Tune in to C#

Mads Torgersen

C# Lead Designer

Microsoft

And now for something completely different…

“This is not an argument!”

- ➢ Nullable reference types in C# 8.0

“It is scratched!”

- ➢ Records in C# 9.0

“You are all individuals!”

- ➢ Roles and extensions in the future maybe

You are all individuals!

Static interface members:

- ➢ Interfaces prescribe ***static*** members for implementing types

Roles:

- ➢ View existing ***values*** as having additional members or types

Extensions:

- ➢ View existing ***types*** as having additional members or types

Static Members in Interfaces

```
interface IMonoid<T>
{
    static T Zero { get; }
    static T operator +(T t1, T t2);
}

struct Int32 : …, IMonoid<Int32>
{
    …
    public static int Zero => 0;
}

public static T AddAll<T>(T[] ts) where T : IMonoid<T>
{
    T result = T.Zero;
    foreach (T t in ts) { result += t; }
    return result;
}

int sixtyThree = AddAll(new [] { 1, 2, 4, 8, 16, 32 });
```

Problem: Interface overload

```
interface IMonoid<T>
{
    static T Zero { get; }
    static T operator +(T t1, T t2);
}

struct Int32 : …, IMonoid<Int32>, IGroup<Int32>, IRing<Int32>,…
{
    …
    public static int Zero => 0;
}

public static T AddAll<T>(T[] ts) where T : IMonoid<T>
{
    T result = T.Zero;
    foreach (T t in ts) { result += t; }
    return result;
}

int sixtyThree = AddAll(new [] { 1, 2, 4, 8, 16, 32 });
```

Problem: Multiple implementations

```
interface IMonoid<T>
{
    static T Zero { get; }
    static T operator +(T t1, T t2);
}

struct Int32 : …, IMonoid<Int32>
{
    …
    public static int Zero => 0;
}
```

```
struct Int32 : …, IMonoid<Int32>
{
    …
    public static int operator +(int x, int y) => x * y;
    public static int Zero => 1;
}
```

```
public static T AddAll<T>(T[] ts) where T : IMonoid<T>
{
    T result = T.Zero;
    foreach (T t in ts) { result += t; }
    return result;
}

int sixtyThree = AddAll(new [] { 1, 2, 4, 8, 16, 32 });
```

Problem: Access

```
interface IMonoid<T>
{
    static T Zero { get; }
    static T operator +(T t1, T t2);
}

struct Int32 : …, IMonoid<Int32>
{
    …
    public static int Zero = 0;
}

public static T AddAll<T>(T[] ts) where T : IMonoid<T>
{
    T result = T.Zero;
    foreach (T t in ts) { result += t; }
    return result;
}

int sixtyThree = AddAll(new [] { 1, 2, 4, 8, 16, 32 });
```

Roles

```
interface IMonoid<T>
{
    static T Zero { get; }
    static T operator +(T t1, T t2);
}

role IntMonoid : Int32
{
    …
    public static int Zero => 0;
}

public static T AddAll<T>(T[] ts) where T : IMonoid<T>
{
    T result = T.Zero;
    foreach (T t in ts) { result += t; }
    return result;
}

IntMonoid[] values = new [] { 1, 2, 4, 8, 16, 32 });
```

Roles

```
interface IMonoid<T>
{
    static T Zero { get; }
    static T operator +(T t1, T t2);
}

role IntMonoid : Int32, IMonoid<Int32>
{
    …
    public static int Zero => 0;
}

public static T AddAll<T>(T[] ts) where T : IMonoid<T>
{
    T result = T.Zero;
    foreach (T t in ts) { result += t; }
    return result;
}

int sixtyThree = AddAll<IntMonoid>(new [] { 1, 2, 4, 8, 16, 32 });
```

Roles

```
interface IMonoid<T>
{
    static T Zero { get; }
    static T operator +(T t1, T t2);
}

role IntMonoid : Int32, IMonoid<Int32>
{
    …
    public static int Zero => 0;
}

public static T AddAll<T>(T[] ts) where T : IMonoid<T>
{
    T result = T.Zero;
    foreach (T t in ts) { result += t; }
    return result;
}

int sixtyThree = AddAll(new IntMonoid[] { 1, 2, 4, 8, 16, 32 });
```

Extensions

```
interface IMonoid<T>
{
    static T Zero { get; }
    static T operator +(T t1, T t2);
}

extension IntMonoid : Int32, IMonoid<Int32>
{
    …
    public static int Zero => 0;
}

public static T AddAll<T>(T[] ts) where T : IMonoid<T>
{
    T result = T.Zero;
    foreach (T t in ts) { result += t; }
    return result;
}

int sixtyThree = AddAll(new [] { 1, 2, 4, 8, 16, 32 });
```