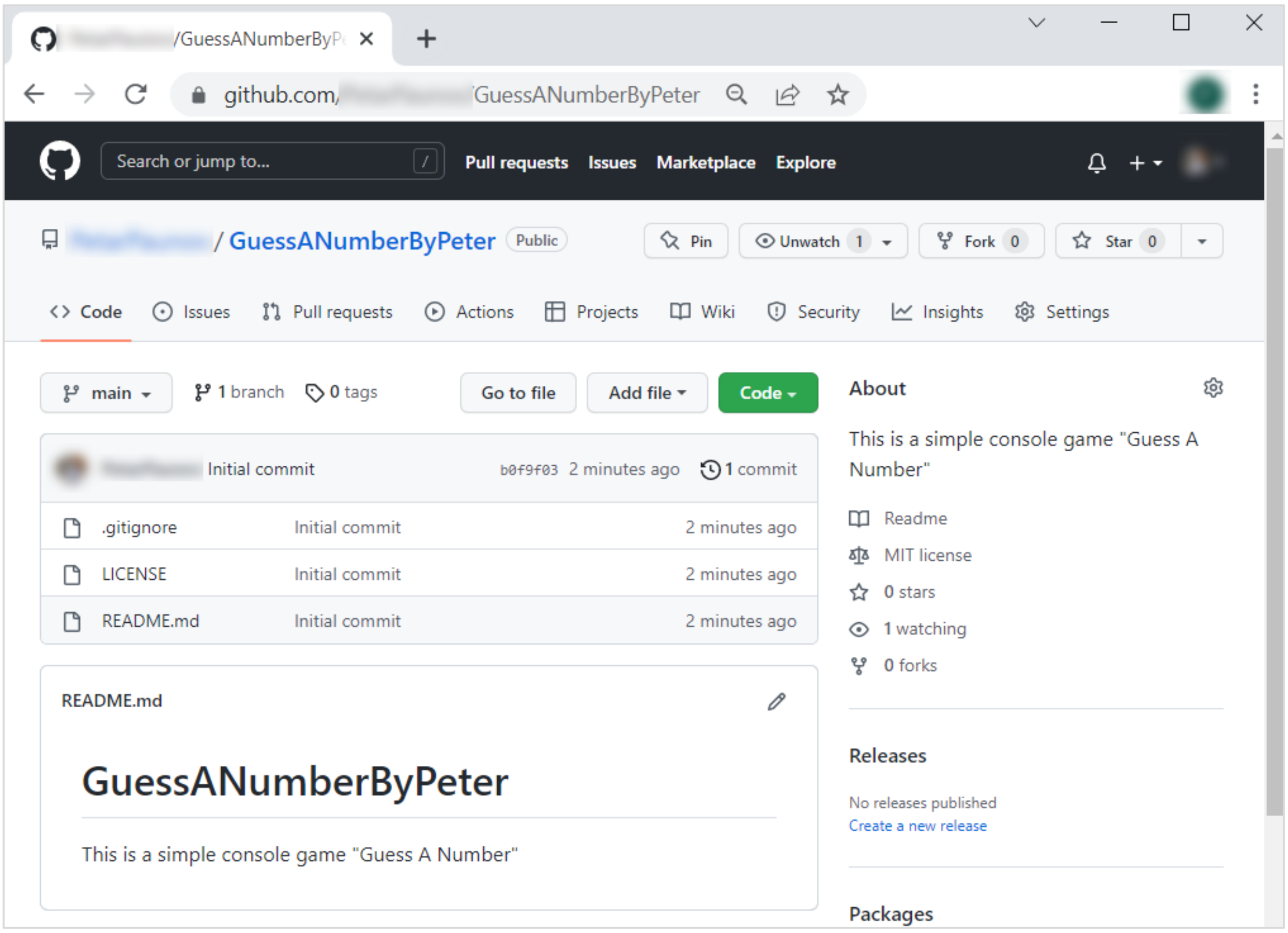
A license tells others what they can and can't do with your code. Learn more.

License: MIT License

This will set `main` as the default branch. Change the default name in your settings.

You are creating a public repository in your personal account.

Now your **repository is created** and looks like this:


[redacted]/GuessANumberByP

github.com/[redacted]/GuessANumberByPeter

Search or jump to... / Pull requests Issues Marketplace Explore

[redacted] / GuessANumberByPeter Public

Pin | Unwatch 1 | Fork 0 | Star 0

Code | Issues | Pull requests | Actions | Projects | Wiki | Security | Insights | Settings

main | 1 branch | 0 tags | Go to file | Add file | Code

[redacted] Initial commit	b0f9f03 2 minutes ago	1 commit
.gitignore	Initial commit	2 minutes ago
LICENSE	Initial commit	2 minutes ago
README.md	Initial commit	2 minutes ago

README.md

GuessANumberByPeter

This is a simple console game "Guess A Number"

About

This is a simple console game "Guess A Number"

Readme
MIT license
0 stars
1 watching
0 forks

Releases

No releases published
Create a new release

Packages

Now let's see how to **write the code** of our game.

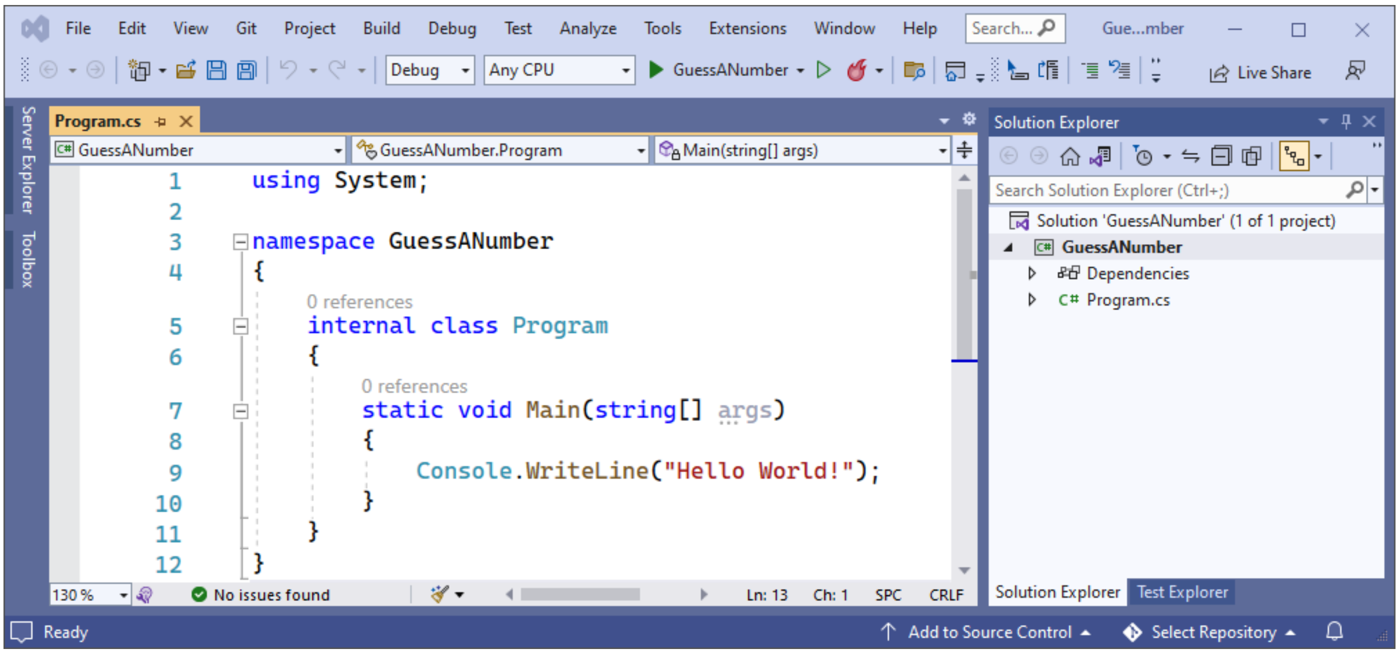
2. Write the Game's Code

Let's create the game and play with it.

Create a Visual Studio Project

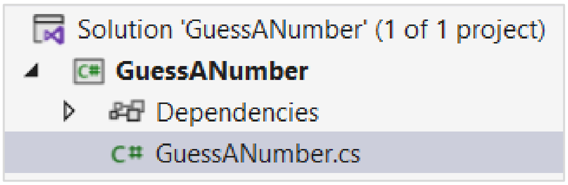
First, we should **start Visual Studio** and **create a new C# console application**. Then, **choose an appropriate name** and a **place to save the project**. On the next screen, choose **`[.NET 3.1 (Long-term support)]`** and create the project.

Our project should be created and should look like this:



```
using System;

namespace GuessANumber
{
    0 references
    internal class Program
    {
        0 references
        static void Main(string[] args)
        {
            Console.WriteLine("Hello World!");
        }
    }
}
```

Before we continue, let's change the name of our main class – **`Program.cs`** to something more **meaningful**. You already know how to do this:



Implement the Game Logic

Now let's start working on our project.

Read Player's Move

Create a variable from type "**Random**", which will help us **choose** a number **randomly** by using the method "**`Next()`**". We will use this **method** so each time the computer can choose a number **between** "**1 and 100**" **randomly**.

```
Random randomNumber = new Random();
int computerNumber = randomNumber.Next(1, 101);
```

A little more information about Random:

.NET Core provides thousands of ready-to-use classes that are packaged into namespaces like the already known **`System`**. The **`System`** namespace contains fundamental classes, one of which is the **Random** class. It provides functionality to generate random numbers in C#. We will learn more about the **Random** class in the Objects and Classes lesson, but let's take a quick overall view of this class.

The line with code below creates a new object, which is an instance of the Random class. In this object will store the randomly generated number that we have to guess.

```
Random randomNumber = new Random();
```

The following code returns a random number, using the **`Next()`** method. This is a method, provided by the **Random** class. By writing "**`1, 101`**" in the brackets, we indicate to the method that we want our randomly generated number to be in the range between **1** and **100**. You should note that the lower bound is inclusive and the upper bound is exclusive, that's why we have **101** as the second parameter of the **`Next()`** method.

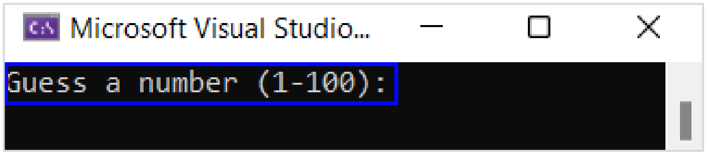
```
int computerNumber = randomNumber.Next(1, 101);
```

You can read more about the **Random** class here https://docs.microsoft.com/en-us/dotnet/api/system.random.next?view=net-6.0.

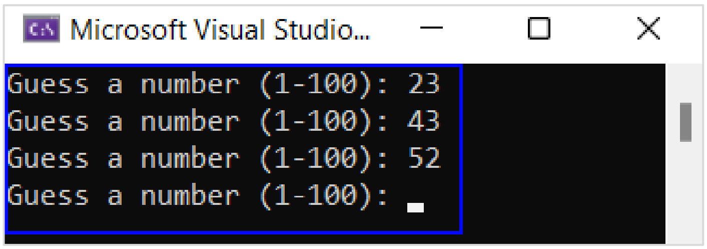
Now write a **`while-loop`** to **iterate,** until the player guesses the computer's **random** number. Write on the console what the player should do and **read** his **input data**. You already know how to do that.

```
while (true)
{
    [redacted]"Guess a number (1-100): ");

    string playerInput = [redacted]
}
```

Now let's run the **app** in the console and check whether our current code **works** properly:



```
Guess a number (1-100):
```



```
Guess a number (1-100): 23
Guess a number (1-100): 43
Guess a number (1-100): 52
Guess a number (1-100): _
```

We can see that we have our text **written** on the console and we should be able to **read** the player’s input **repeatedly** because of our **`while-loop`**.

Check the Player's Input

Now **check** the player's input using the **`int.TryParse()`** method. It will review the input data and return us "**`true`**" or "**`false`**" depending on the data **submitted** by the player. If It's a **number** (**what we expect**) the method will return "**`true`**" otherwise "**`false`**".

As a **second parameter** of the method, we give the **`type`** and **`name`** of the **variable** in which the player's data will be **recorded** if he has submitted **correct** data. Do it as follow:

```
while (true)
{
    Console.Write("Guess a number (1-100): ");

    string playerInput = Console.ReadLine();
    bool isValid = int.TryParse(playerInput, out int playerNumber);
```

A little more information about **`int.TryParse():`** https://docs.microsoft.com/en-us/dotnet/api/system.int32.tryparse?view=net-6.0 .

Now we have created our **Boolean** variable "**`isValid`**" and we can write our **`if-else`** statements. First, we should check if the player's input data "**isValid**":

```
if (isValid)
{

}
```

If data is valid write a **nested `if-else` statement** in which we will check all **three** possible cases.

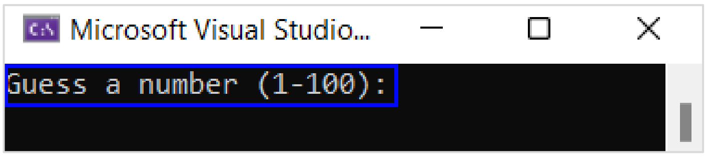
First, if the player's number is **equal** to the computer's number that means the player **guessed** the computer's number, so you should **write** a message, and **stop** the application by using the keyword "**`break`**". Do it like this:

```
if (isValid)
{
    if (playerNumber == computerNumber)
    {
        Console.WriteLine("You guessed it!");
        break;
    }
```

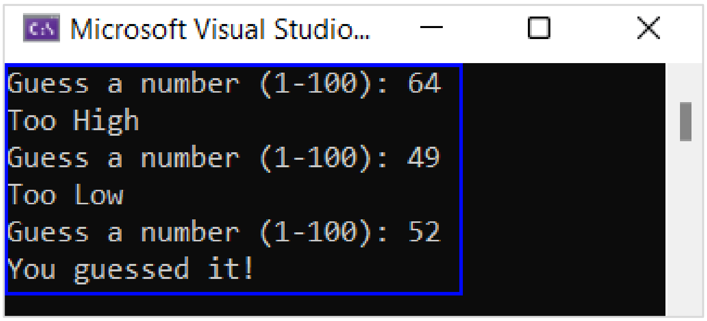
The other **two** cases are if the player's number is **higher** than the computer's number and the player’s number is **less** than the computer's number. Write the rest of the **`else-if` statement** by yourself:

```
if (isValid)
{
    if (playerNumber == computerNumber)
    {
        Console.WriteLine("You guessed it!");
        break;
    }
    else if ([redacted])
    {
        [redacted]("Too High");
    }
    else
    {
        [redacted]("Too Low");
    }
}
```

Now let's run the **app** in the console and check whether our current code **works** properly:



```
Guess a number (1-100):
```



```
Guess a number (1-100): 64
Too High
Guess a number (1-100): 49
Too Low
Guess a number (1-100): 52
You guessed it!
```

We can see that our application work's properly, but we are not finished yet.

Check for Invalid Input

Now what is left is to write **`else`** statement for the **final** case where the player's input is **invalid.** That's all it takes for the **game to work**.

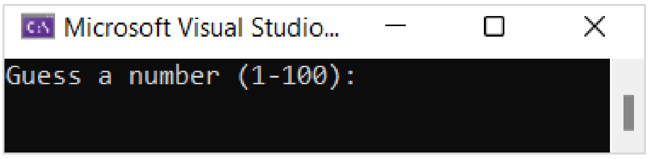
```
    else
    {
        Console.WriteLine("Invalid input.");
    }
}
```

Your entire code should be similar to the following:

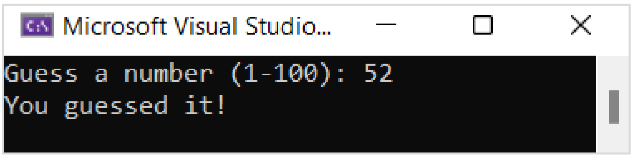
```
Random randomNumber = new Random();
int computerNumber = randomNumber.Next(1, 101);

while (true)
{
    Console.Write("Guess a number (1-100): ");

    string playerInput = Console.ReadLine();
    bool isValid = int.TryParse(playerInput, out int playerNumber);

    if (isValid)
    {
        if (playerNumber == computerNumber)...
        else if (playerNumber > computerNumber)...
        else...
    }
    else
    {
        Console.WriteLine("Invalid input.");
    }
}
```

Test the Application

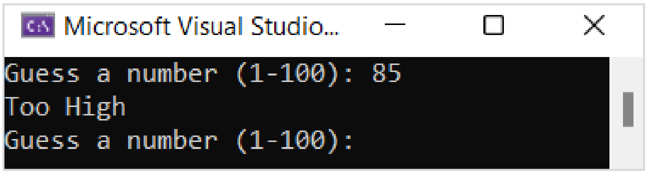
After you **run it**, the game should look like this:



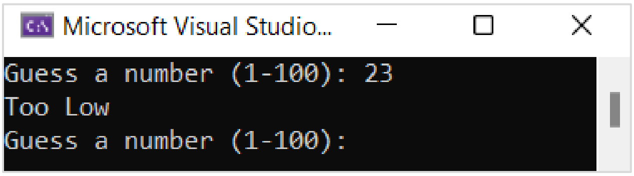
```
Guess a number (1-100):
```



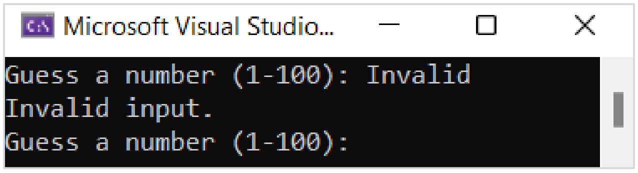
```
Guess a number (1-100): 52
You guessed it!
```



```
Guess a number (1-100): 85
Too High
Guess a number (1-100):
```



```
Guess a number (1-100): 23
Too Low
Guess a number (1-100):
```



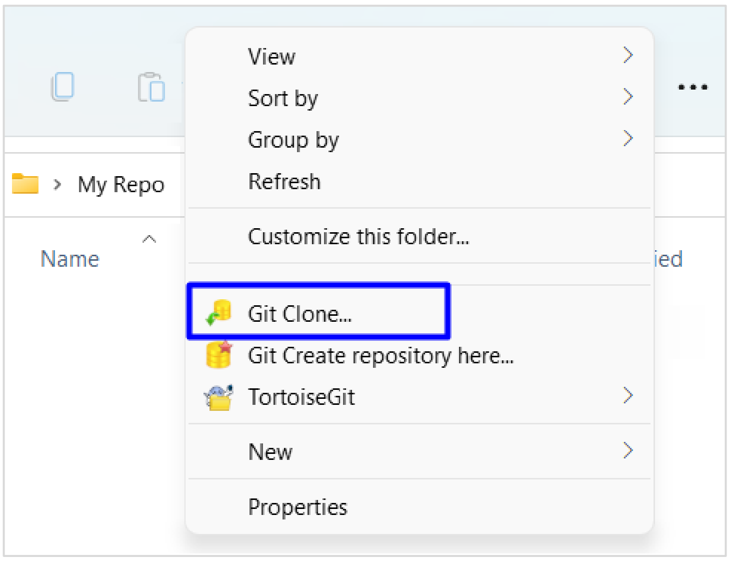
```
Guess a number (1-100): Invalid
Invalid input.
Guess a number (1-100):
```

3. Upload Your Project to Github

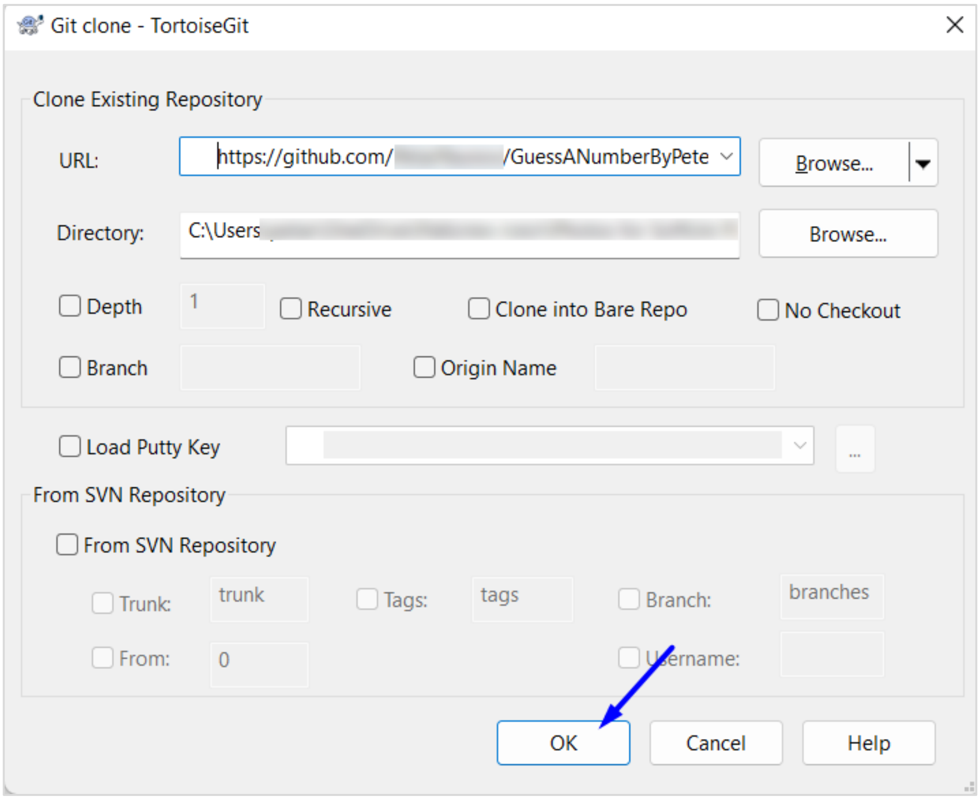
We already know how to clone our repository by using **Git Bash** or **TortoiseGit**.

Use TortoiseGit (Option 1)

Use Git **clone** for cloning with **`TortoiseGit`**. Go to the desired directory, **right-click** on a blank space **anywhere** in the folder and click **`[Git Clone]`**.

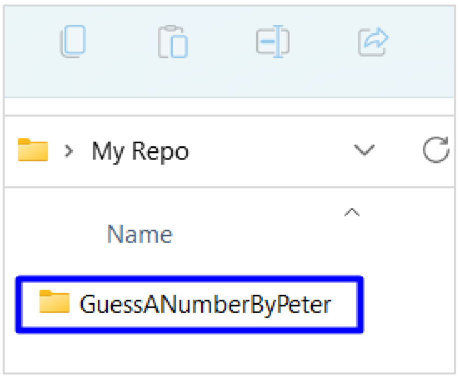


Now go to our newly created **repository** and copy the **repository's URL**, you should already know how to do this. The last thing that we should do is to open **`TortoiseGit`** and paste the **URL** and click **`[OK]`**.

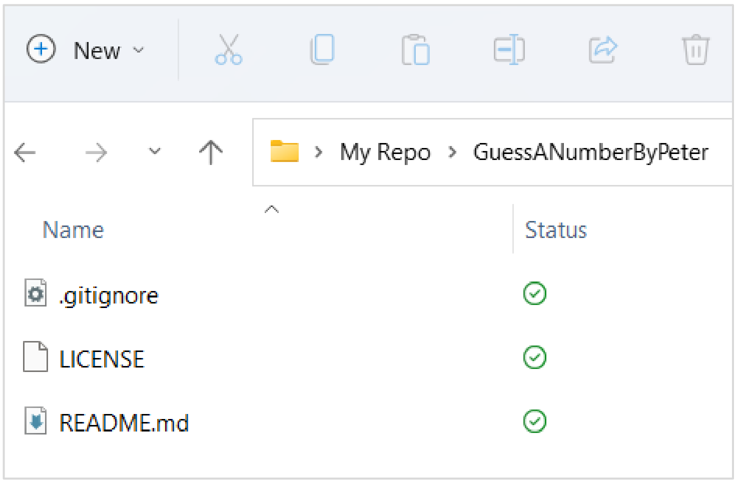


URL: https://github.com/[redacted]/GuessANumberByPete
Directory: C:\Users[redacted]
Depth 1 · Recursive · Clone into Bare Repo · No Checkout
Branch · Origin Name
From SVN Repository: Trunk: trunk · Tags: tags · Branch: branches · From: 0 · Username:

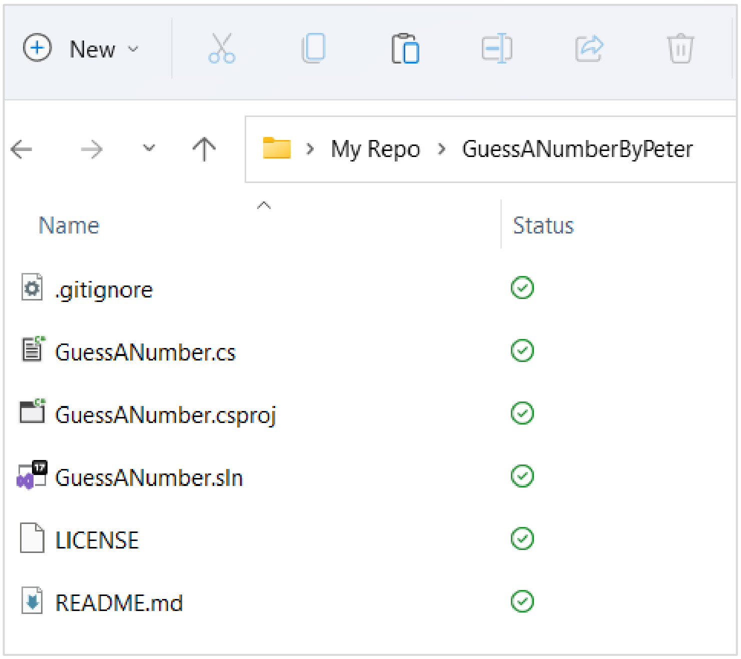
Your files from your GitHub repo will be downloaded to a **sub-folder** called as your project in GitHub, "**GuessANumberByPeter**" in our case.



When we open the cloned **repository sub-folder**, it should look like this:

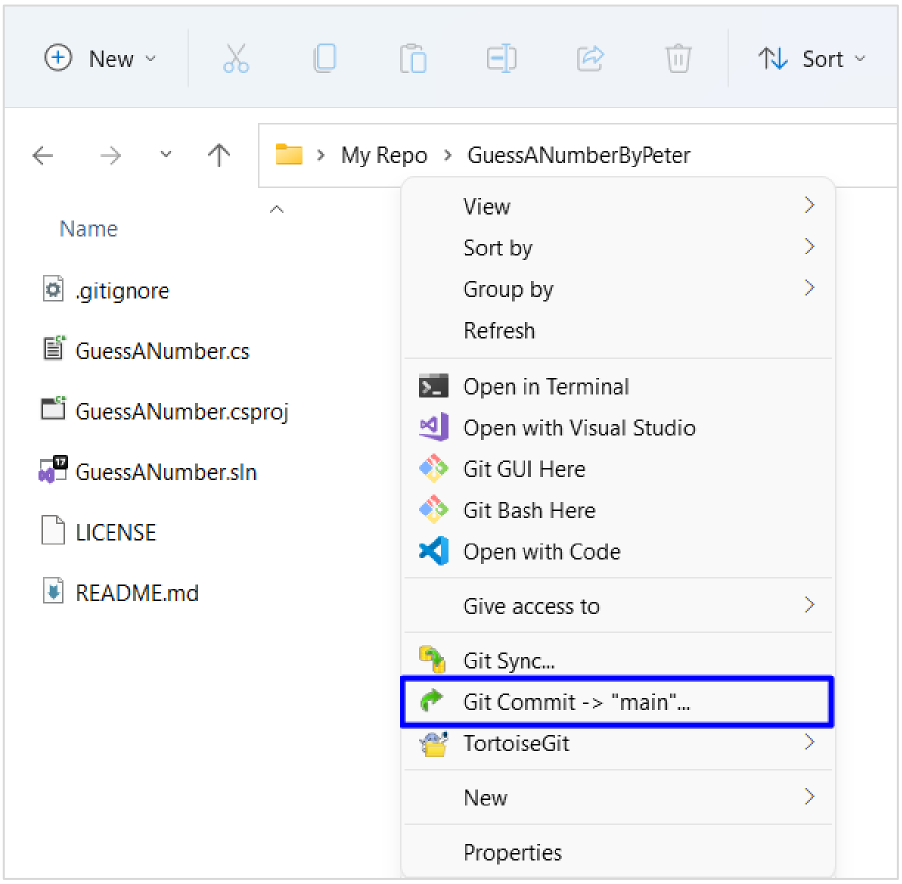


The next thing to do is to **add** the **project** to our **cloned repository**. You can move your C# source code files from your **old project folder** to your **new repo folder**. You can use "**Cut & Paste**". It should look like this:

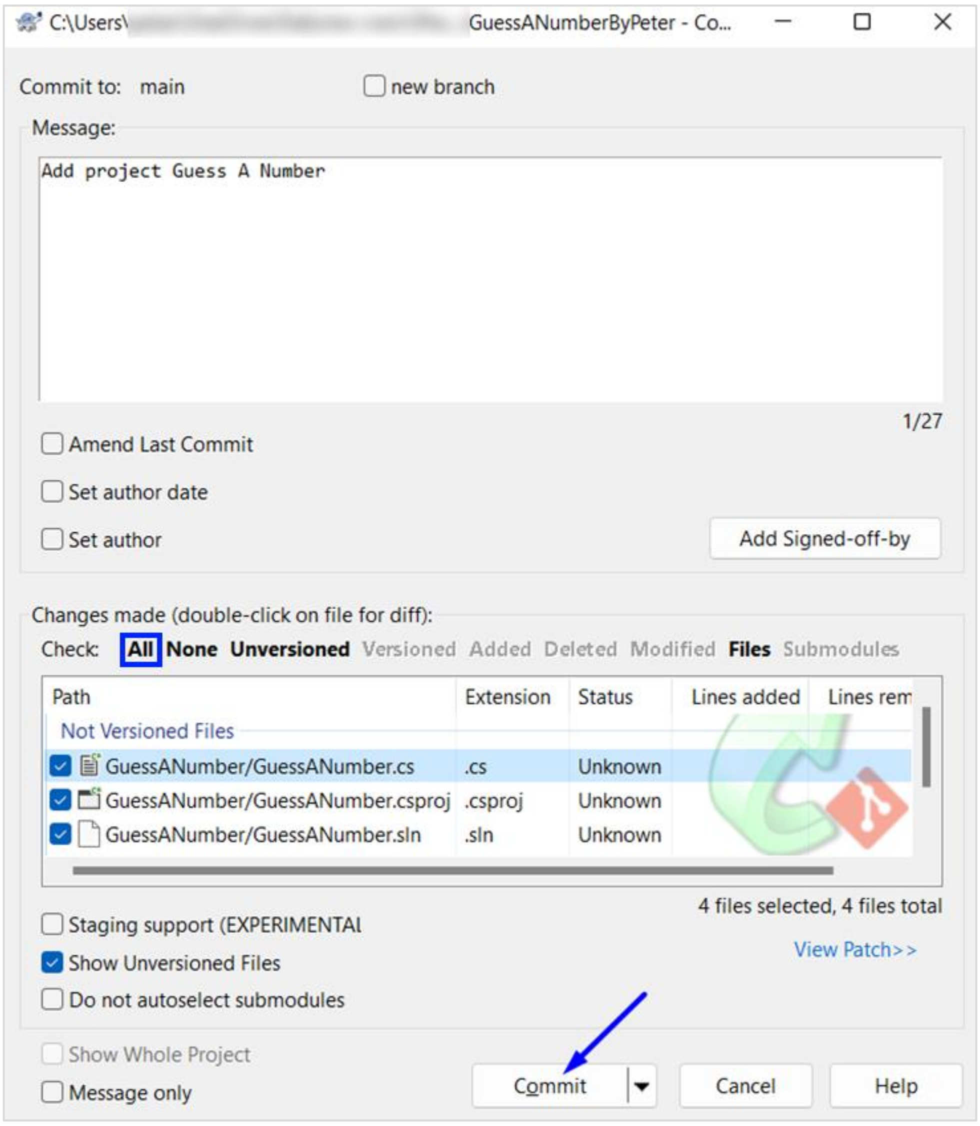


Now to **upload** our changes from our working project folder to GitHub.

We can use TortoiseGit's **`[Git Commit…]`**. Go to your project's folder, **right-click** on blank space anywhere in the folder and click **`[Git Commit -> "main"…]`**.



Add an **appropriate** message and click **`[Add]`** so you don't miss any files, finally click **`[Commit]`**.



Commit to: main ☐ new branch

Message:

```
Add project Guess A Number
```

1/27

☐ Amend Last Commit
☐ Set author date
☐ Set author

Changes made (double-click on file for diff):

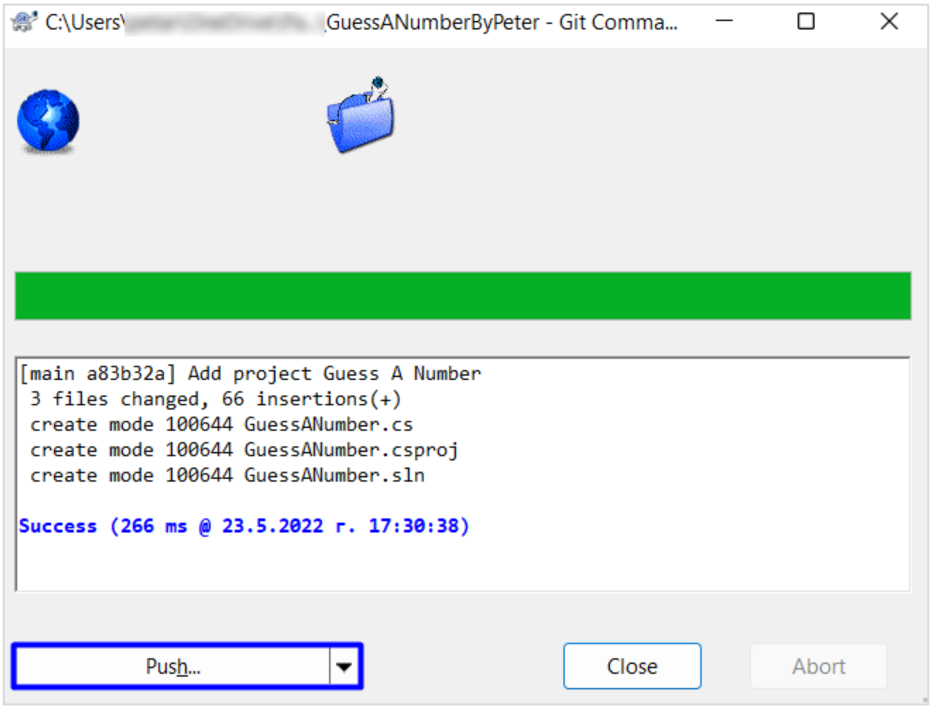
Check: **All** **None** **Unversioned** Versioned Added Deleted Modified **Files** Submodules

Path	Extension	Status	Lines added	Lines rem
Not Versioned Files				
✓ GuessANumber/GuessANumber.cs	.cs	Unknown		
✓ GuessANumber/GuessANumber.csproj	.csproj	Unknown		
✓ GuessANumber/GuessANumber.sln	.sln	Unknown		

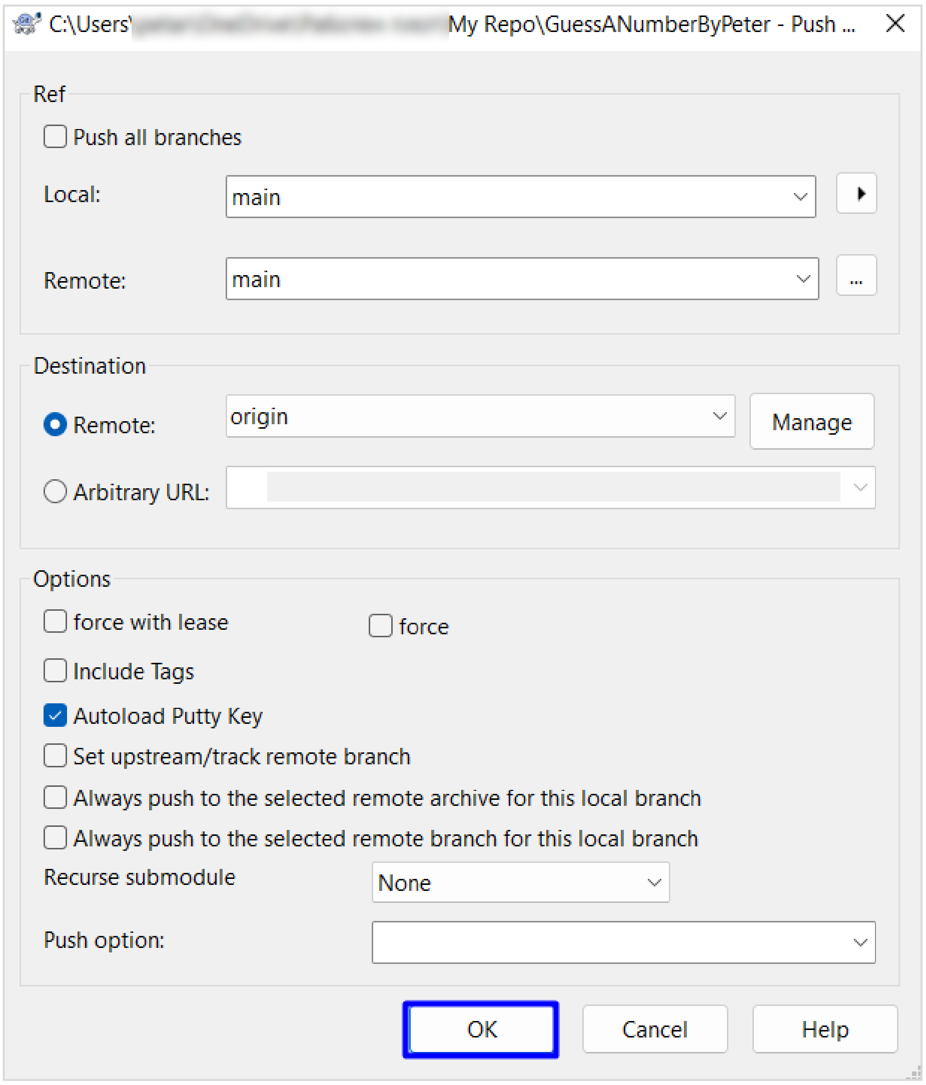
4 files selected, 4 files total

☐ Staging support (EXPERIMENTAL
✓ Show Unversioned Files
☐ Do not autoselect submodules
☐ Show Whole Project
☐ Message only

After that click **`[Push]`**and then **`[OK]`**:



```
[main a83b32a] Add project Guess A Number
 3 files changed, 66 insertions(+)
 create mode 100644 GuessANumber.cs
 create mode 100644 GuessANumber.csproj
 create mode 100644 GuessANumber.sln

Success (266 ms @ 23.5.2022 г. 17:30:38)
```


C:\Users\[redacted]My Repo\GuessANumberByPeter - Push ...

Ref

- [ ] Push all branches

Local: main

Remote: main

Destination

- [✓ Remote:] origin
- Arbitrary URL:

Options

- [ ] force with lease
- [ ] force
- [ ] Include Tags
- [✓] Autoload Putty Key
- [ ] Set upstream/track remote branch
- [ ] Always push to the selected remote archive for this local branch
- [ ] Always push to the selected remote branch for this local branch

Recurse submodule: None

Push option:

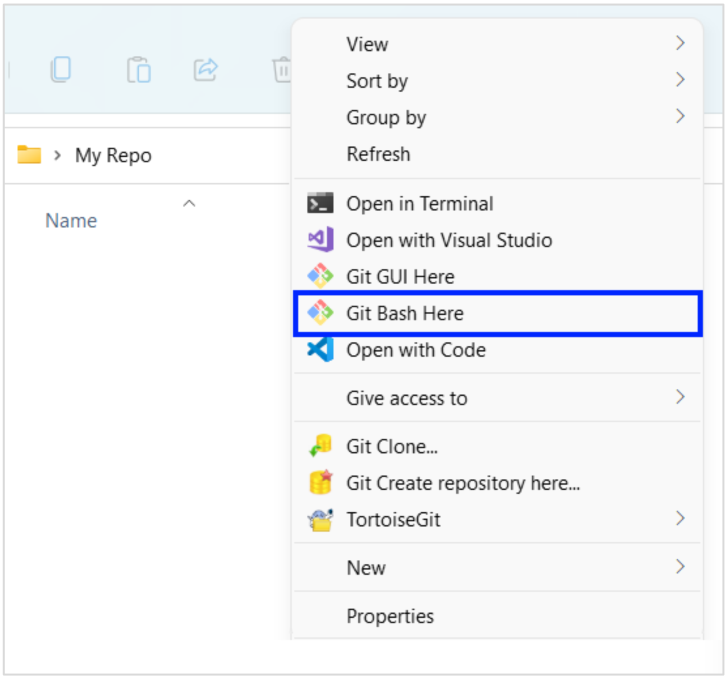
This is all you need to **upload your project source code** to your **GitHub repository** using **`TortoiseGit`**.

Use Git Bash (Option 2)

As **alternative to the previous step**, if you don't have "**TortoiseGit**", you could use the "**Git Bash**" command line tool to upload your project to your **`GitHub`** repo.

First, if you don't have **Git** on your **computer**, you should **install it** from https://git-scm.com/downloads.

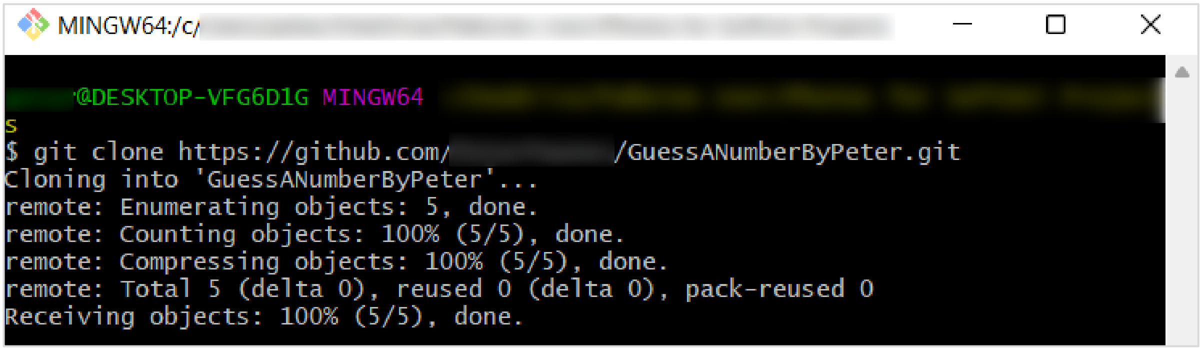
Go to the desired **directory**, right click on blank space **anywhere** in the folder, select "**Git Bash Here**" to open the Git command line console. If the "**Git Bash Here**" menu is missing, you should first install Git.



Type **"`git clone`"** command followed by the link of your **repository**:

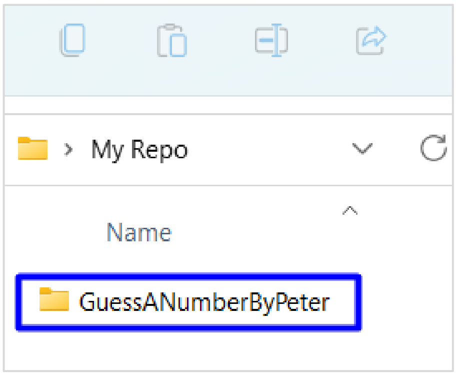
```
git clone
```

This command is for cloning with **Git Bash**, paste your **repository URL** after the command.

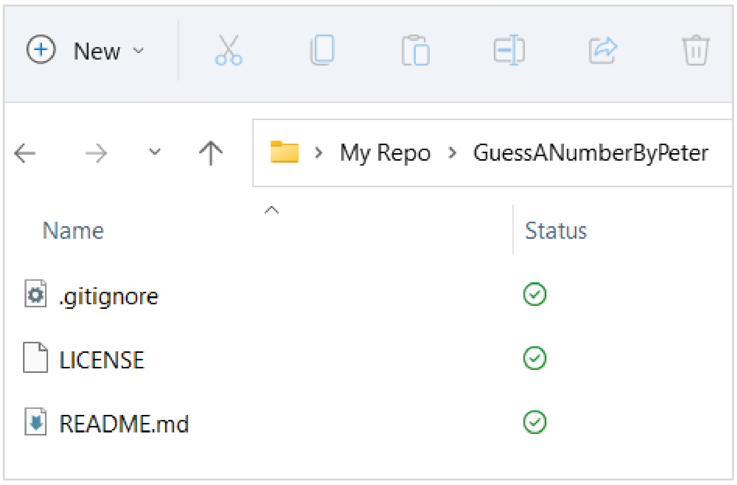


```
MINGW64:/c/[redacted]
[redacted]@DESKTOP-VFG6D1G MINGW64 [redacted]
$
$ git clone https://github.com/[redacted]/GuessANumberByPeter.git
Cloning into 'GuessANumberByPeter'...
remote: Enumerating objects: 5, done.
remote: Counting objects: 100% (5/5), done.
remote: Compressing objects: 100% (5/5), done.
remote: Total 5 (delta 0), reused 0 (delta 0), pack-reused 0
Receiving objects: 100% (5/5), done.
```

Your files from your GitHub repo will be downloaded to a **sub-folder** called as your project in GitHub, "**GuessANumberByPeter**" in our case.

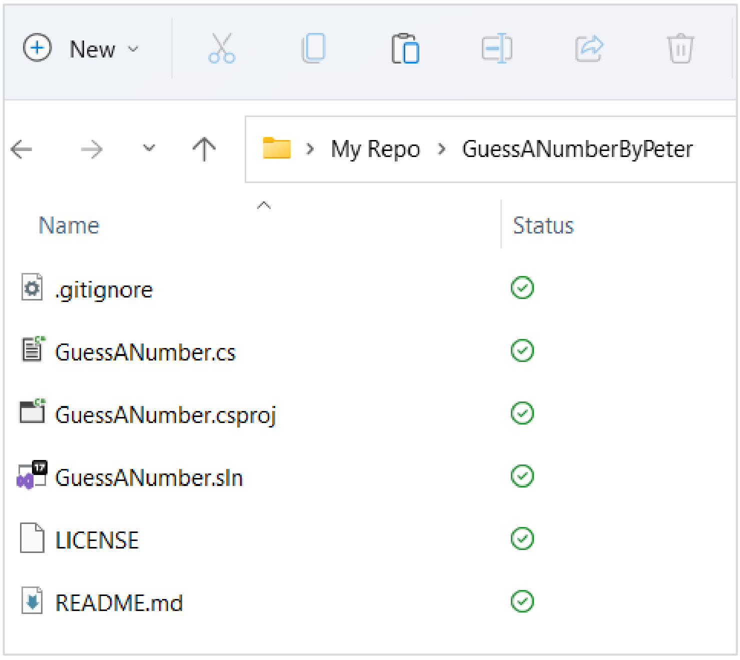


When we open the cloned **repository sub-folder**, it should look like this:



```
.gitignore       ✓
LICENSE          ✓
README.md        ✓
```

Next thing to do is to **add** your **project files** into your **cloned repository folder**. It should look like this:



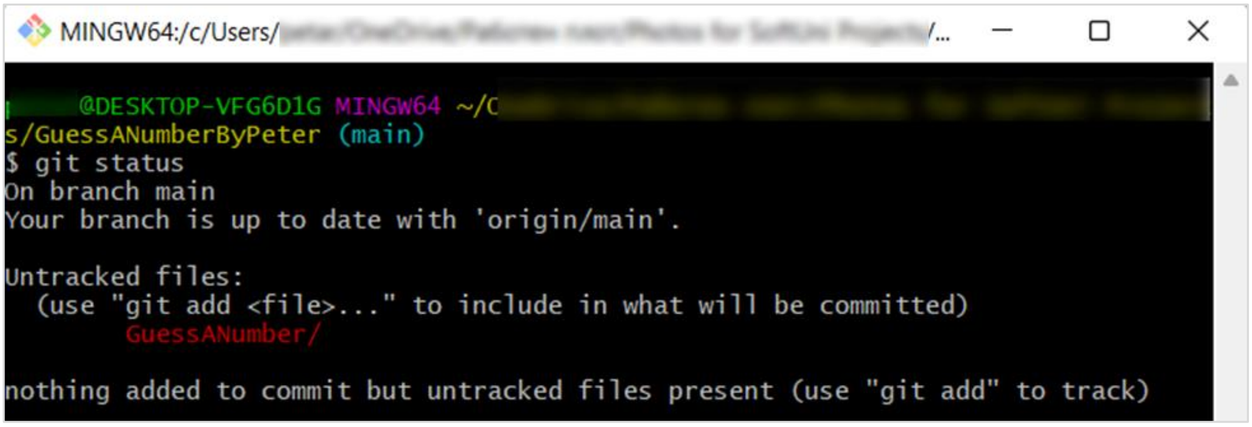
```
.gitignore              ✓
GuessANumber.cs         ✓
GuessANumber.csproj     ✓
GuessANumber.sln        ✓
LICENSE                 ✓
README.md               ✓
```

Now we are ready to upload our changes from "**Git Bash clone**". Go to the desired **folder**, right click on blank space anywhere in the folder, select "**`Git Bash Here`**" and run the following **commands**.

Type the following command:

```
git status
```

The **git status** command displays the state of the working directory and the **staging area**.



```
MINGW64:/c/Users/[redacted]/...
[redacted]@DESKTOP-VFG6D1G MINGW64 ~/C[redacted]
s/GuessANumberByPeter (main)
$ git status
On branch main
Your branch is up to date with 'origin/main'.

Untracked files:
  (use "git add <file>..." to include in what will be committed)
        GuessANumber/

nothing added to commit but untracked files present (use "git add" to track)
```

Now type:

```
git add .
```

This command **adds** all modified files.

Next type:

```
git commit -m "Your message here"
```

This command **commits** your changes. We also should **add** an appropriate **message**.

Second to the last type.

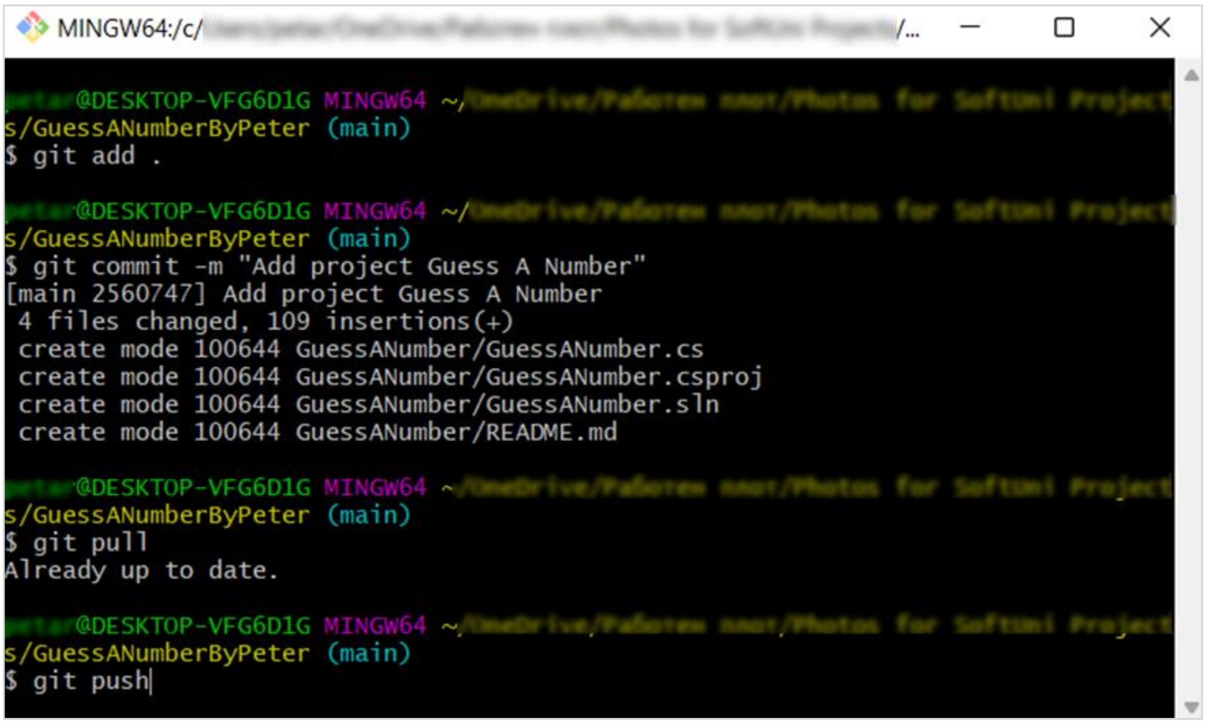
```
git pull
```

This command **updates** your local **repository**.

Now the last thing that we should do is to **push** our changes by using the command:

```
git push
```

This command **pushes** your changes to our local **repository**.



```
MINGW64:/c/[redacted]/...
[redacted]@DESKTOP-VFG6D1G MINGW64 ~/[redacted]
s/GuessANumberByPeter (main)
$ git add .

[redacted]@DESKTOP-VFG6D1G MINGW64 ~/[redacted]
s/GuessANumberByPeter (main)
$ git commit -m "Add project Guess A Number"
[main 2560747] Add project Guess A Number
 4 files changed, 109 insertions(+)
 create mode 100644 GuessANumber/GuessANumber.cs
 create mode 100644 GuessANumber/GuessANumber.csproj
 create mode 100644 GuessANumber/GuessANumber.sln
 create mode 100644 GuessANumber/README.md

[redacted]@DESKTOP-VFG6D1G MINGW64 ~/[redacted]
s/GuessANumberByPeter (main)
$ git pull
Already up to date.

[redacted]@DESKTOP-VFG6D1G MINGW64 ~/[redacted]
s/GuessANumberByPeter (main)
$ git push
```

This is all you need to **update** your **repository** with **Git Bash**.

A little more information about it here: https://git-scm.com/about.

4. * Modify the Code, Write Your Own Features

Now, it's time to **play with the code** and **modify it**.

⚠	This is your own project. **Be unique**. Don't be a copy-paster! • Implement your **own features**. • **Implement the code yourself**, using your own coding style, code formatting, comments, etc.

	• Make the project **more interesting**. Learn by playing with the code and adding your own changes.

Below are a few **ideas** of what you can implement or modify as an addition to your code.

Add Difficulty

You can add logic for difficulty, so the player can have **only a few tries** to guess the number.

Restart the Game

You can automatically **restart the game** after it is finished (or ask the player to play again).

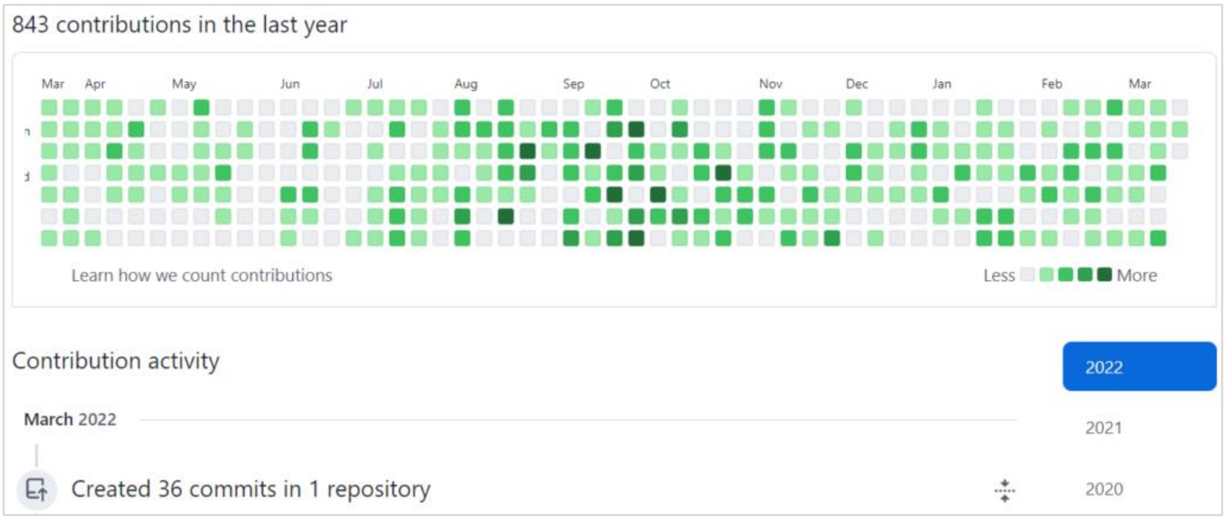
Additional Ideas

- You can **add levels** so every time when the player guesses the number, the range between the minimum and maximum number gets bigger e. g. **Level 1 (1 - 100)**, **Level 2 (1-200)** etc.
- You can add anything else to your code, based on your own ideas?

Commit to GitHub

Now **commit and push your code changes** to your GitHub repo!

It is very important to **commit frequently** your code to GitHub. This way you create a **rich commit history** for your project and your GitHub contribution graph is growing:


843 contributions in the last year

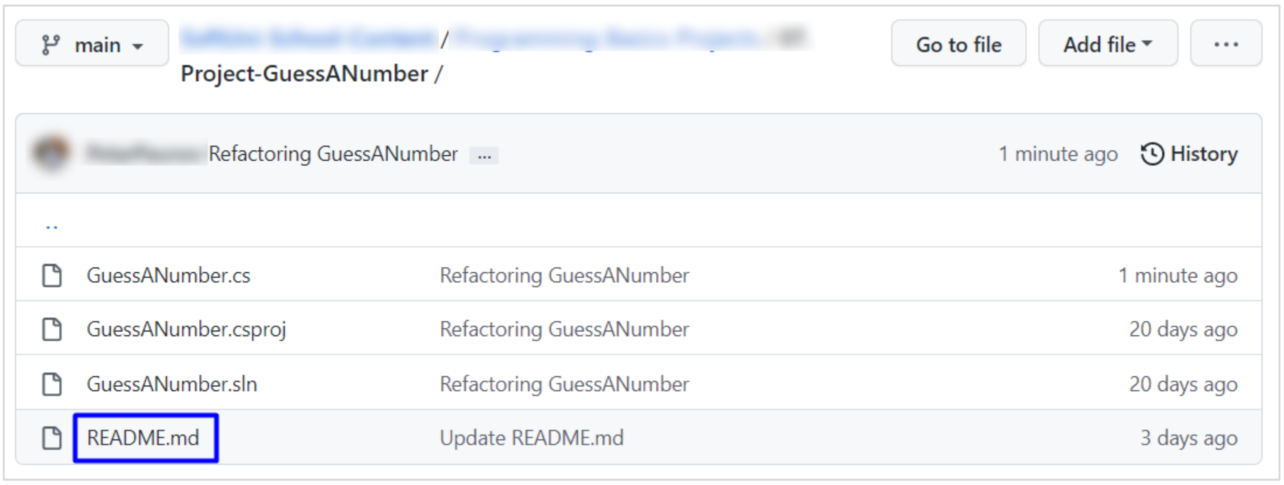
Contribution activity

March 2022

Created 36 commits in 1 repository

5. Create a README.md File

It's highly recommended to provide documentation as part of your project in GitHub to describe what the project is doing. So, let's make one for this **project**. Let's start by editing the **`README.md`** file from our repo at GitHub:



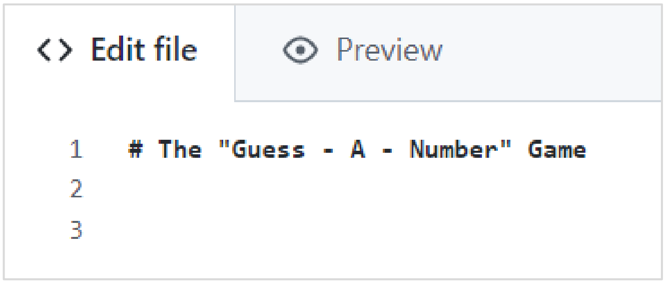
main [redacted] / [redacted] / Project-GuessANumber /

Go to file | Add file | ...

[redacted] Refactoring GuessANumber ... 1 minute ago History

..		
GuessANumber.cs	Refactoring GuessANumber	1 minute ago
GuessANumber.csproj	Refactoring GuessANumber	20 days ago
GuessANumber.sln	Refactoring GuessANumber	20 days ago
README.md	Update README.md	3 days ago

Add a project name. Use **"#"** in front of the text to indicate the **title**:



```
# The "Guess - A - Number" Game


```

You can **view** the current progress by pressing the **[Preview]** button:

Documentation Sections

Add **information** about your project in your **README.md** file: project goals, technologies used, screenshots, live demo, etc. Typically, you should have the following **sections**:

- **Project title** (should answer the question "What's inside this project?")
- **Project goals** (what problem we solve, e. g. we implement a certain game)
- **Solution** (should describe how we solve the problem → algorithms, technologies, libraries, frameworks, tools, etc.)
- **Source code link** (give a direct link to your source code)
- **Screenshots** (add screenshots from your project in different scenarios of its usage)
- **Live demo** (add a one-click live demo of your code)

Use Markdown

Note that the GitHub **README.md** file is written in the **Markdown language**. Markdown combines text and special formatting tags to describe formatted text documents.

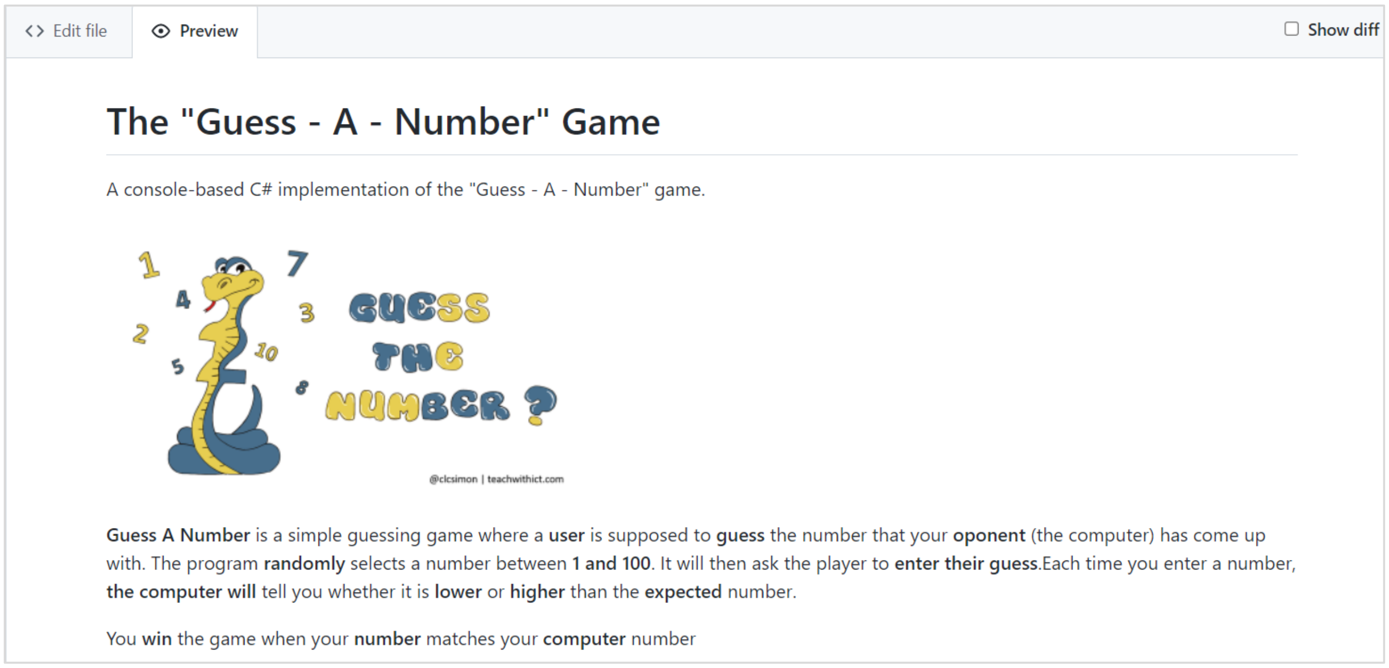
You can learn more about **Markdown** here: https://docs.github.com/en/get-started/writing-on-github/getting-started-with-writing-and-formatting-on-github/basic-writing-and-formatting-syntax.

Project Goals

Start your documentation by describing your **project goals**. What problem does your project solve?

Sample Documentation

This is an **example** of how you can document your project. Don't copy and paste it!



The "Guess - A - Number" Game

A console-based C# implementation of the "Guess - A - Number" game.

Guess A Number is a simple guessing game where a **user** is supposed to **guess** the number that your **oponent** (the computer) has come up with. The program **randomly** selects a number between **1 and 100**. It will then ask the player to **enter their guess**.Each time you enter a number, **the computer will** tell you whether it is **lower** or **higher** than the **expected** number.

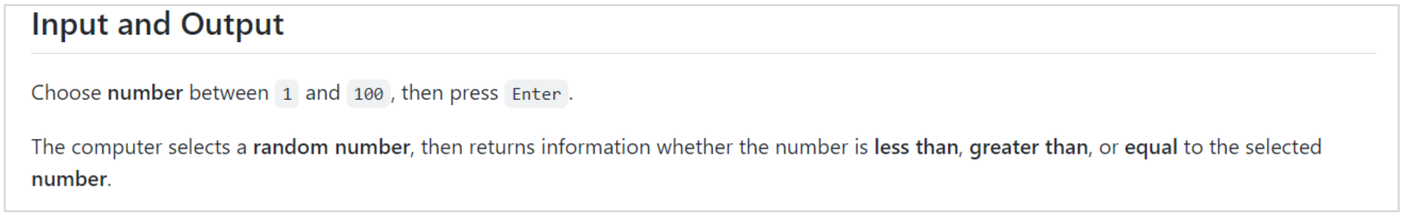
You **win** the game when your **number** matches your **computer** number

⚠	**Write the project documentation yourself**. Don't copy and paste it! This is your **unique GitHub profile** and your own unique project. **Be different** from others.

Find an **appropriate image** and add it. You can add **images** as follows:

```
<img alt="Image" width="200px" src="[redacted]" />
```

You can add information about the **inputs** and **outputs** of the project:



Input and Output

Choose **number** between `1` and `100`, then press `Enter`.

The computer selects a **random number**, then returns information whether the number is **less than**, **greater than**, or **equal** to the selected **number**.

Your Solution

Describe how you **solve the problem**: algorithms, technologies, libraries, frameworks, tools, etc.

Link to the Source Code

Add a **link** to your **source code** as follows:

```
[Source Code](GuessANumber.cs)
```

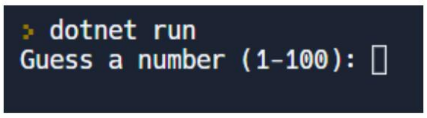
Screenshots

Add **screenshots** of your project:

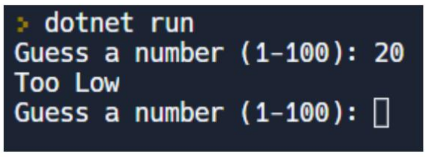
1. **Take a screenshot** with your favourite tool (e.g. the Snipping Tool in Windows).
2. **Paste** the screenshot in the GitHub Markdown editor, using **`[Ctrl+V]`**:

Example screenshots for the "**Guess a Number**" game:

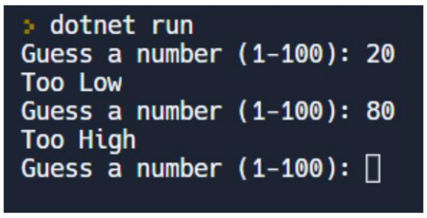
Screenshots



```
dotnet run
Guess a number (1-100): 
```



```
dotnet run
Guess a number (1-100): 20
Too Low
Guess a number (1-100): 
```

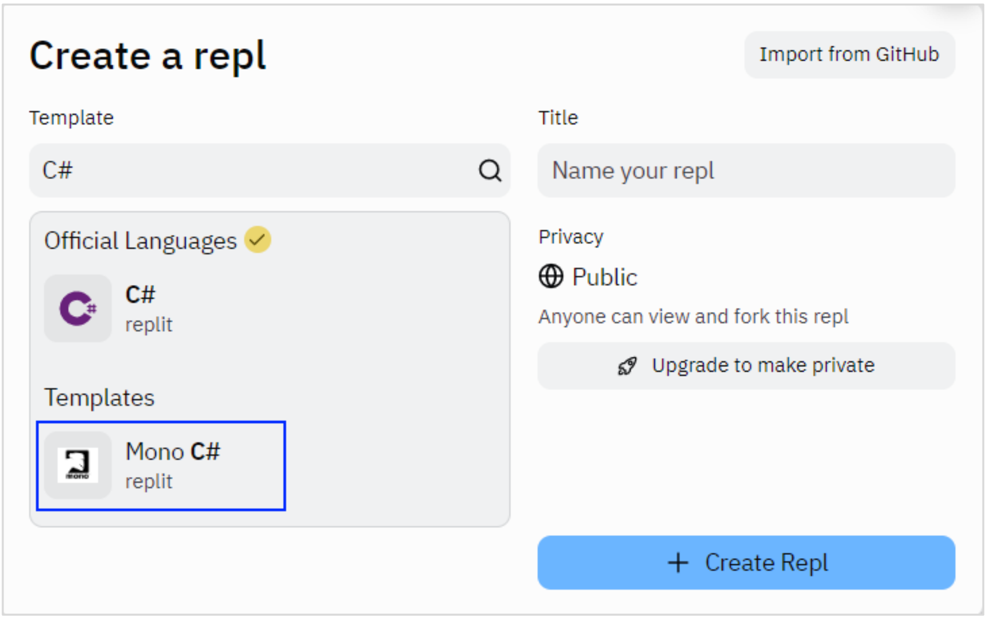


```
dotnet run
Guess a number (1-100): 20
Too Low
Guess a number (1-100): 80
Too High
Guess a number (1-100): 
```

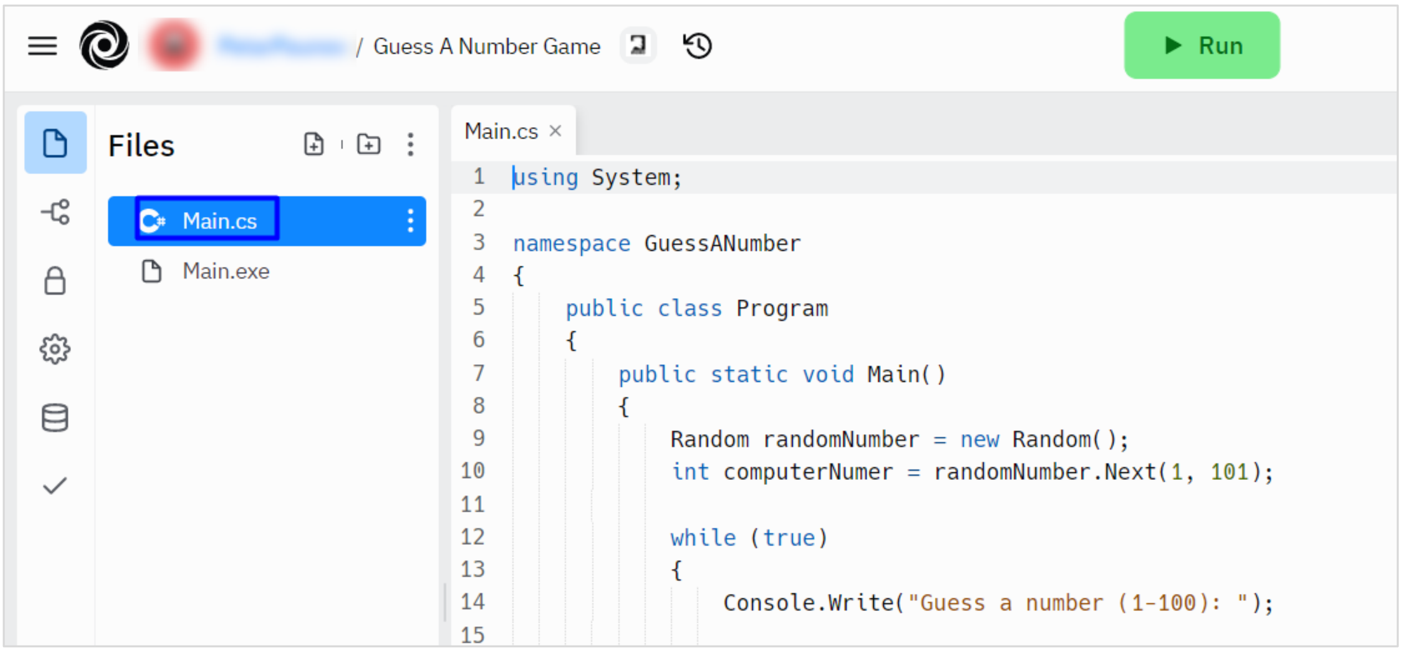
6. Upload Your App to Replit

You already should have a **Replit** profile. Now let's add our **project** there so we can share it with our **friends** and add it to our **GitHub** profile. You already should know how to do that.

Open the **menu** in the upper **left corner**. Click "**Create**", then select the **language** in which your project is **written**, select a name, and **create** the project. If your project is in **C#**, choose "**Mono C#**". In **`Replit`** the C# projects work faster with Mono, than with .NET 6.

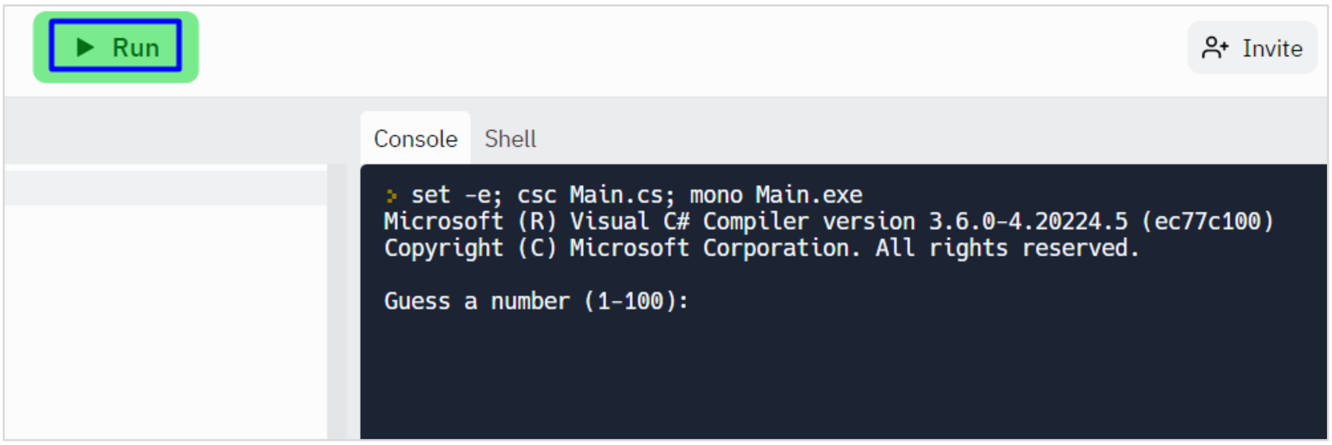


Add your code in "**`Main.cs`**" file.



```
using System;

namespace GuessANumber
{
    public class Program
    {
        public static void Main()
        {
            Random randomNumber = new Random();
            int computerNumer = randomNumber.Next(1, 101);

            while (true)
            {
                Console.Write("Guess a number (1-100): ");

```

Click [Run] and enjoy your console application directly in the Web browser:



```
> set -e; csc Main.cs; mono Main.exe
Microsoft (R) Visual C# Compiler version 3.6.0-4.20224.5 (ec77c100)
Copyright (C) Microsoft Corporation. All rights reserved.

Guess a number (1-100):
```

You can now **share** your app with your friends.

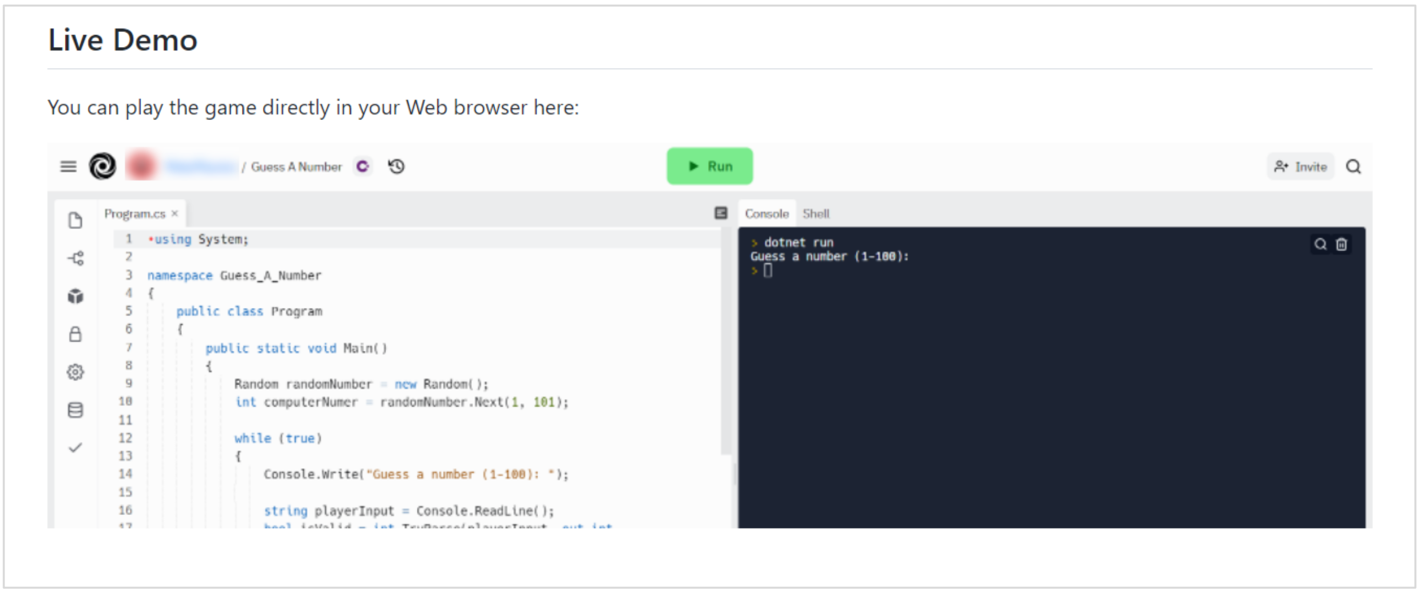
7. Add Replit Link to Your README.md

Now add a "**one-click live demo**" of your project from your GitHub project documentation. You can do it as follows:

```
## Live Demo

You can play the game directly in your Web browser here:

[<img alt="Play Button" src="https://user-images.githubusercontent.com/85368212/167706022-5128fd5d-d315-4eb9-9001-3bf9008cd91e.png" />]
(https://replit.com/[redacted]/Guess-A-Number-Game#Main.cs)
```

You can take a **screenshot** from Replit.com and **paste it** into the GitHub documentation editor directly with **[Ctrl+V]**.

This is what it should look like after the changes in your **README.md** documentation:



Live Demo

You can play the game directly in your Web browser here:

```
using System;

namespace Guess_A_Number
{
    public class Program
    {
        public static void Main()
        {
            Random randomNumber = new Random();
            int computerNumer = randomNumber.Next(1, 101);

            while (true)
            {
                Console.Write("Guess a number (1-100): ");

                string playerInput = Console.ReadLine();
```

```
> dotnet run
Guess a number (1-100):
> 
```

Now we have completed our **second console game** and we have our second **project** in our **`GitHub`** portfolio.