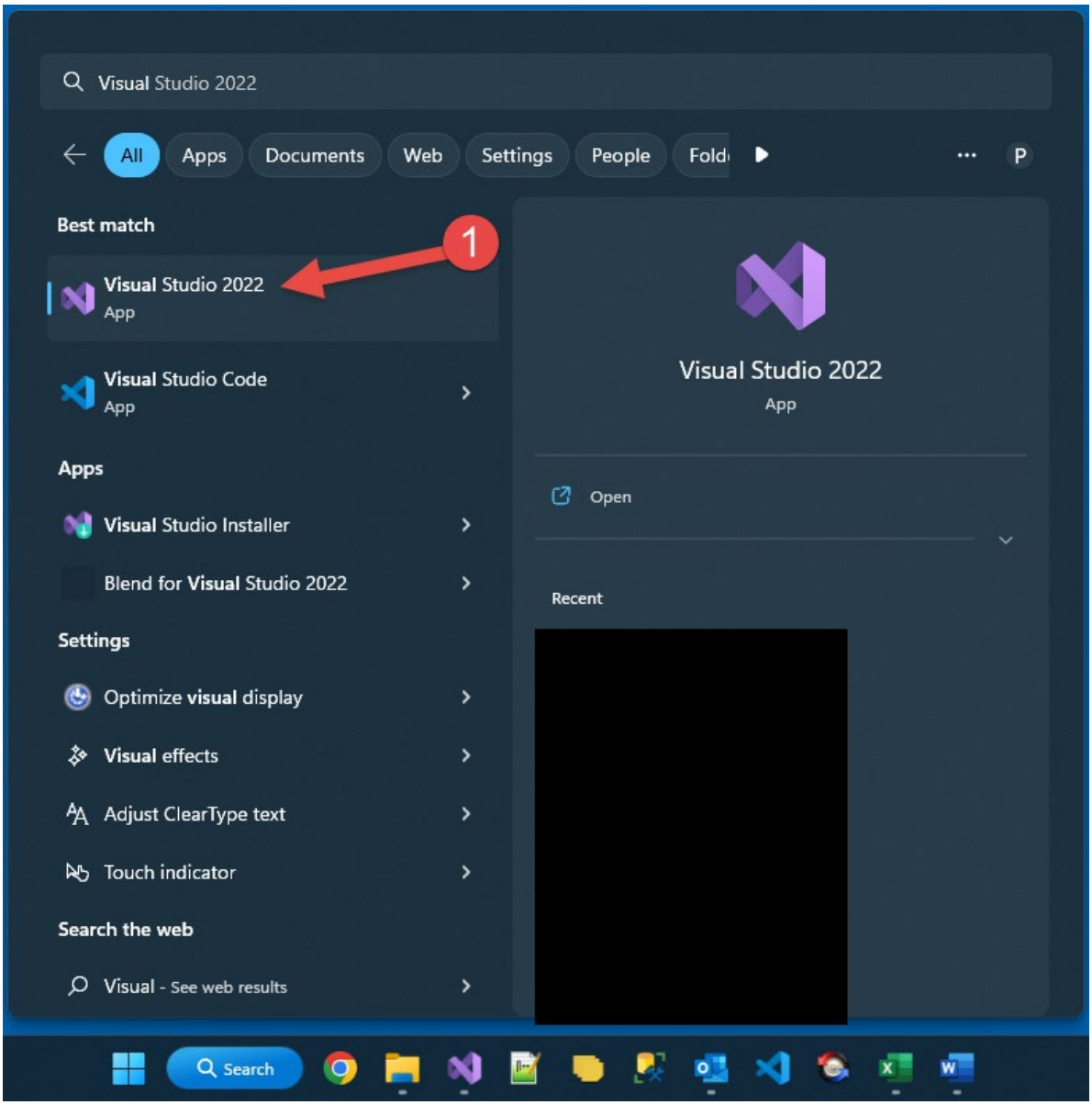
Properties Lab

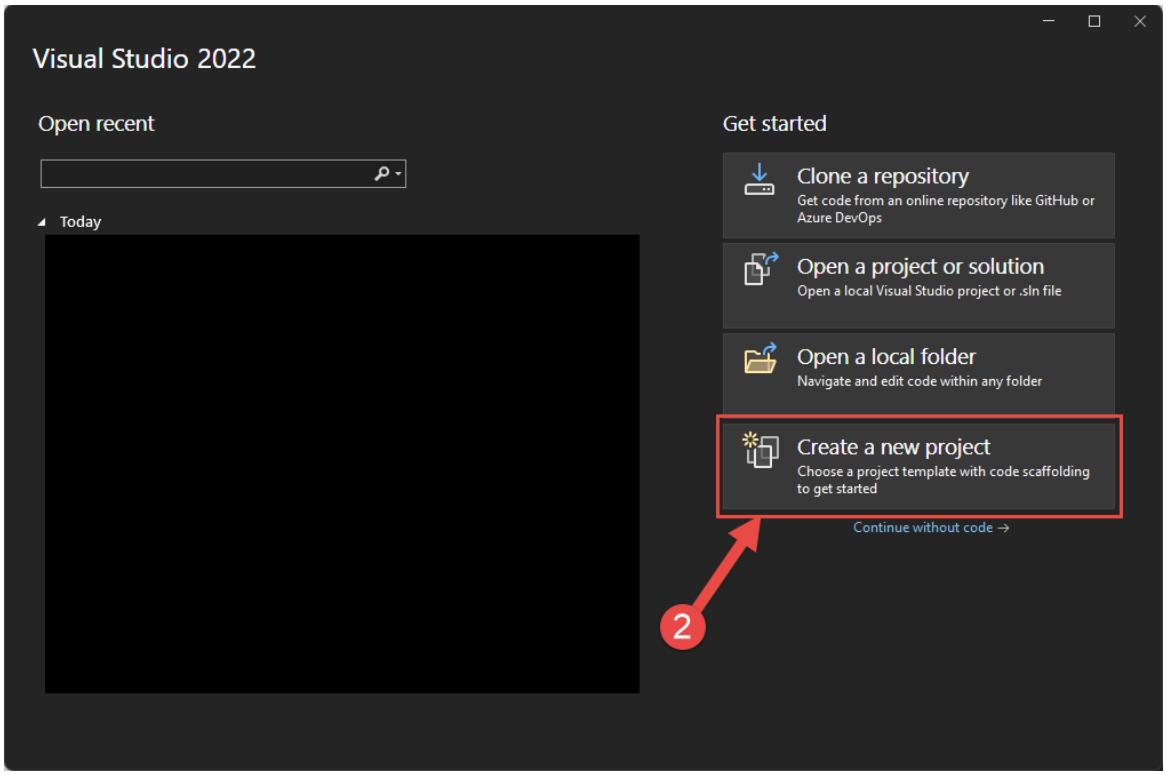
Perform these labs on your own computer using Visual Studio 2022 to ensure you understand the lessons presented in the corresponding videos and lectures.

Lab 1: Create Application in Visual Studio

1. Open **Visual Studio 2022**.

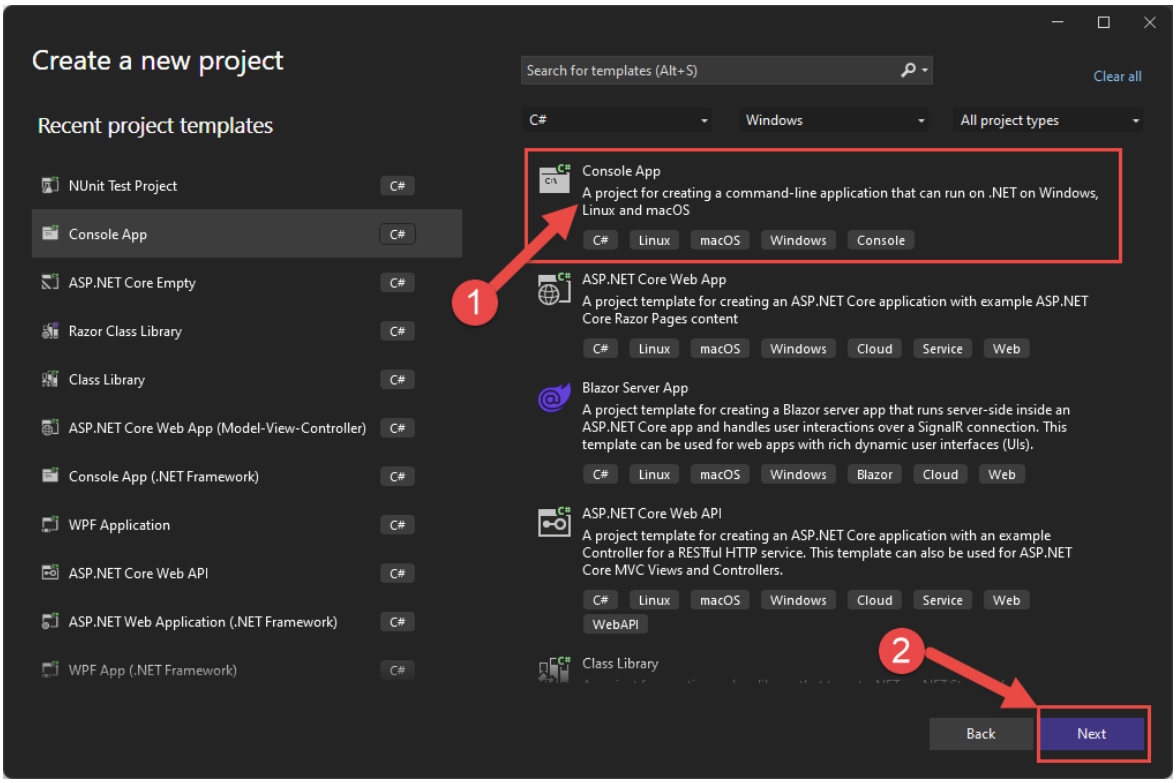


2. Click on the **Create a new project** option.



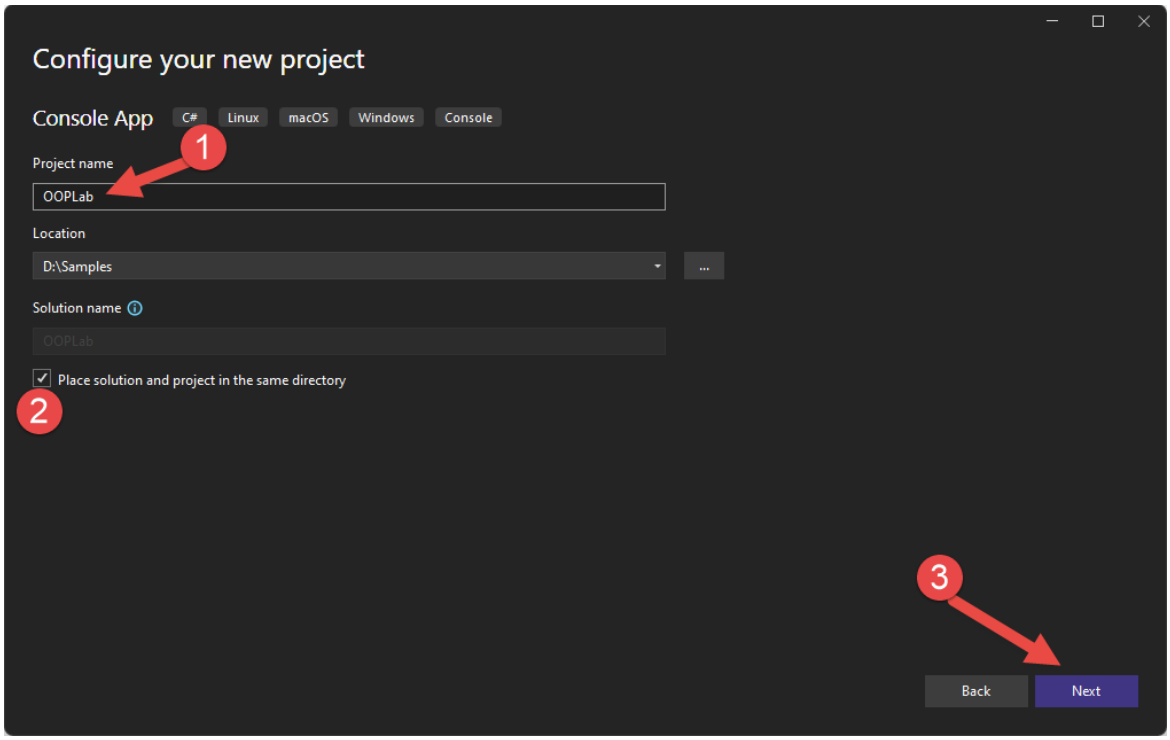
Select Console Application for .NET 6

1. Choose the **Console App "A project for creating a command-line application that can run on Windows, Linux and macOS". DO NOT** choose the project template for the .NET Framework.
2. Click the **Next** button.



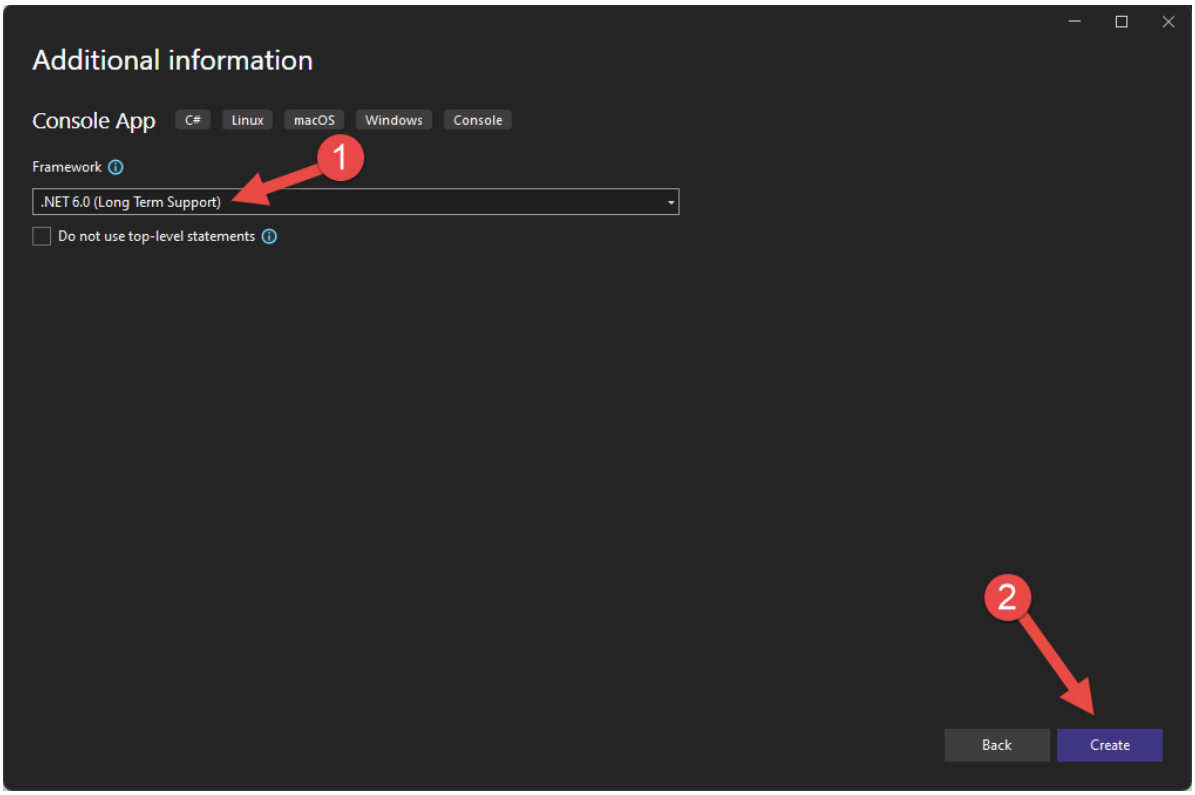
Set the Project Name

1. Set the Project name to **OOPLab**.
2. Check the Place solution and project in the same directory.
3. Click the **Next** button.

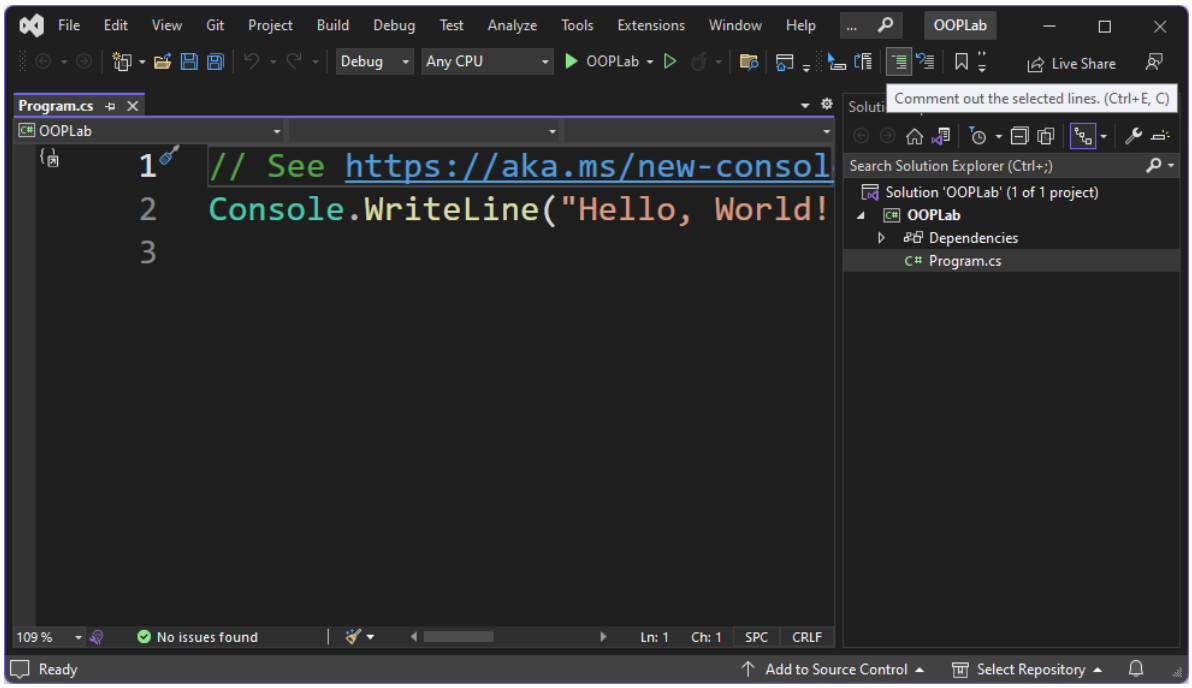


Select the .NET Version

1. Ensure the Framework is set to **.NET 6.0 (Long-term support) or .NET 8.0 (long-term support)**
2. Click the Create button



Your project should now look similar to the following.



Lab 2: Create Class and a Simple Property

Right mouse-click on the **OOPLab** project and select **Add | Class…**

Set the name to **Customer** and click the **Add** button.

Modify the class to look like the following:

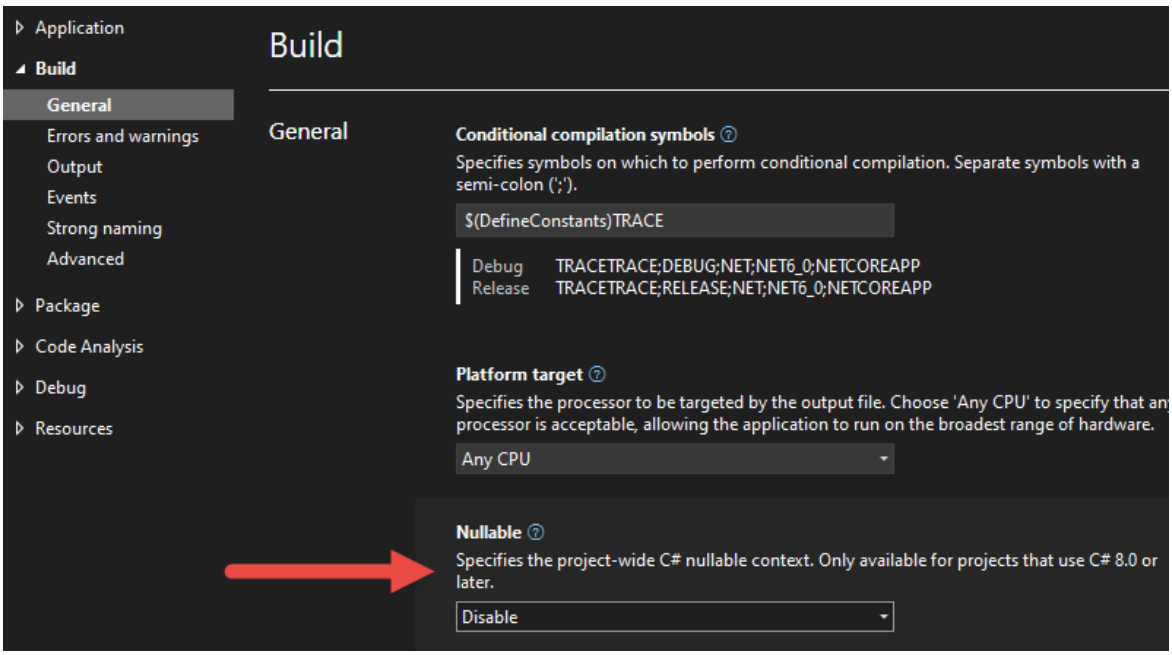
```
namespace OOPLab;

public class Customer {

}
```

Add three properties to hold customer data

```
public int CustomerId { get; set; }
public string FirstName { get; set; }
public string LastName { get; set; }
```

Right mouse-click on the OOPLab project and select **Properties** from the menu

Click on the **Build | General** tab and change the **Nullable** setting to **Disable** as shown in the following screen shot.



Open the **Program.cs** file and add the following code

```
using OOPLab;

Customer entity = new();

entity.CustomerId = 1;
entity.FirstName = "John";
entity.LastName = "Smith";

Console.Write(entity.CustomerId);
Console.Write(" - ");
Console.Write(entity.FirstName);
Console.Write(" ");
Console.WriteLine(entity.LastName);
```

Try it Out

Run the application and the output should look like the following.

```
1 - John Smith
```

Lab 3: Alternate Syntax for Creating Object

Instead of setting each property via the *entity* variable, refactor the declaration of the Customer object to look like the following.

```
Customer entity = new() {
  CustomerId = 1,
  FirstName = "John",
  LastName = "Smith"
};
```

Try it Out

Run the application and the output should be the same as the before.

Lab 4: Create Full Property

Open the **OOPLab** solution and open the **Customer.cs** file.

Add two new full properties to the Customer class

```
private string _CompanyName;

public string CompanyName
{
 get { return _CompanyName; }
 set { _CompanyName = value; }
}

private string _EmailAddress;

public string EmailAddress
{
 get { return _EmailAddress; }
 set { _EmailAddress = value; }
}
```

Try it Out

Open the **Program.cs** file and initialize these two new properties when creating the instance of the Customer class.

```
Customer entity = new() {
  CustomerId = 1,
  FirstName = "John",
  LastName = "Smith",
  CompanyName = "Smith, Inc.",
  EmailAddress = "John.Smith@smithinc.com"
};
```

Add two new Console.WriteLine() statements at the end of the file.

```
Console.WriteLine(entity.CompanyName);
Console.WriteLine(entity.EmailAddress);
```

Run the application and the output should look like the following:

```
1 - John Smith
Smith, Inc.
John.Smith@smithinc.com
```

Lab 5: Create a Read-Only Full Property

Open the **Customer.cs** file and add a new property named **CreditLimit** to the Customer class

```
private decimal _CreditLimit;

public decimal CreditLimit
{
  get { return _CreditLimit; }
}
```

There is nothing to try out currently. You will use this property in later labs.