

C# for Java Developers Crash Course

PAL Val | Discord: @*valknight.xyz*

PAL Jude | Discord: @jude_birch

who is this for?

who is this for?

Fairly new to C#



on the Java note

a quick warning in advance

C#

MICROSOFT'S JAVA

things you can keep if you already know java

Java Virtual Machine -> Common Language Runtime

Most of the syntax

Lots of the same OO language features

Garbage collection (we'll be talking about this!)

>> this session will assume you are semi-comfortable with Java principles!

on the games note

last warning, then content, i swear

this is written with Unity in mind

this is written with Unity in mind

but nearly all of this is transferable

this is written with Unity in mind

but nearly all of this is transferable

and for gamedev folks, you'll be using Unity as part of your degree

C# Fundamental 0:

What IDE do I use?

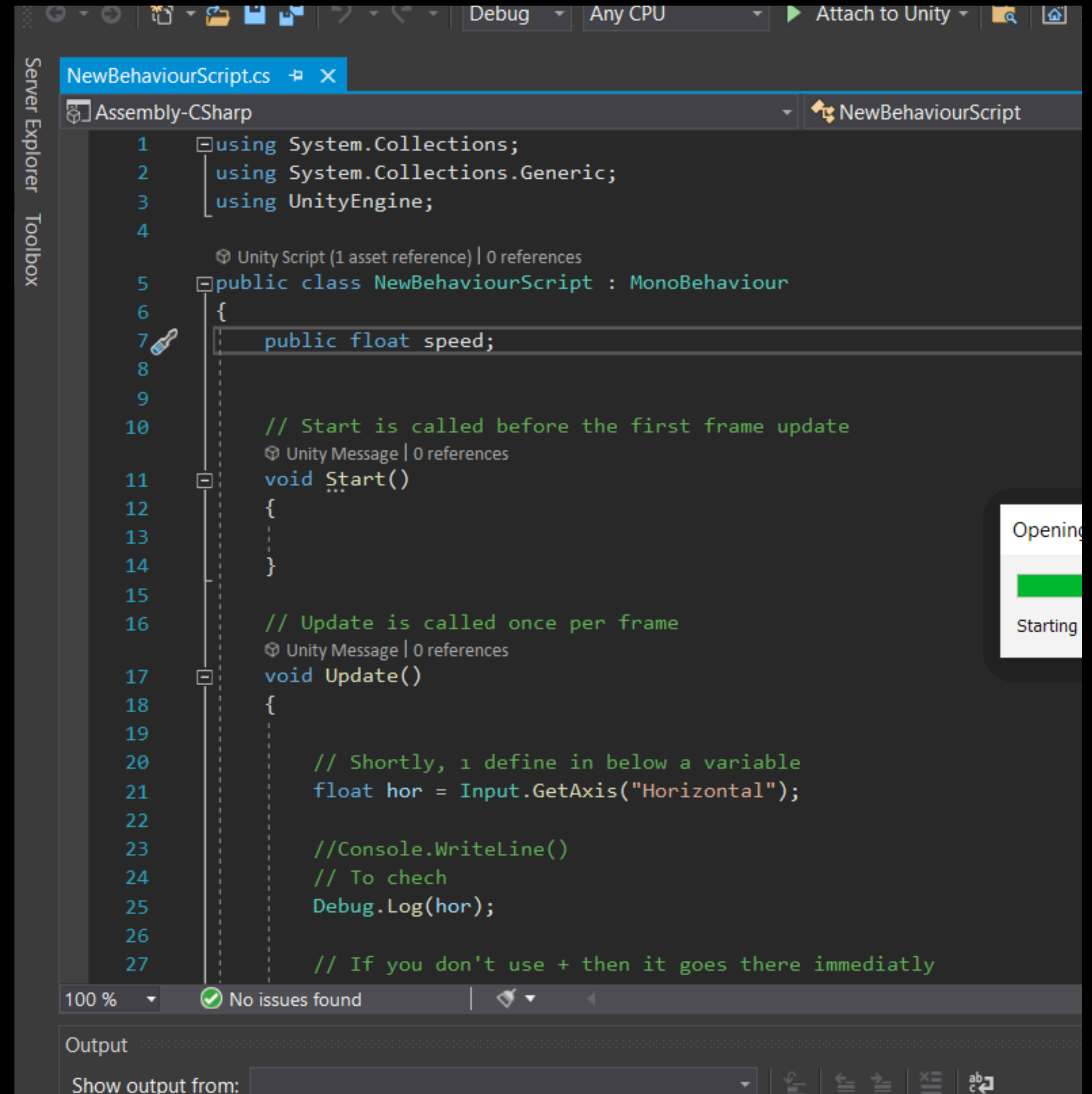
C# Fundamental 0:

**Which of the three different Visual Studios
should I use?**

Visual Studio

Windows

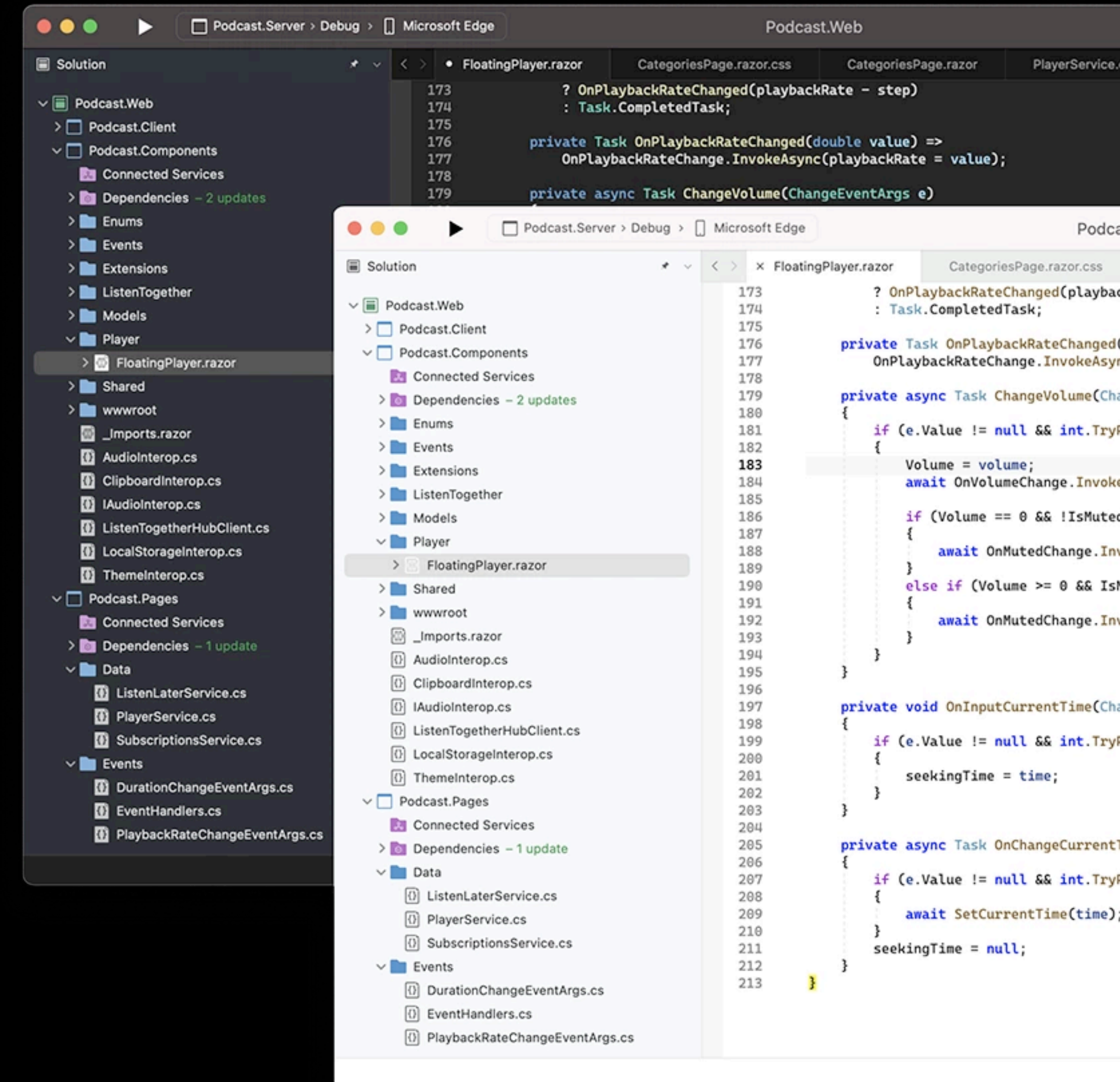
- + Free below a revenue cap
- + The OG C# IDE
- + Great Unity integration OOB
- + Lots, and lots of features
- Windows only
- Quite heavy
- Lots, and lots of features



Visual Studio for Mac

macOS

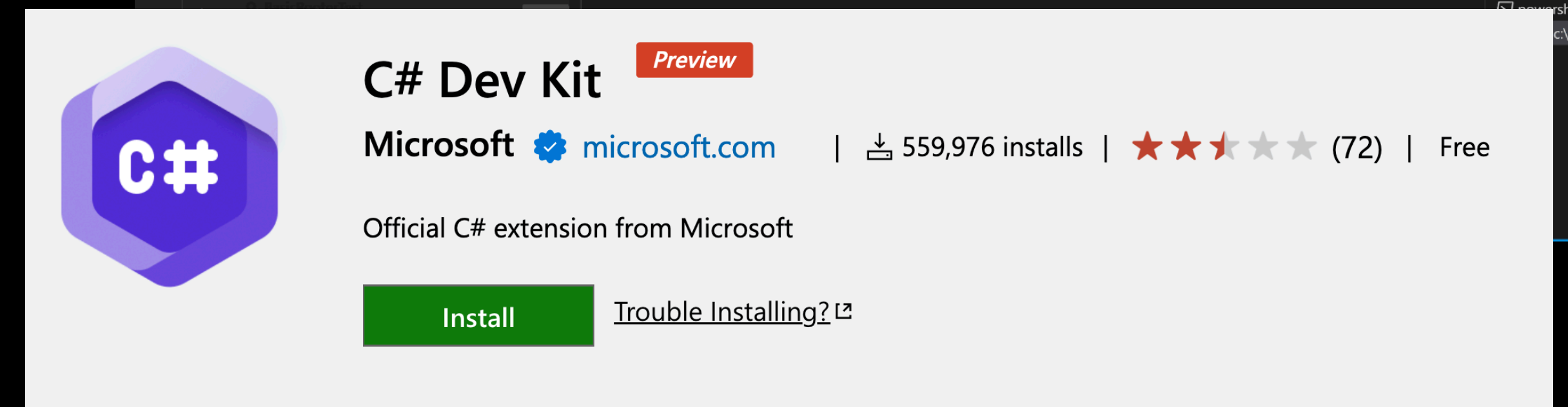
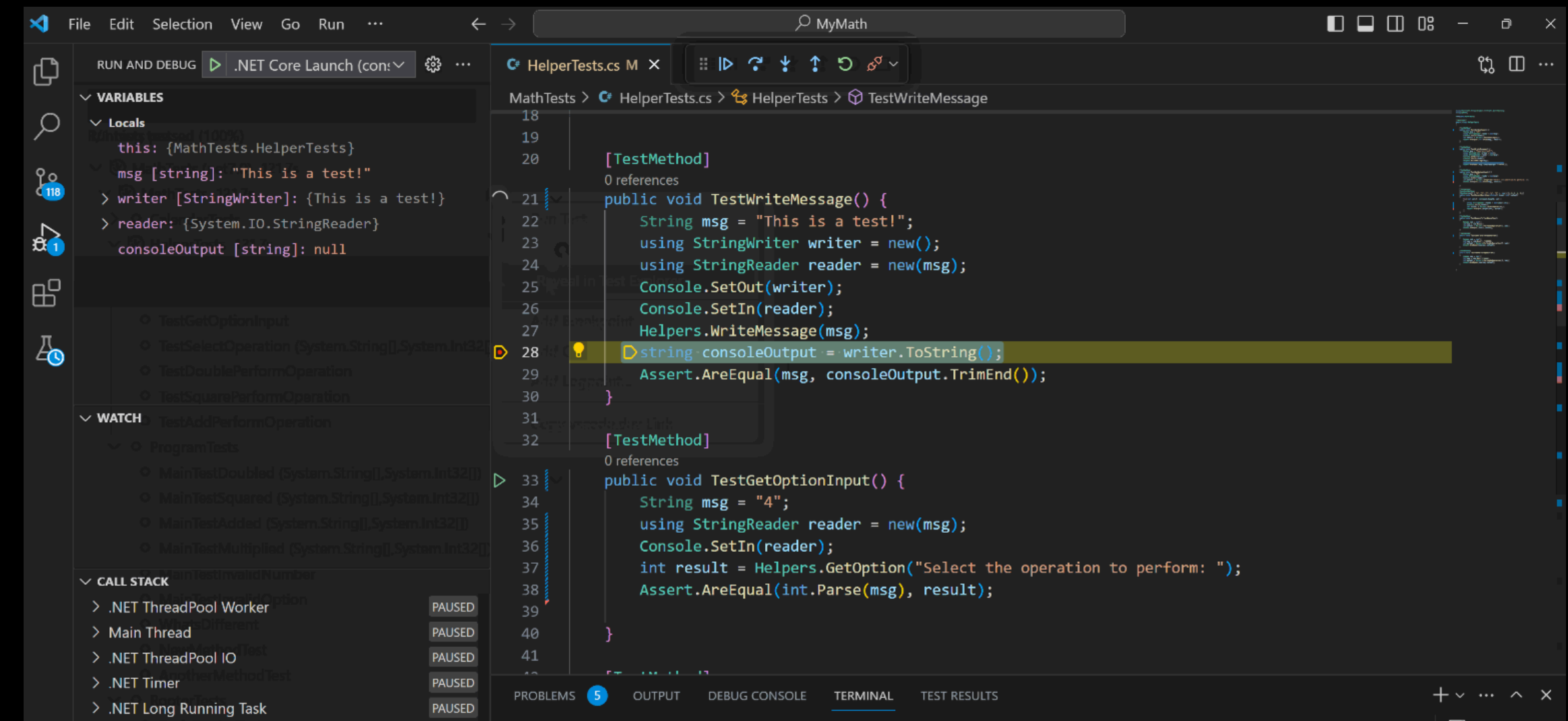
- + Free below a revenue cap
- + Good Unity integration OOB
- + Quick to get started with
- Already deprecated by Microsoft
- Buggy and clunky
- Lacking in a lot of features compared to its Windows counterpart
- Not really Visual Studio - it's just an old IDE called Xamarin Studio, which itself was an older IDE called MonoDevelop



Visual Studio Code

Windows, macOS, Linux

- + Base editor is free and open source
- + The only supported option on Linux (and soon to be macOS...)
- + Extension support - can keep your one IDE & workflows for C#, Python, JS...
- Unity support is still in preview, and is still a little rough around the edges
- Requires installation of extensions for C#
- Extension is *not* open source, still subject to same VS license, for less features than the other VS IDEs...



Visual Studio

Windows

Visual Studio for Mac

macOS

Visual Studio Code

Windows, macOS, Linux

Visual Studio

Windows

Visual Studio for Mac

macOS

Visual Studio Code

Windows, macOS, Linux



A new foe has appeared!

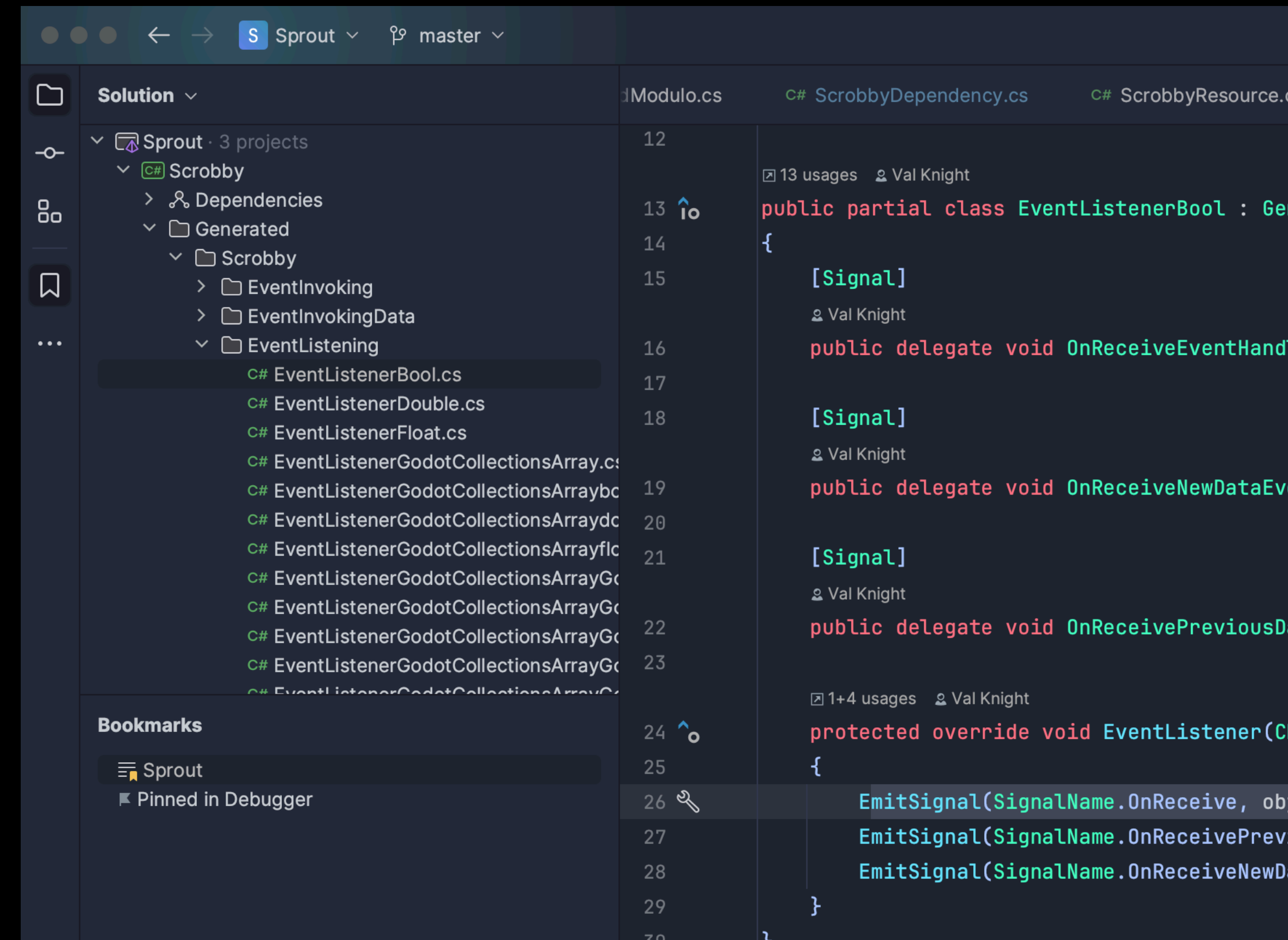
CHALLENGER APPROACHING



JetBrains Rider

Windows, macOS, Linux

+ Free while you're a student, or if you use the Nightly build



Using Rider to write C# makes me happy. I have never seen code refactoring tools that actually work - always without exception. It's amazing when you can rely on it.

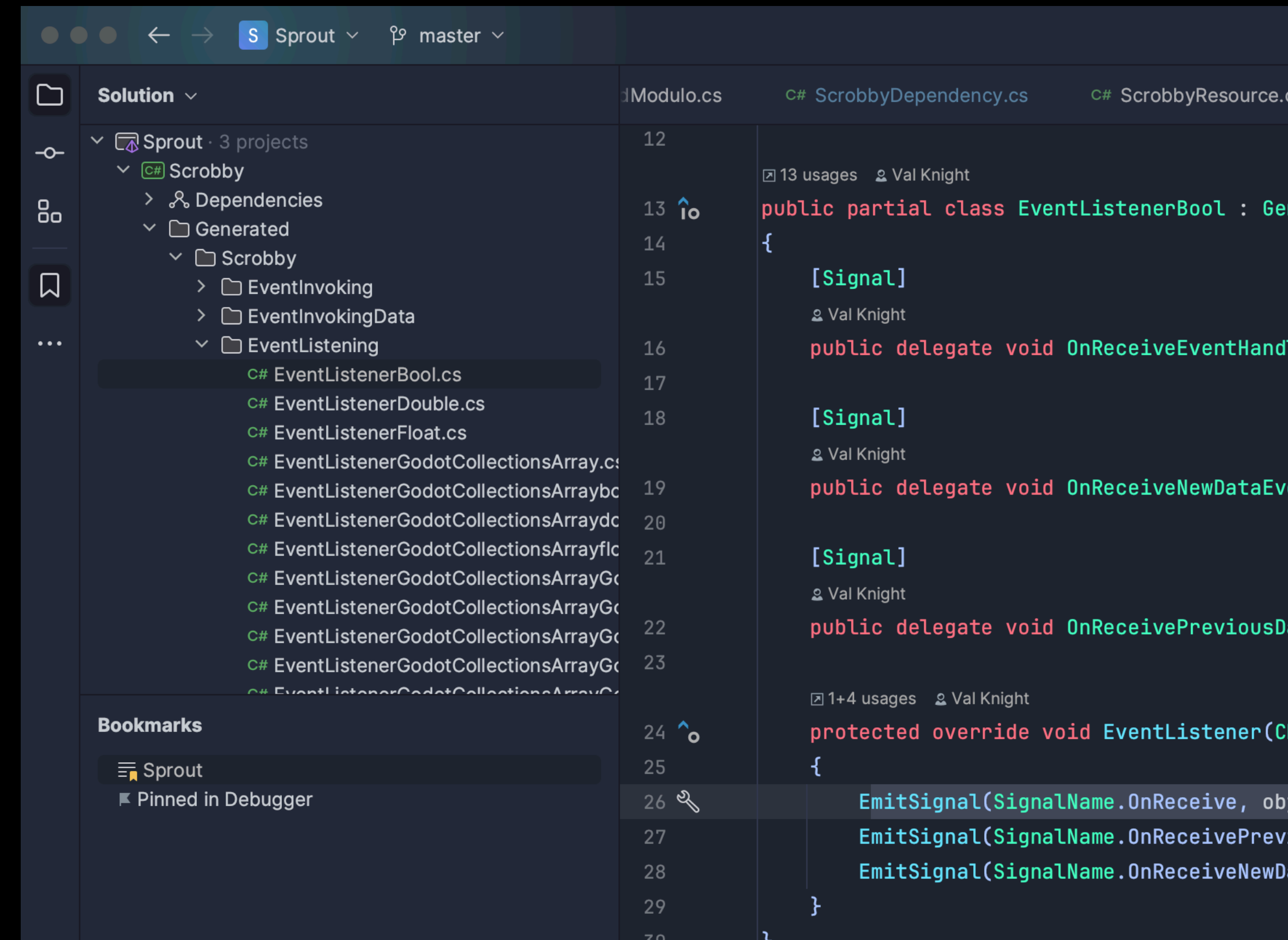
Joachim Ante,
Unity CTO & Founder

JetBrains Rider

Windows, macOS, Linux

+ Free while you're a student, or if you use the Nightly build

+ Mature, stable, reliable



Using Rider to write C# makes me happy. I have never seen code refactoring tools that actually work - always without exception. It's amazing when you can rely on it.

Joachim Ante,
Unity CTO & Founder

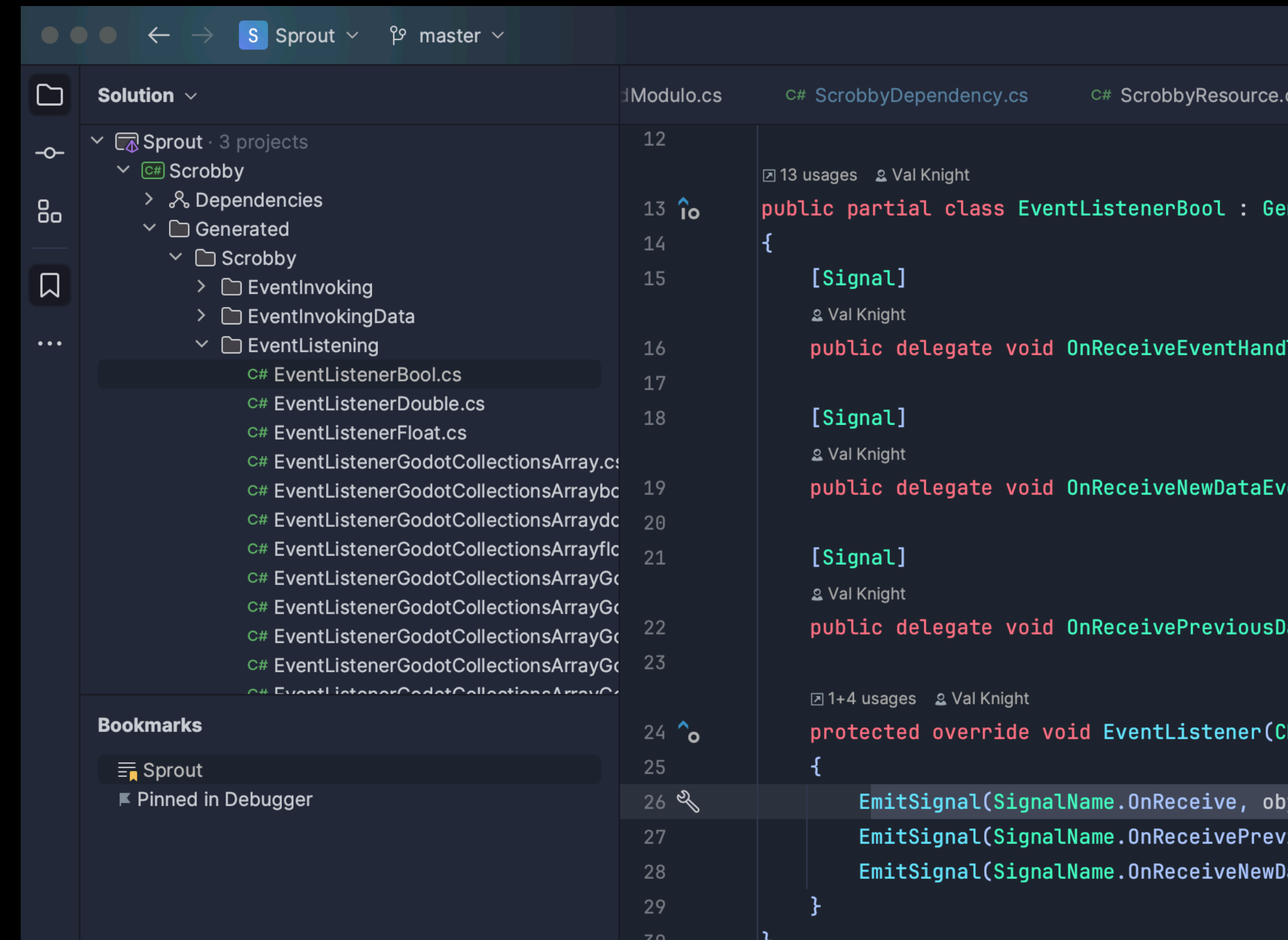
JetBrains Rider

Windows, macOS, Linux

+ Free while you're a student, or if you use the Nightly build

+ Mature, stable, reliable

+ Supports cutting edge Unity features like ECS, Burst



Using Rider to write C# makes me happy. I have never seen code refactoring tools that actually work - always without exception. It's amazing when you can rely on it.

Joachim Ante,
Unity CTO & Founder

JetBrains Rider

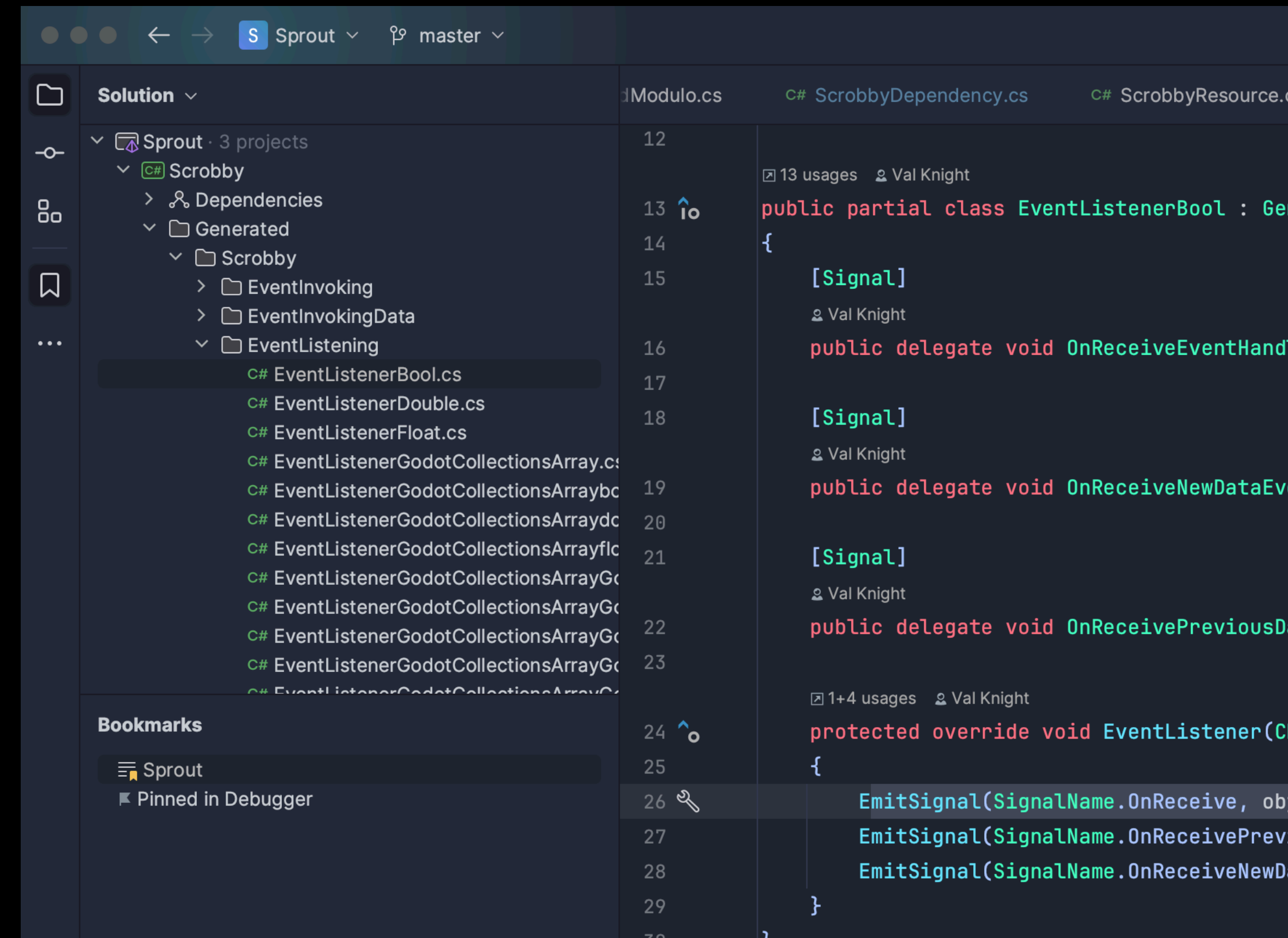
Windows, macOS, Linux

+ Free while you're a student, or if you use the Nightly build

+ Mature, stable, reliable

+ Supports cutting edge Unity features like ECS, Burst

+ Shader support + Performance warnings as part of IntelliSense warnings



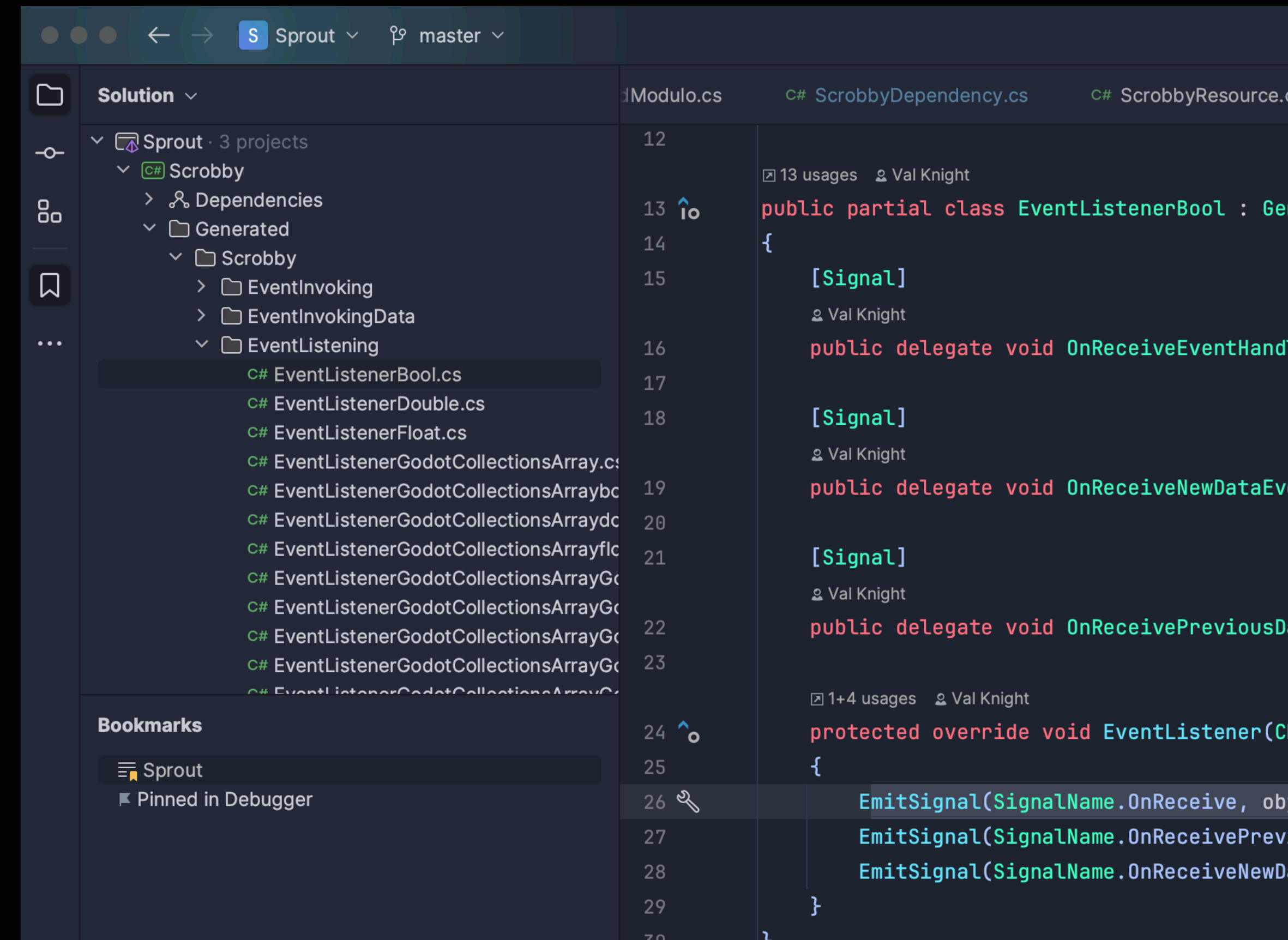
Using Rider to write C# makes me happy. I have never seen code refactoring tools that actually work - always without exception. It's amazing when you can rely on it.

Joachim Ante,
Unity CTO & Founder

JetBrains Rider

Windows, macOS, Linux

- + Free while you're a student, or if you use the Nightly build
- + Mature, stable, reliable
- + Supports cutting edge Unity features like ECS, Burst
- + Shader support + Performance warnings as part of IntelliSense
- + Integration with the Unity Editor and Unity Documentation



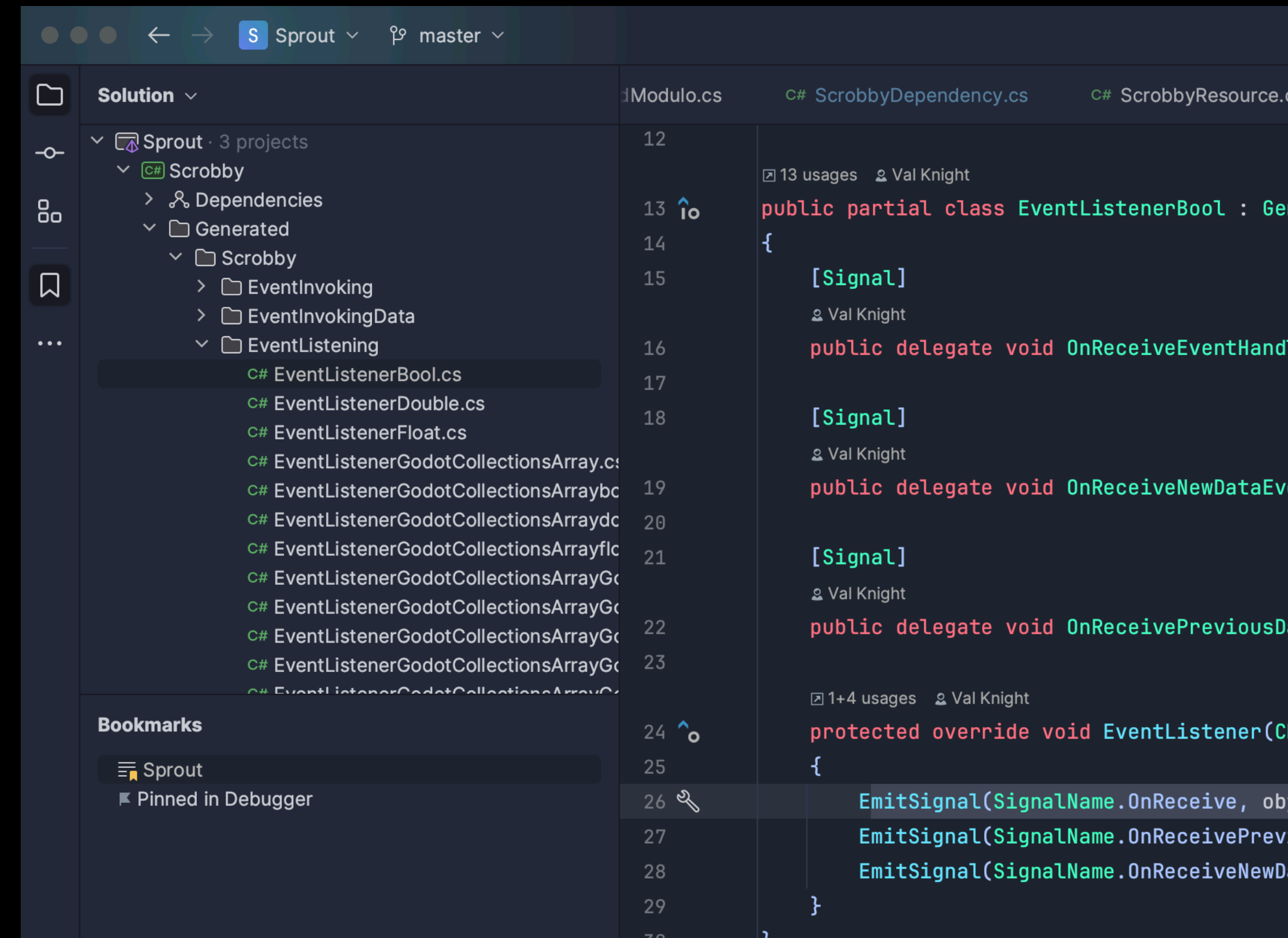
Using Rider to write C# makes me happy. I have never seen code refactoring tools that actually work - always without exception. It's amazing when you can rely on it.

Joachim Ante,
Unity CTO & Founder

JetBrains Rider

Windows, macOS, Linux

- + Free while you're a student, or if you use the Nightly build
- + Mature, stable, reliable
- + Supports cutting edge Unity features like ECS, Burst
- + Shader support + Performance warnings as part of IntelliSense warnings
- + Integration with the Unity Editor and Unity Documentation
- + Recommended by one of the cofounders of Unity



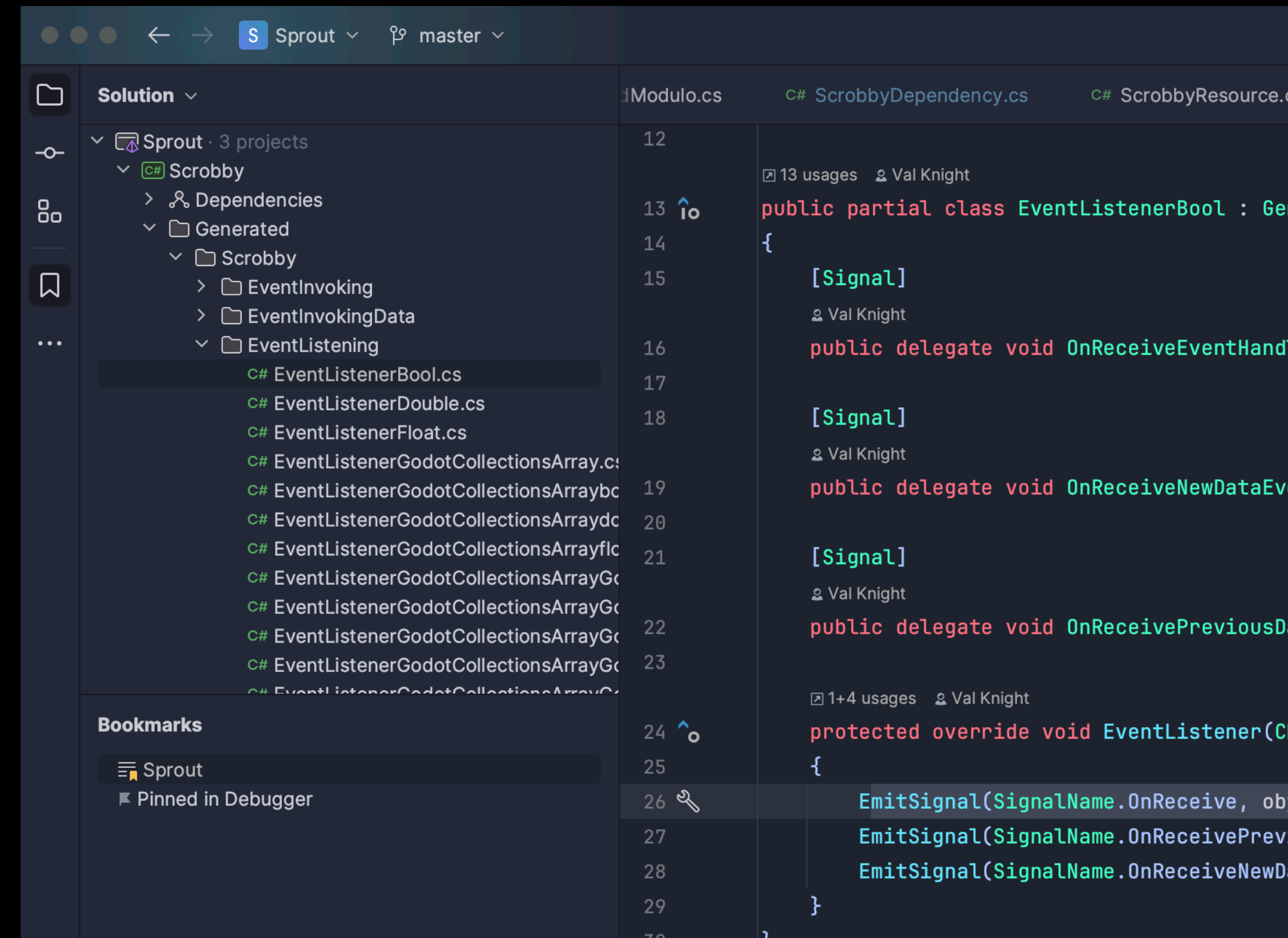
Using Rider to write C# makes me happy. I have never seen code refactoring tools that actually work - always without exception. It's amazing when you can rely on it.

Joachim Ante,
Unity CTO & Founder

JetBrains Rider

Windows, macOS, Linux

- + Free while you're a student, or if you use the Nightly build
- + Mature, stable, reliable
- + Supports cutting edge Unity features like ECS, Burst
- + Shader support + Performance warnings as part of IntelliSense warnings
- + Integration with the Unity Editor and Unity Documentation
- + Recommended by one of the cofounders of Unity
- + Super familiar if you've used IntelliJ



Using Rider to write C# makes me happy. I have never seen code refactoring tools that actually work - always without exception. It's amazing when you can rely on it.

Joachim Ante,
Unity CTO & Founder

JetBrains Rider

Windows, macOS, Linux

- Pretty memory intensive (but, so is VS Code...)

Rider

Cross-platform .NET IDE



first year

£119.00

second year

£95.00

third year onwards

£71.00

[Get quote](#)

Buy

Installed

Update all



Rider

2023.2 Nightly

Update



JetBrains Rider

Windows, macOS, Linux

- Pretty memory intensive (but, so is VS Code...)
- Stable release requires signing up to JetBrains Student Program

Rider

Cross-platform .NET IDE



first year

£119.00

second year

£95.00

third year onwards

£71.00

[Get quote](#)

Buy

Installed

Update all



Rider

2023.2 Nightly

Update



JetBrains Rider

Windows, macOS, Linux

- Pretty memory intensive (but, so is VS Code...)
- Stable release requires signing up to JetBrains Student Program
- Post graduation you (or your employer) will have to pay*
 - Rider Early Access / Nightly is completely free, but can have bugs

Rider

Cross-platform .NET IDE



first year

£119.00

second year

£95.00

third year onwards

£71.00

[Get quote](#)

Buy

Installed

Update all



Rider

2023.2 Nightly

Update



JetBrains Rider

Windows, macOS, Linux

- Pretty memory intensive (but, so is VS Code...)
- Stable release requires signing up to JetBrains Student Program
- Post graduation you (or your employer) will have to pay*
 - Rider Early Access / Nightly is completely free, but can have bugs
 - There is also a pretty heavy graduate discount, and you get to keep the version of Rider you pay for as a perpetual license



For graduates

Check out the 40% graduation discount for former Student License holders! Select the product of your choice, and the discount will be applied automatically:

Rider



£71.40

£119.00

Buy now

Installed

Update all



Rider

2023.2 Nightly

Update



Visual Studio

Windows

Visual Studio for Mac

macOS

Visual Studio Code

Windows, macOS, Linux

JetBrains Rider

Windows, macOS, Linux

Visual Studio

Windows

Windows devs: choose either

JetBrains Rider

Windows, macOS, Linux


macOS / Linux: just use Rider

C# Fundamental 1: Value vs Reference Types

Value Type

- *Directly* contains it's data
- Generally simple primitives (floats, ints, bools, ***not strings!***)
- ***Cannot be null!***
- Passing it to another method creates a copy of the data to work
with unless you specify it's a reference with the "ref" keyword
- Stored on the stack, meaning fast allocation & deallocation

Value Type

- *Directly* contains it's data
- Generally simple primitives (floats, ints, bools, ***not strings!***)
- ***Cannot be null!***
- Passing it to another method creates a copy of the data to work
with unless you specify it's a reference with the "ref" keyword
- Stored on the stack, meaning fast allocation & deallocation
 - Don't worry if you don't know what the stack means right now! Feel free to grab Val  and she can explain if you're interested :)
 - Time constraints my beloved

Reference Type

- All your objects (strings, arrays lists, dictionaries, classes you define...)
- Passing it to another method is just referencing the same underlying data
- *Reference* to data existing in the “managed heap”
 - Again, feel free to ask Val what this means!
- Stored on the heap, with references existing on the stack
 - New references to existing data are still fast!
 - Creating data is relatively slow :(
 - Can lead to garbage, causing really bad performance issues :((((

a quick game

VALUE

or

REFERENCE

```
int someInt;
```

VALUE

```
int someInt;
```



```
string someString;
```

REFERENCE

```
string someString;
```

```
float someInt;
```

VALUE

```
float someInt;
```

```
string[] arrayOfStrings;
```

REFERENCE

```
string[] arrayOfStrings;
```

```
int[] myIntArray;
```

REFERENCE

```
int[] myIntArray;
```

**Arrays of value types are
still reference types**


```
CoolCustomClass myObject;
```

```
public class CoolCustomClass  
{  
    public int x;  
    public int y;  
}
```

REFERENCE

```
CoolCustomClass myObject;
```

```
public class CoolCustomClass  
{  
    public int x;  
    public int y;  
}
```

```
CoolCustomStruct myStruct;
```

```
public struct CoolCustomStruct  
{  
    public int x;  
    public int y;  
}
```

VALUE

```
CoolCustomStruct myStruct;
```

```
public struct CoolCustomStruct  
{  
    public int x;  
    public int y;  
}
```

```
StructOfStrings myStruct;
```

```
public struct StructOfStrings  
{  
    public string x;  
    public string y;  
}
```

VALUE

```
StructOfStrings myStruct;
```

```
public struct StructOfStrings  
{  
    public string x;  
    public string y;  
}
```

VALUE

(but *x* and *y* are still reference types)

```
StructOfStrings myStruct;
```

```
public struct StructOfStrings  
{  
    public string x;  
    public string y;  
}
```

Structs

- Great for small collections of other fields
- Can support methods, same as a class
- Structs are a *value typed* collection of fields
 - As such, passing a struct to another function will create a *copy*, not a reference by default!
 - You *can* pass a struct by reference with the *ref* keyword - ask Val, and she can show you demos (time constraints!)

demo

C# Fundamental 2:

Garbage Collection

What is garbage?

A reference type, with no references to it

What is garbage collection?

Your C# runtime going 🙄🙄 Looking 🙄🙄, specifically for objects in the managed heap with no references on the stack

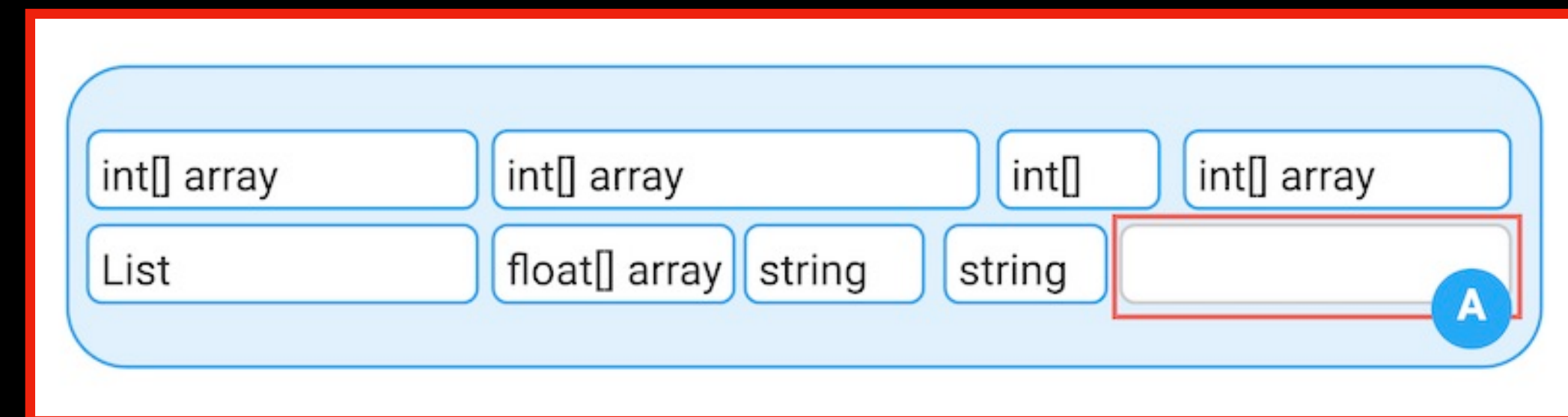


Diagram of the Managed Heap

Copyright Unity Technologies, 2023

Why is garbage bad?

- It's expensive to allocate managed objects
- Garbage Collection is a **very** expensive operation to run, so we want to avoid generating garbage where possible

What do you mean by “expensive”?

- If your game is targeting 60FPS, you have 16.67ms to get a frame out to the display
- Garbage Collection takes a lot of time to complete, especially if you have a lot of garbage
- This could cause frequent stuttering!

How do I avoid generating garbage?

- Keep an eye out for when you're constantly making new objects
 - Could it be done with a value type?
 - Could you modify the values of an object you allocate once?
- Watch out for strings and arrays!
 - Strings are references types; as such, generating and passing around strings can lead to suboptimal performance
 - This includes string concatenations (IE: "foo" + "bar" will generate garbage)
 - Consider alternatives, such as passing an enum instead

Is garbage the end of the world?

- No!

Is garbage the end of the world?

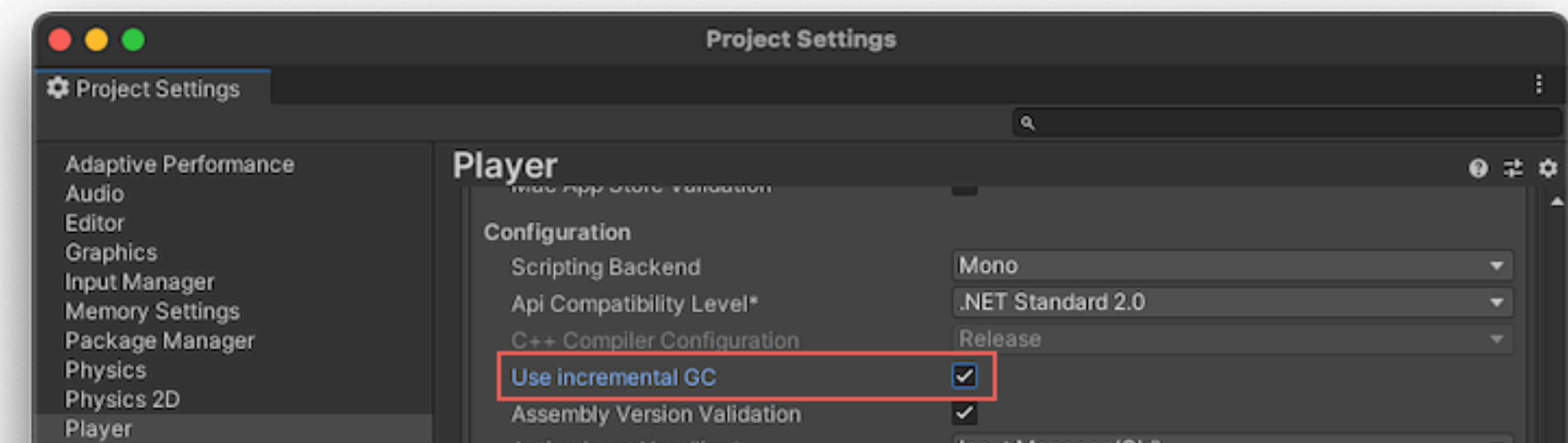
- No!
- But it's useful to have an understanding, and get performance gains where you can

Is garbage the end of the world?

- No!
- But it's useful to have an understanding, and get performance gains where you can
- Unity's "Incremental GC" won't save you, but, it can help to reduce GC "spikes"

Incremental garbage collection

Incremental [garbage collection](#) (GC) spreads out the process of garbage collection over multiple frames. This is the default garbage collection behavior in Unity.

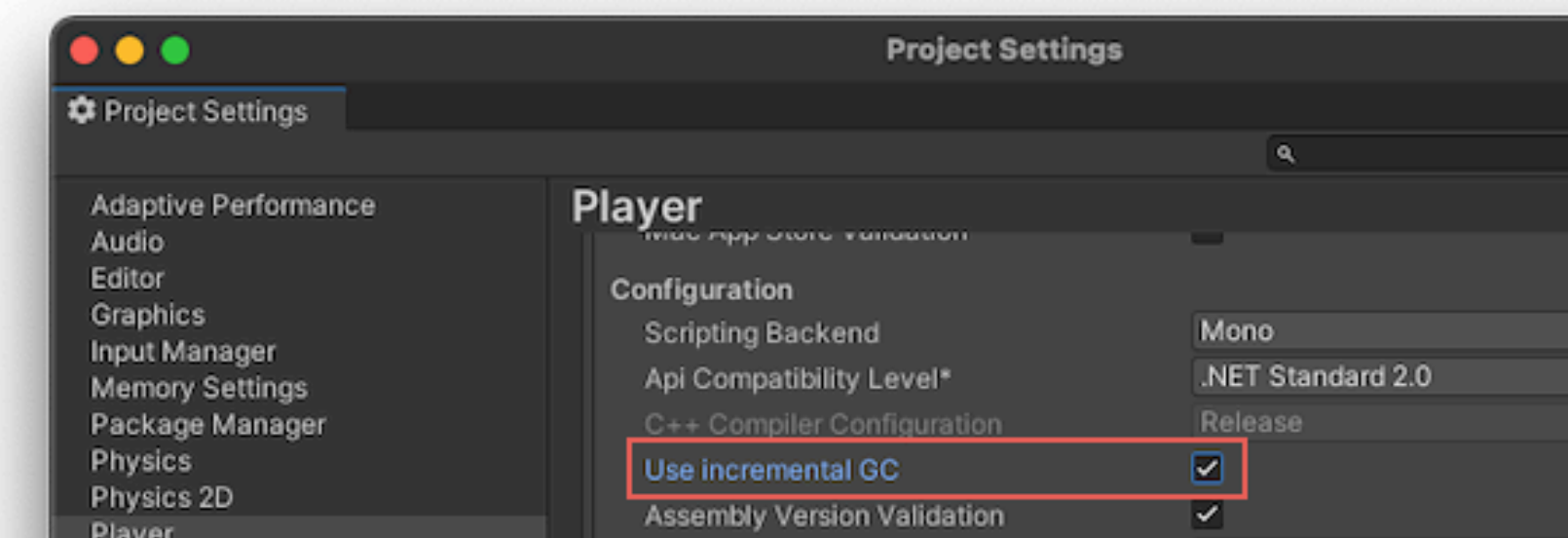


Is garbage the end of the world?

- No!
- But it's useful to have an understanding, and get performance gains where you can
- Unity's "Incremental GC" won't save you, but, it can help to reduce GC "spikes"
 - PAL will likely be doing a hands-on profiling workshop!
 - Keep an eye out 🙄🙄

Incremental garbage collection

Incremental [garbage collection](#) (GC) spreads out the process of garbage collection over multiple collection behavior in Unity.



C# Fundamental 3: Properties

Java Land

getters, setters? more like get-out-of-town-ers. gottem

```
public class Player {  
    private float playerHealth;  
  
    public float GetPlayerHealth() {  
        return playerHealth;  
    }  
  
    public void SetPlayerHealth(float f) {  
        playerHealth = f;  
    }  
}
```

C# Land

```
public class Player {  
    private float playerHealth;  
  
    public float GetPlayerHealth() {  
        return playerHealth;  
    }  
  
    public void SetPlayerHealth(float f) {  
        playerHealth = f;  
    }  
}
```

C# Land

```
public class Player {  
    public float playerHealth;  
}
```


C# Land

```
public class Player {  
    public float PlayerHealth;  
}
```

* naming conventions

but what about encapsulation???

C# Land

the cool part

```
public class Player
{
    public float PlayerHealth;
}
```

C# Land

the cool part

```
public class Player
{
    private float _playerHP;
    public float PlayerHealth
    {
        get
        {
            return _playerHP;
        }
        set
        {
            _playerHP = value;
        }
    }
}
```

C# Land

the cool part

```
public class Player
{
    private float _playerHP;
    public float PlayerHealth
    {
        get
        {
            return _playerHP;
        }
        set
        {
            _playerHP = value;
        }
    }
}
```

What used to be just a value has now been changed into a free getter and setter

C# Land

the cool part

```
public class Player
{
    private float _playerHP;
    public float PlayerHealth
    {
        get
        {
            return _playerHP;
        }
        set
        {
            _playerHP = value;
        }
    }
}
```

What used to be just a value has now been changed into a free getter and setter

This means you do not need to write, generate or otherwise include getters and setters!

C# Land

the cool part

```
public class Player
{
    private float _playerHP;
    public float PlayerHealth
    {
        get
        {
            return _playerHP;
        }
        set
        {
            _playerHP = value;
        }
    }
}
```

What used to be just a value has now been changed into a free getter and setter

This means you do not need to write, generate or otherwise include getters and setters!

This “fake” field is called a “property”!

demo

C# Fundamental 4: Delegates

What is a callback?

- A callback is *some* method you provide to another method, to run when Something happens
- e.g. - downloading files
 - you have a function called `ProcessText(string text);`
 - you have a function called `DownloadText`
 - you WANT to tell `DownloadText` to call `ProcessText` when it's done
 - But, you don't want to hard-code this

Callbacks in Java

- SO! Sometimes, we need callbacks!
- *We could* do something with an interface

Interface Based Callbacks in Java

```
interface ICallback<T> {
    public void CallbackMethod(T data);
}

class Foo {
    public void DoAction(ICallback<Integer> c) {
        // this is doing complex maths
        int randomNumber = 4;
        c.CallbackMethod(randomNumber);
    }
}

public class Bar implements ICallback<Integer> {
    public static void main(String[] args) {
        Foo f = new Foo();
        Bar b = new Bar();
        // this will allow Foo to call our callback!
        f.DoAction(b);
    }

    public void CallbackMethod(Integer data) {
        System.out.println(data);
    }
}
```

This uses generics!
but; don't super worry if generics aren't your jam
just focus on how ICallback is used!

Interface Based Callbacks in C#

```
interface ICallback<T> {  
    public void CallbackMethod(T data);  
}  
  
class Foo {  
    public void DoAction(ICallback<int> c) {  
        // this is doing complex maths  
        int randomNumber = 4;  
        c.CallbackMethod(randomNumber);  
    }  
}  
  
public class Bar: ICallback<int> {  
    public static void main(String[] args) {  
        Foo f = new Foo();  
        Bar b = new Bar();  
        // this will allow Foo to call our callback!  
        f.DoAction(b);  
    }  
  
    public void CallbackMethod(int data) {  
        Debug.Log(data);  
    }  
}
```

This uses generics!
but; don't super worry if generics aren't your jam
just focus on how ICallback is used!

Callbacks in Java

- Sometimes, we need callbacks!
- *We could* do something with an interface
- But this isn't very flexible
 - Everything needs to inherit the interface
 - Very rigid, and can't dynamically decide what's the callback at runtime

Callbacks in C#

- Sometimes, we need callbacks!
- *We could* do something with an interface
- But this isn't very flexible
 - Everything needs to inherit the interface
 - Very rigid, and can't dynamically decide what's the callback at runtime
- Maybe C# has a better way...

Interface Based Callbacks in C#

```
interface ICallback<T> {  
    public void CallbackMethod(T data);  
}  
  
class Foo {  
    public void DoAction(ICallback<int> c) {  
        // this is doing complex maths  
        int randomNumber = 4;  
        c.CallbackMethod(randomNumber);  
    }  
}  
  
public class Bar: ICallback<int> {  
    public static void main(String[] args) {  
        Foo f = new Foo();  
        Bar b = new Bar();  
        // this will allow Foo to call our callback!  
        f.DoAction(b);  
    }  
  
    public void CallbackMethod(int data) {  
        Debug.Log(data);  
    }  
}
```


Delegate Based Callbacks in C#

```
class Foo {  
    public delegate void SetTextCallback(int data);  
  
    public void DoAction(SetTextCallback cb) {  
        // this is doing complex maths  
        int randomNumber = 4;  
        cb(randomNumber);  
    }  
}  
  
public class Bar {  
    public static void main(String[] args) {  
        Foo f = new Foo();  
        Bar b = new Bar();  
        // Explicitly send CallbackMethod!  
        f.DoAction(b.CallbackMethod);  
    }  
  
    public void CallbackMethod(int data) {  
        Debug.Log(data);  
    }  
}
```

Callbacks in C#

- Sometimes, we need callbacks!
- *We could* do something with an interface
- But this isn't very flexible
 - Everything needs to inherit the interface
- Maybe C# has a better way...
- As a delegate is a datatype, we can do more complex things (e.g, lists of delegates for callbacks, delegates in dictionaries, etc etc etc...)

demo

Example use of delegates

- How would you build an achievement system?
- Do you check all your achievements, every frame?
 - No!
- Instead, lets have other bits of the game simply tell the achievements when things happen...
- E.g., when the player moves, the achievement system will just have a list of delegates to run!

Example use of delegates

- How would you build an achievement system?
- Do you check all your achievements, every frame?
 - No!
- Instead, lets have other bits of the game simply tell the achievements when things happen...
- E.g., when the player moves, the achievement system will just have a list of delegates to run!
 - We have just come up with the ***Observer Programming Pattern***

Takeaways

Takeaways

- Pick an IDE - Visual Studio or JetBrains Rider are recommended!
- Keep in mind whether something is a value, or a reference!
- Be aware of garbage, and try to minimise generating garbage in your code!
- Delegates can really help clean up your code!

Further reading

Further reading

- This QR code should take you to a bunch of links for further reading
 - Outside of this, also explore learn.microsoft.com, Unity Learn, and the Unity Documentation!
 - Microsoft's & Unity's documentation are both really good, especially compared to what you may be used to
- It also has a direct link to the page to get the JetBrains student plan, for free Rider!



Further reading / Some highlights!

- Unity Learn
 - Great for beginners, and is usually up to date and trusted
 - (love u Unity Learn team <3)
- Catlike Coding
 - Great resource for when you want to go a bit deeper!
- Acerola
 - Amazing channel if you ever want to explore the more shadery / technical art side of gamedev - we'll likely be running some shader workshops in future, but, well worth a watch!



! YouTube Tutorial / ChatGPT Warning !



Untitled 4 — Edited

```
hey guys welcome to my C# tutorial, today we will be  
learning how to make .NET applications
```

⚠ YouTube Tutorial / ChatGPT Warning ⚠

- Be careful with YouTube tutorials (or, StackOverflow, Unity Forums/Discussions, etc etc...) and using ChatGPT
- Unity has changed quite a bit in the past few years
- AI tools are primarily trained on hobbyist code - there aren't many open source commercial Unity games! - so, common mistakes will get repeated, and it will happily spout misinformation
- If you need advice on if a tutorial should be trusted, reach out to us on PAL! We can take a quick skim over, and give a temperature check for you

Next Sessions

Next Sessions

- **Thursday**

- Live game making workshop! Come along, and learn lots of practical Unity tips, remaking a certain cult-classic in the Informatics department

 **Thursday 12th,**  **FTL Lab,**  **14:00 -> 15:00**

- **Other upcoming sessions**

- Will be announced on the EngInf Discord!
- PAL can also help out with specific gamedev queries you may have!

