Creating Classes Lab

Lab 1: Create Class and a Simple Property

Right mouse-click on the project and add a new class named **Product**.

Modify the class to look like the following.

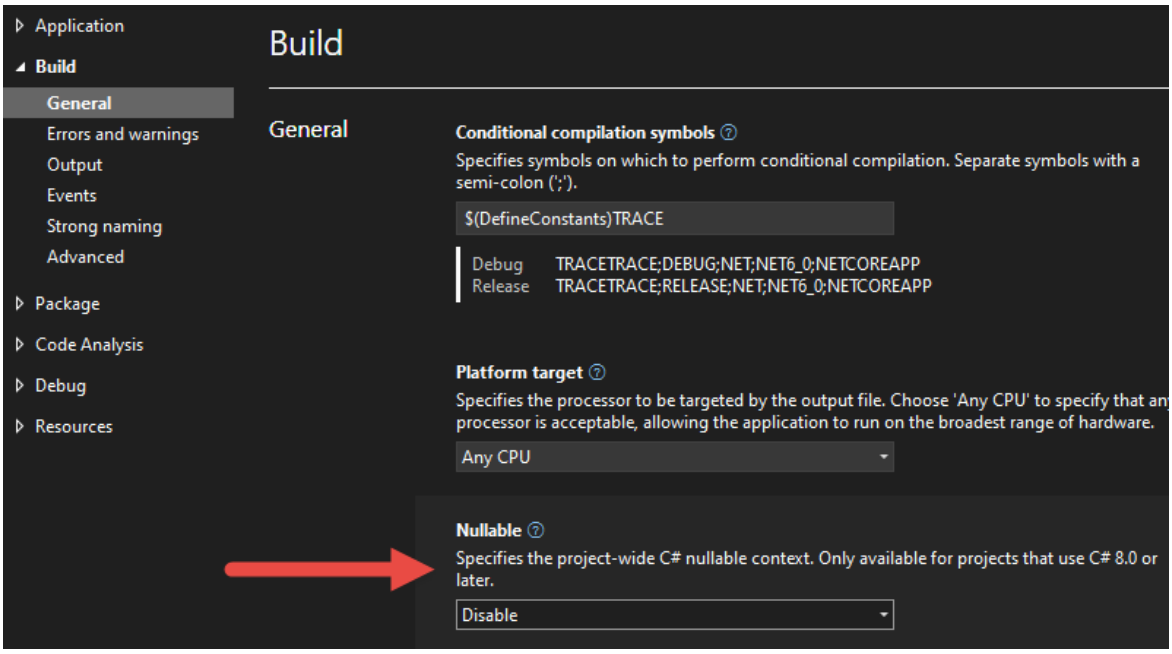
```
namespace CSharpSamples {
  public class Product {

  }
}
```

Add two properties.

```
public int ProductId { get; set; }
public string Name { get; set; }
```

Set the Global nullable setting as shown in the following screen shot



Open the **Program.cs** file and replace the entire contents of the file with the following code.

```
using CSharpSamples;

Product entity = new();

entity.ProductId = 1;
entity.Name = "Bicycle";

Console.Write(entity.ProductId);
Console.Write(" - ");
Console.WriteLine(entity.Name);
```

Try It Out

Run the application and view the output.

Lab 2: Alternate Syntax for Creating Object

Open the **Program.cs** file and replace the entire contents of the file with the following code.

```
using CSharpSamples;

Product entity = new() {
  ProductId = 1,
  Name = "Bicycle"
};

Console.Write(entity.ProductId);
Console.Write(" - ");
Console.WriteLine(entity.Name);
```

Try It Out

Run the application and view the output.

Lab 3: Create Full Property

Open the **Product.cs** file and add two new properties to the Product class

```
private decimal _ListPrice;

public decimal ListPrice
{
  get { return _ListPrice; }
  set { _ListPrice = value; }
}

private decimal _StandardCost;

public decimal StandardCost
{
  get { return _StandardCost; }
  set { _StandardCost = value; }
}
```

Modify the **Program.cs** file to look like the following code.

```
Product entity = new() {
  ProductId = 1,
  Name = "Bicycle",
  StandardCost = 2.99M,
  ListPrice = 6.99M
};

Console.Write(entity.ProductId);
Console.Write(" - ");
Console.WriteLine(entity.Name);
Console.WriteLine("Cost = " + entity.StandardCost.ToString("c"));
Console.WriteLine("Price = " + entity.ListPrice.ToString("c"));
```

Try It Out

Run the application and view the output.

Lab 4: Void Method

Open the **Product.cs** file and add a new property.

```
public decimal Profit { get; set; }
```

Now add a method to this class.

```
private void CalculateProfit() {
  Profit = _ListPrice - StandardCost;
}
```

Modify the setters to call this new method.

Add a new line to the end of the **Program.cs** file.

```
Console.WriteLine("Profit = " + entity.Profit.ToString("c"));
```

Try It Out

Run the application and view the output.

Lab 5: Method to Return a Value

Open the **Product.cs** file and add two new properties.

```
public DateTime SellStartDate { get; set; }
public DateTime SellEndDate { get; set; }
```

Add a new method to calculate the number of days between these two dates.

```
public int GetNumberOfSellDays() {
  return (SellEndDate - SellStartDate).Days;
}
```

Open the **Program.cs** file and modify the declaration of the Product class as shown in the code in bold below.

```
Product entity = new() {
  ProductId = 1,
  Name = "Helmet",
  StandardCost = 2.99M,
  ListPrice = 6.99M,
  SellStartDate = DateTime.Parse("10/1/2022"),
  SellEndDate = DateTime.Parse("12/31/2022")
};
```

Add a new line at the end of the **Program.cs** file.

```
Console.WriteLine("Days to Sell: " + entity.GetNumberOfSellDays());
```

Try It Out

Run the application and view the output.