Implementing Equality for Value Types

Simon Robinson
http://TechieSimon.com
@TechieSimon

pluralsight
hardcore dev and IT training

4. Overriding Equality: Value Types
5. Overriding Equality: Reference Types
8. Comparers and Equality Comparers
7. Strings
10. Structural Equality
3. == in C#
6. Comparisons in .NET
9. Hash Codes
2. Equality in .NET
1.Why Is Equality So Hard?

- Why implement equality for value types?
- Demo of implementing equality:
  - Override `object.Equals()`.
  - Implement `IEquatable<T>.Equals()`.
  - Implement == and != overloads.
  - Override `object.GetHashCode()`.

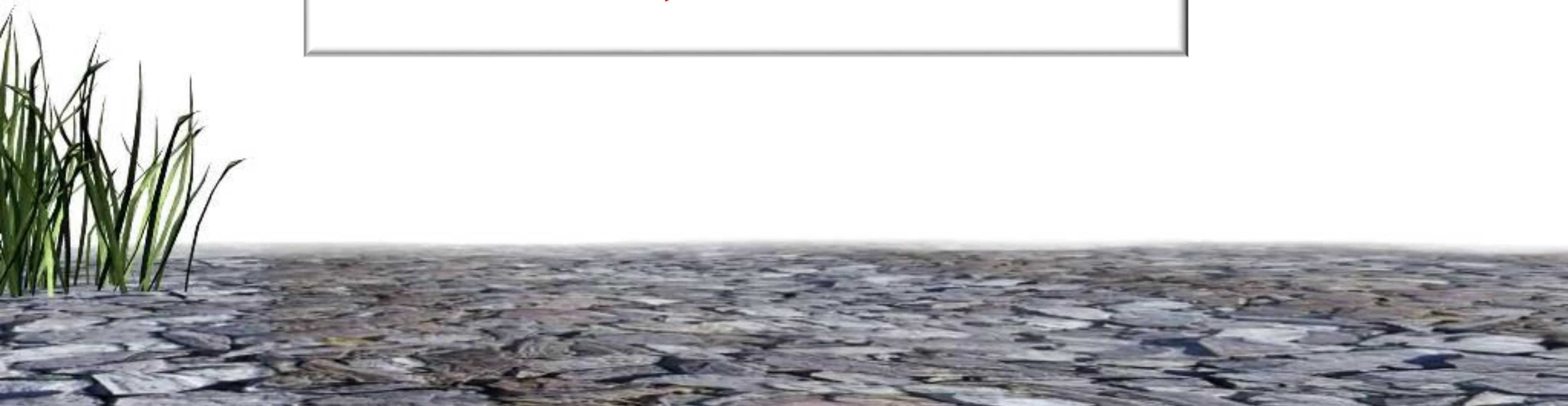
Terminology in this Course

Implement Equality

Override Equality

= Define what equality does for a type

Not standard terms
(there aren't any standard terms for this)



Why Override Equality?

Use `operator ==` for your type	Performance	Change Meaning of Equality
== won't work for value types unless you overload it	To avoid: Boxing Reflection	(Default is: Equal if calling `Equals()` on all fields returns `true`

Why Override Equality?

Use operator ==
for your type

==
for
unless

Performance

Change Meaning of Equality

(Default is:
Equal if calling Equals()
on all fields
returns true

```
struct MyStruct
{
    // for debugging only.
    private int _version;

    // etc.
```

Can implement equality
to ignore this field

Why Override Equality?

Use `operator ==` for your type

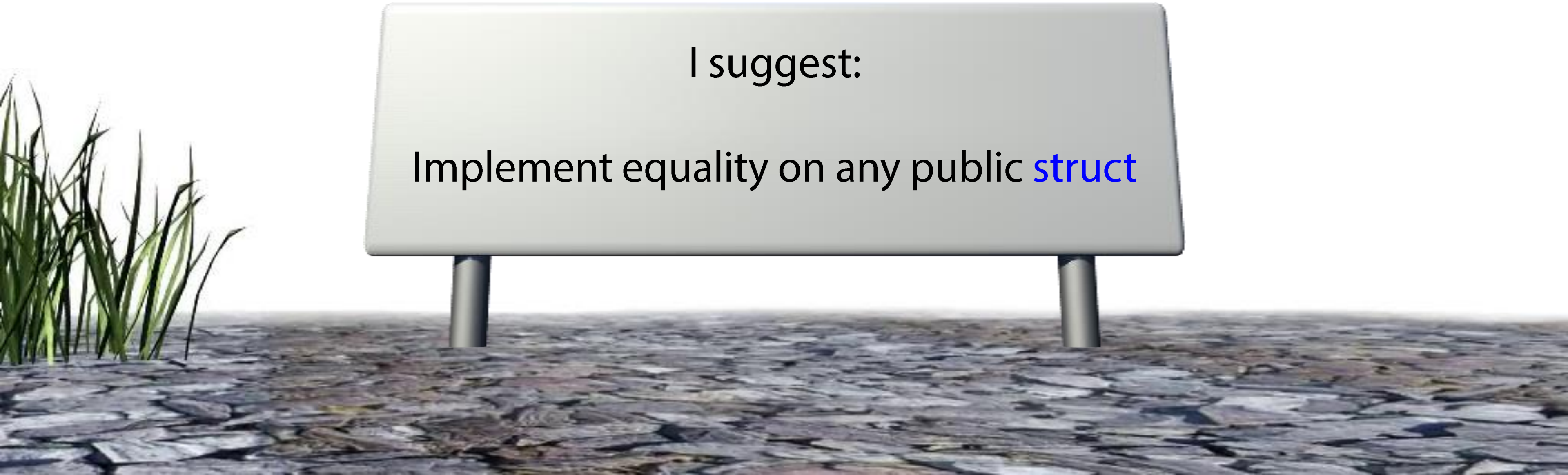
== won't work
for value types
unless you overload it

Performance

To avoid:
Boxing
Reflection

Change Meaning of Equality

(Default is:
Equal if calling `Equals()`
on all fields
returns `true`



I suggest:

Implement equality on any public `struct`

Code Demo

To Override Equality for FoodItem…

Override `object.Equals()`

Avoid reflection

Implement `IEquatable<FoodItem>`

Avoid boxing

Give type safety

Implement `==`

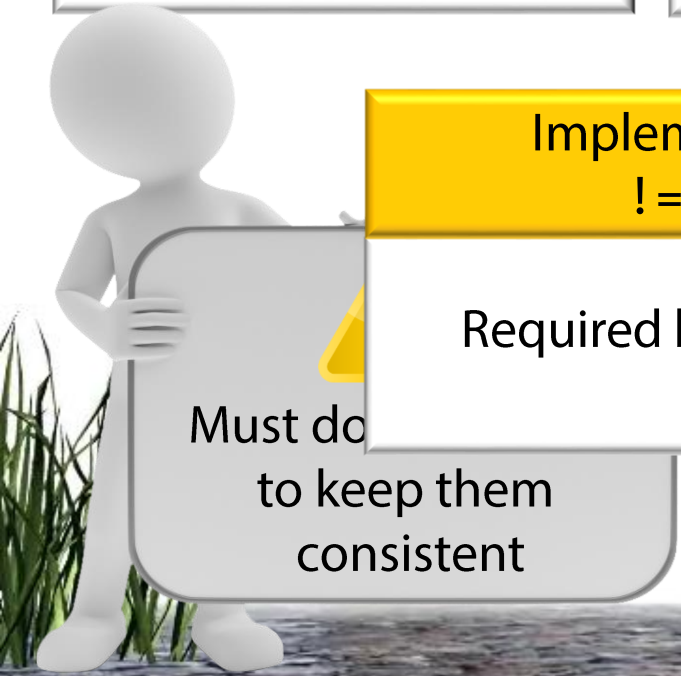
Allow using ==

Implement `!=`

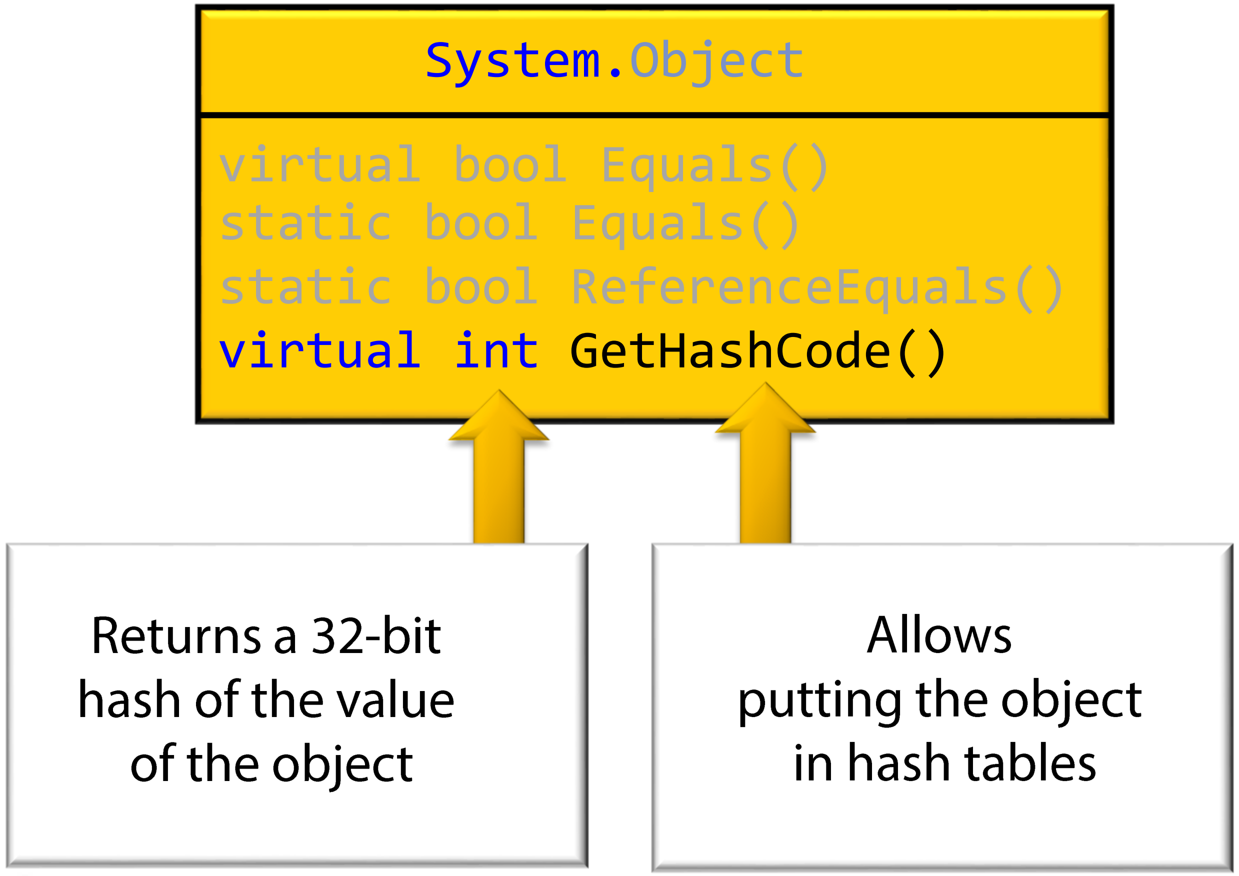
Required by C#

Implement `object.GetHashCode()`

Good practice



GetHashCode()



```
System.Object
virtual bool Equals()
static bool Equals()
static bool ReferenceEquals()
virtual int GetHashCode()
```

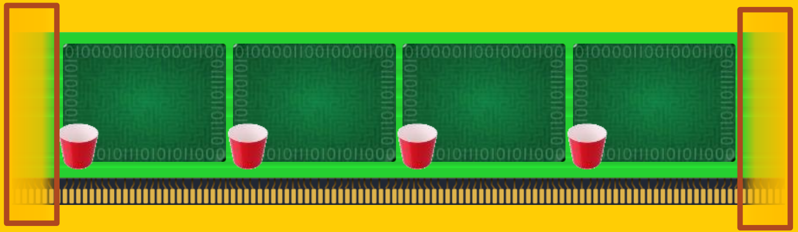
Returns a 32-bit hash of the value of the object
Allows putting the object in hash tables

GetHashCode()

```
System.Object
virtual bool Eq
static bool Equ
static bool Ref
virtual int GetH
```

```
Dictionary<TKey, TValue>
```

Contains a hash table



Hash tables require that:

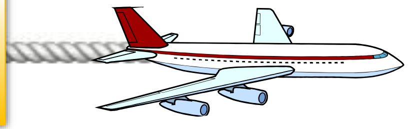
If `x.Equals(y)`

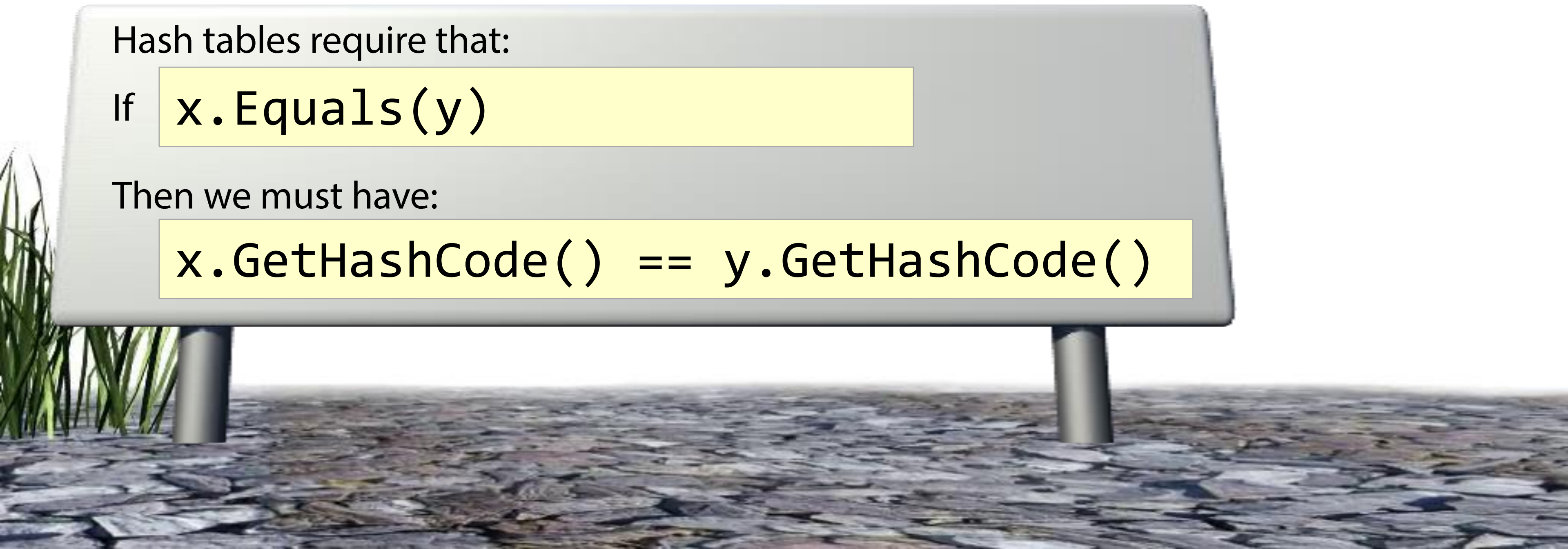
Then we must have:

```
x.GetHashCode() == y.GetHashCode()
```

GetHashCode()

If you override `Equals()`…
then you must override `GetHashCode()` to match





Hash tables require that:

If
```
x.Equals(y)
```

Then we must have:
```
x.GetHashCode() == y.GetHashCode()
```

Code Demo

What We Did…

Override
`object.Equals()`

Type check then call
`IEquatable<T>`
`.Equals()`

Implement
`IEquatable<FoodItem>`

Actual equality
logic here

Implement
`==`

Call `IEquatable<T>`
`.Equals()`

Implement
`!=`

Call `IEquatable<T>`
`Equals()`

Implement
`object.GetHashCode()`

XOR field hash codes

Great way to
implement equality
for value types

Don't do it this way
for reference types

Summary

- ‘Override/implement equality’ = Implement custom equality behaviour
- Implementing equality usually good for value types:
- Procedure:
  - Override `object.Equals()`.
  - Implement `IEquatable<T>.Equals()`.
  - Implement == and != overloads.
  - Override `object.GetHashCode()`.
- Keep equality logic in one place.