Handling Nulls

Deborah Kurata
CONSULTANT | SPEAKER | AUTHOR | MVP | GDE

@deborahkurata | blogs.msmvps.com/deborahk/

The absence of a value

Not the same as zero

Not the same as an empty string

Default Values of Class Properties

```
   // Value types
   bool IsInStock;                 // false
   int Count;                      // 0
   decimal Price;                  // 0
   DateTime EffectiveDate;         // 1/1/0001
```

```
   // Reference types
   string Reason;                  // null
   Discount PriceDiscount;         // null
   List<Discount> Discounts;       // null
```

```
Debug.WriteLine(IsInStock);    // false
```

```
Debug.WriteLine(Reason.Length);
Debug.WriteLine(PriceDiscount.PercentOff);
```

“Object reference not set to an instance of an object”

C# application

"I call it my billion-dollar mistake."

Tony Hoare
https://en.wikipedia.org/wiki/Tony_Hoare

Module Overview

Declaring nullable value types

Defending our code from null nullable value types

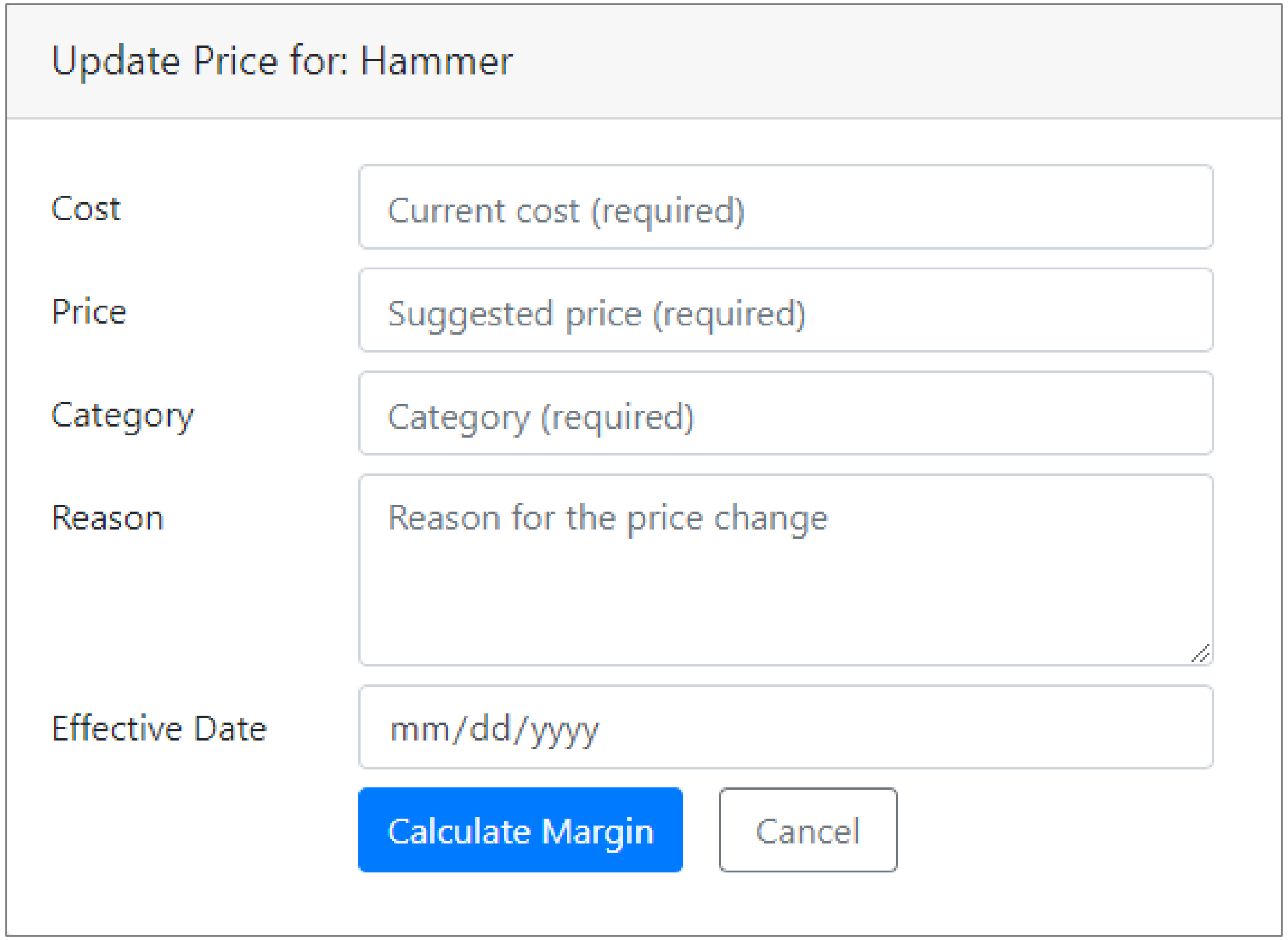
Defending our code from null reference types

Reference type nullability features

Why Use Nulls?

```
// Reference types
string Reason;                 // null
Discount PriceDiscount;        // null
List<Discount> Discounts;      // null
```

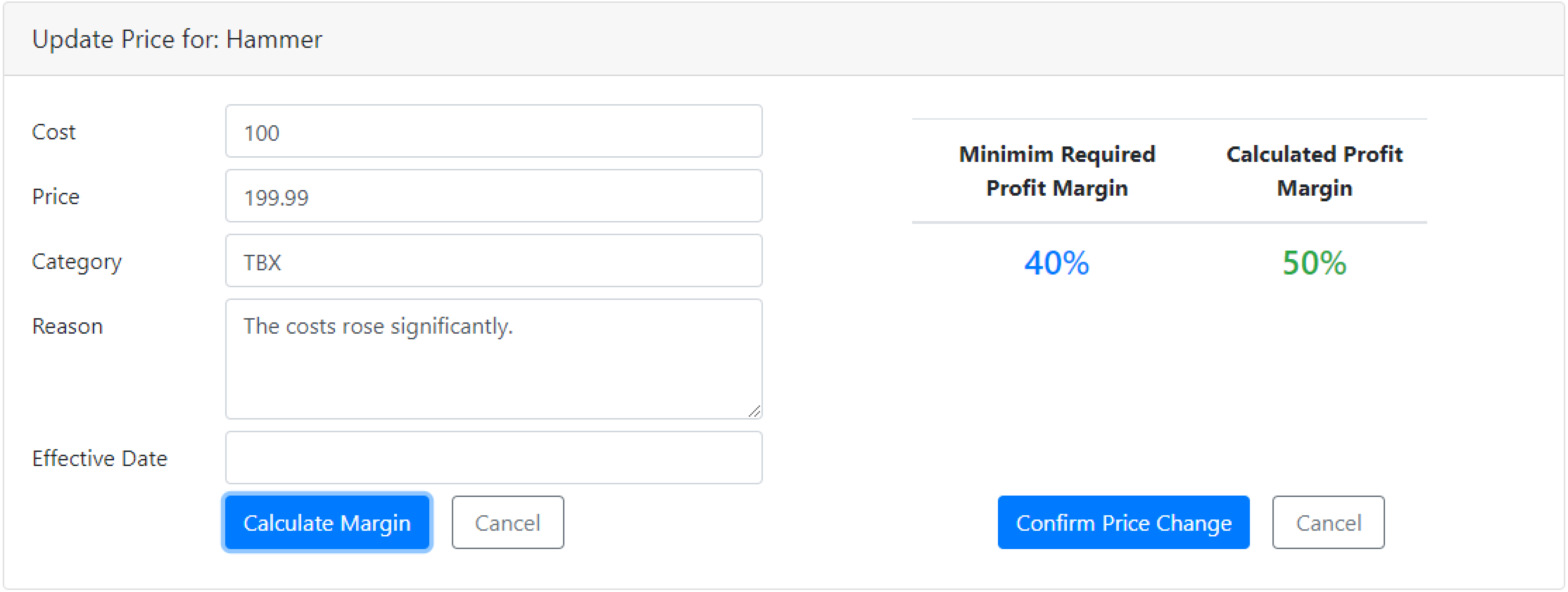
Why Use Nulls?



Update Price for: Hammer

Cost	Current cost (required)
Price	Suggested price (required)
Category	Category (required)
Reason	Reason for the price change
Effective Date	mm/dd/yyyy

Why Use Nulls?



Update Price for: Hammer

Cost	100
Price	199.99
Category	TBX
Reason	The costs rose significantly.
Effective Date	

Minimim Required Profit Margin	Calculated Profit Margin
40%	50%

Nullable Value Types

```
// Value types
bool IsInStock;                // false
int Count;                     // 0
decimal Price;                 // 0
DateTime EffectiveDate;        // 1/1/0001
```

```
// Nullable value types
bool? IsInStock;                // null
int? Count;                     // null
decimal? Price;                 // null
DateTime? EffectiveDate;        // null
```

Demo

Defending our code from null

nullable value types

```
DateTime? EffectiveDate;    // null
```

Demo

Defending our code from null reference types

```
  Discount discount; // null
```

Reference Type Nullability

```
// Value types
bool IsInStock;
int Count;
decimal Price;
DateTime EffectiveDate;
```

```
// Non-nullable reference types
string Reason = "";
Discount PriceDiscount = new Discount();
List<Discount> Discounts = new List<Discount>();
```

```
// Nullable value types
bool? IsInStock;
int? Count;
decimal? Price;
DateTime? EffectiveDate;
```

```
// Nullable reference types
string? Reason;
Discount? PriceDiscount;
List<Discount>? Discounts;
```

Minimum: C# 8, .NET Core 3 OR .NET Standard 2.1

Reference Type Nullability

```
// Non-nullable reference types
string Reason = "";
Discount PriceDiscount =
              new Discount();
List<Discount> Discounts =
              new List<Discount>();
```

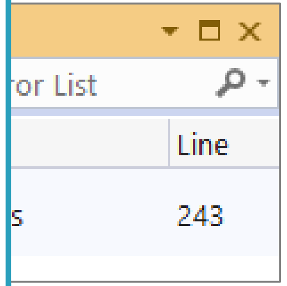
- Reference type variable can not be null
- Must be initialized to a value
- Compiler verified

```
// Nullable reference types
string? Reason;
Discount? PriceDiscount;
List<Discount>? Discounts;
```

- Reference type variable may be null
- Default of null
- Compiler suggests check for a null reference

Null-forgiving Operator

```
[Fact]
public void CalculateMargin_WhenInvalidCostIsNull_ShouldGenerateError()
{
  // Arrange
  string? cost = null;
  string price = "100";
  var product = new Product();

  // Act
  Action act = () => product.CalculateMargin(cost!, price);

  // Assert
  var ex = Assert.Throws<ArgumentException>(act);
  Assert.Equal("Please enter the cost", ex.Message);
}
```



Nullable Context

Nullable annotation context

Nullable warning context

Controls how the compiler interprets reference type variables

```
// Nullable reference type
string? Reason;
```

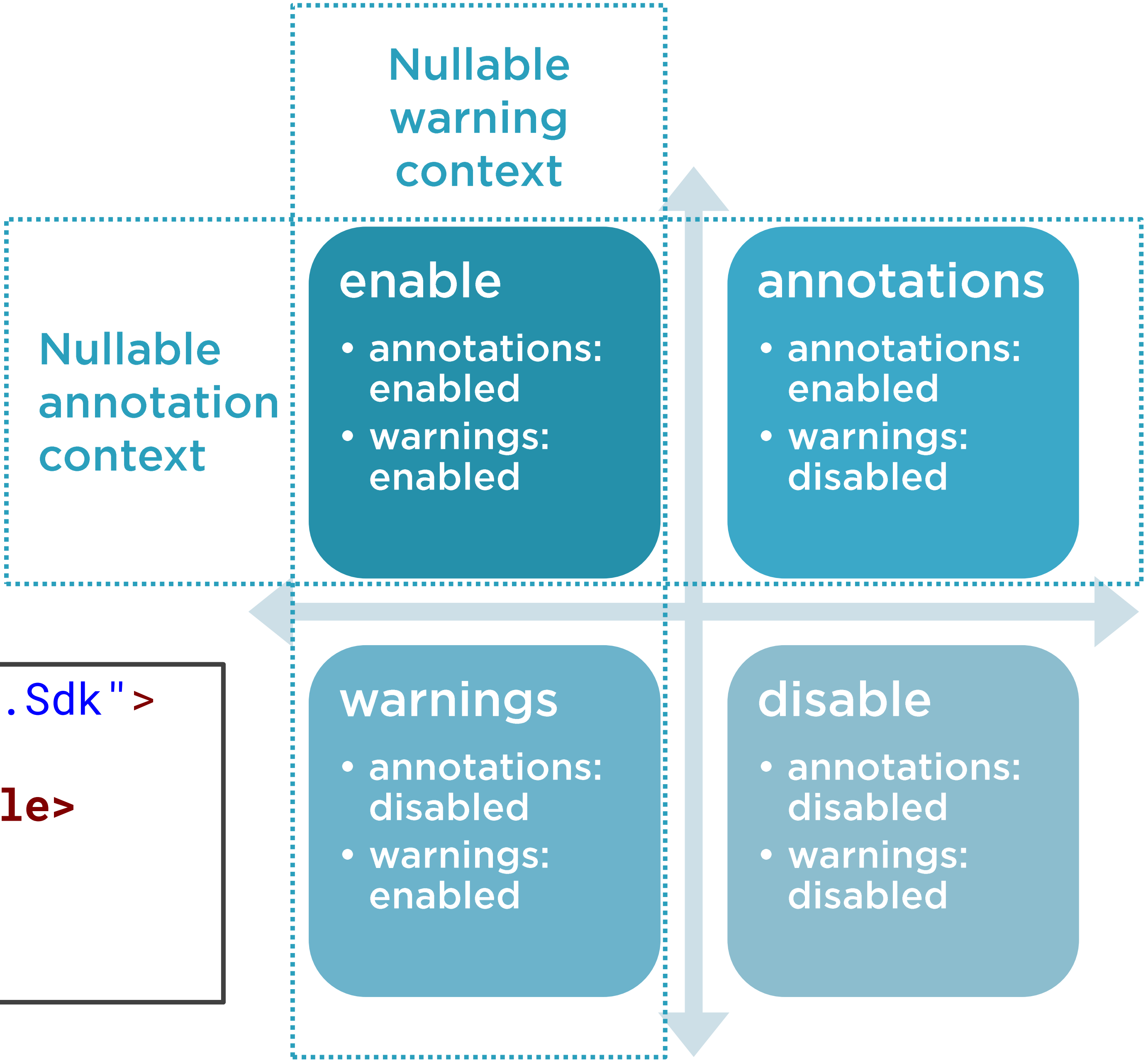
Controls the warnings generated by the compiler

Project File Nullable Element

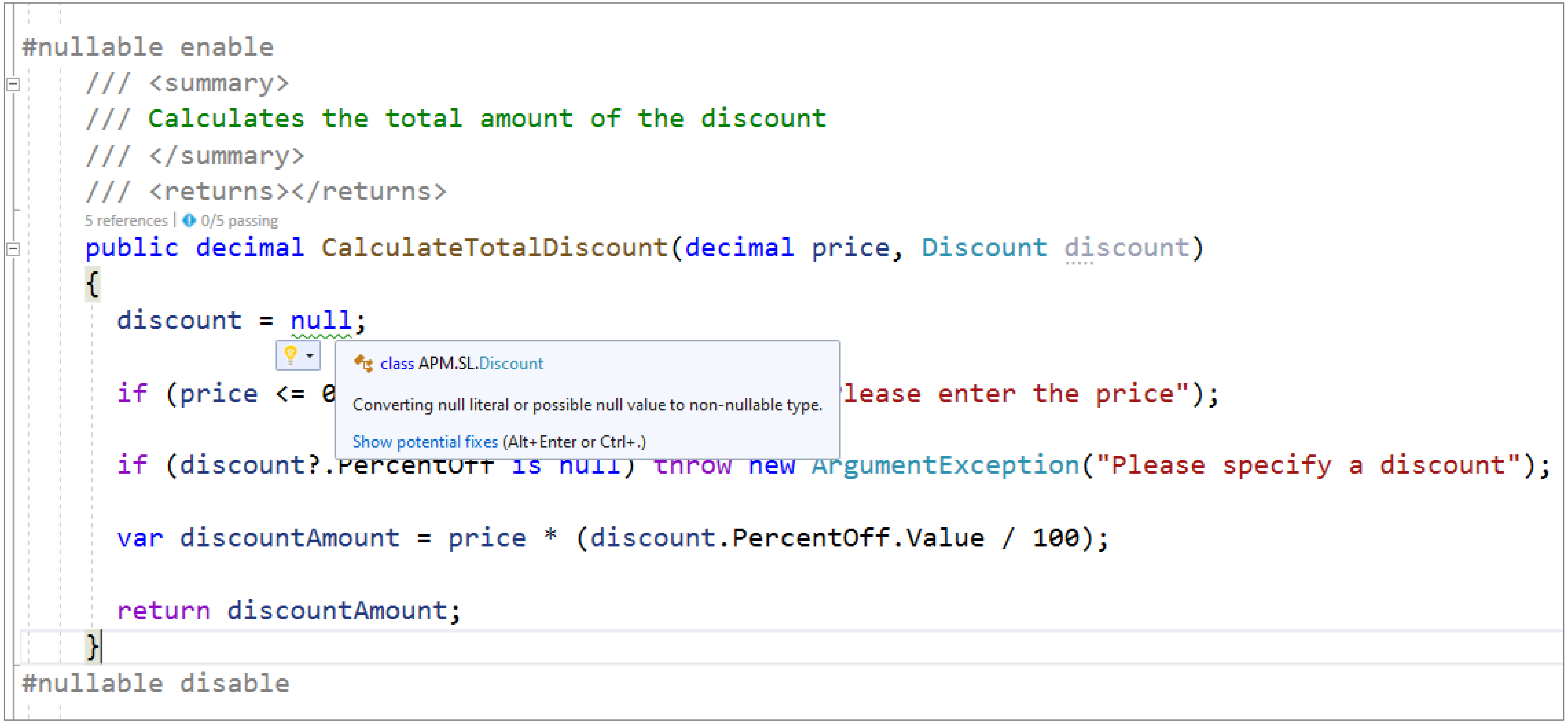
```xml
<Project Sdk="Microsoft.NET.Sdk">

  <PropertyGroup>
    <TargetFramework>netcoreapp3.1</TargetFramework>
    <RootNamespace>APM.SL</RootNamespace>
    <Nullable>enable</Nullable>
  </PropertyGroup>

</Project>
```

Nullable Element Values

```
<Project Sdk="Microsoft.NET.Sdk">
  ...
   <Nullable>disable</Nullable>
  ...

</Project>
```



Nullable Directive



```
#nullable enable
    /// <summary>
    /// Calculates the total amount of the discount
    /// </summary>
    /// <returns></returns>
    5 references | 0/5 passing
    public decimal CalculateTotalDiscount(decimal price, Discount discount)
    {
      discount = null;

      if (price <= 0                                              lease enter the price");

      if (discount?.PercentOff is null) throw new ArgumentException("Please specify a discount");

      var discountAmount = price * (discount.PercentOff.Value / 100);

      return discountAmount;
    }
#nullable disable
```

class APM.SL.Discount

Converting null literal or possible null value to non-nullable type.

Show potential fixes (Alt+Enter or Ctrl+.)

Demo

Reference type nullability features

C# 8 (or higher) ONLY

- .NET Core 3.0 (or higher)
- .NET Standard 2.1 (or higher)

Guidelines and Summary

Use Nullable Value Types as Needed

```
// Value types
bool IsInStock;                   // false
int Count;                        // 0
decimal Price;                    // 0
DateTime EffectiveDate;           // 1/1/0001
```

Use to distinguish between
“not assigned” (null)
and
“set to a value”

```
// Nullable value types
bool? IsInStock;                    // null
int? Count;                         // null
decimal? Price;                     // null
DateTime? EffectiveDate;            // null
```

Guard Against Null Nullable Value Types

Use HasValue to determine if the variable has a value

```
public bool ValidateEffectiveDate(DateTime? effectiveDate)
{
  if (!effectiveDate.HasValue) return false;

  if (effectiveDate.Value < DateTime.Now.AddDays(7)) return false;

  return true;
```

Use Value to obtain the value

Guard Against Null Reference Types

```
public decimal CalculateTotalDiscount(decimal price, Discount discount)
{
  if (price <= 0) throw new ArgumentException("Please enter the price");

  if (discount is null) throw new ArgumentException("Enter a discount");

  var discountAmount = price * (discount.PercentOff / 100);

  return discountAmount;
```

Use `is null` to check for a null reference type

Use the Null-conditional Operator

```
if (pricing is null) throw ...;

if (pricing.Discount is null) throw ...;

if (pricing.Discount.PercentOff is null) throw ...;
```

```
if (pricing?.Discount?.PercentOff is null) throw ...;
```

Use the null-conditional operator for short-circuiting

Enable Reference Type Nullability Features

```xml
<Project Sdk="Microsoft.NET.Sdk">

  <PropertyGroup>
    <TargetFramework>netcoreapp3.0</TargetFramework>
    <RootNamespace>APM.SL</RootNamespace>
    <Nullable>enable</Nullable>
  </PropertyGroup>

</Project>
```

Add to each project file

Minimum: C# 8 (.NET Standard 2.1 OR .NET Core 3)

Use Nullable and Non-nullable Reference Types

Compiler warns if attempt to assign a null

```
// Non-nullable reference types
string Reason = "";                                   // !null
Discount PriceDiscount = new Discount();              // !null
List<Discount> Discounts = new List<Discount>();      // !null
```

```
// Nullable reference types
string? Reason;                                       // null
Discount? PriceDiscount;                              // null
List<Discount>? Discounts;                            // null
```

Compiler warns if attempt to access without null check