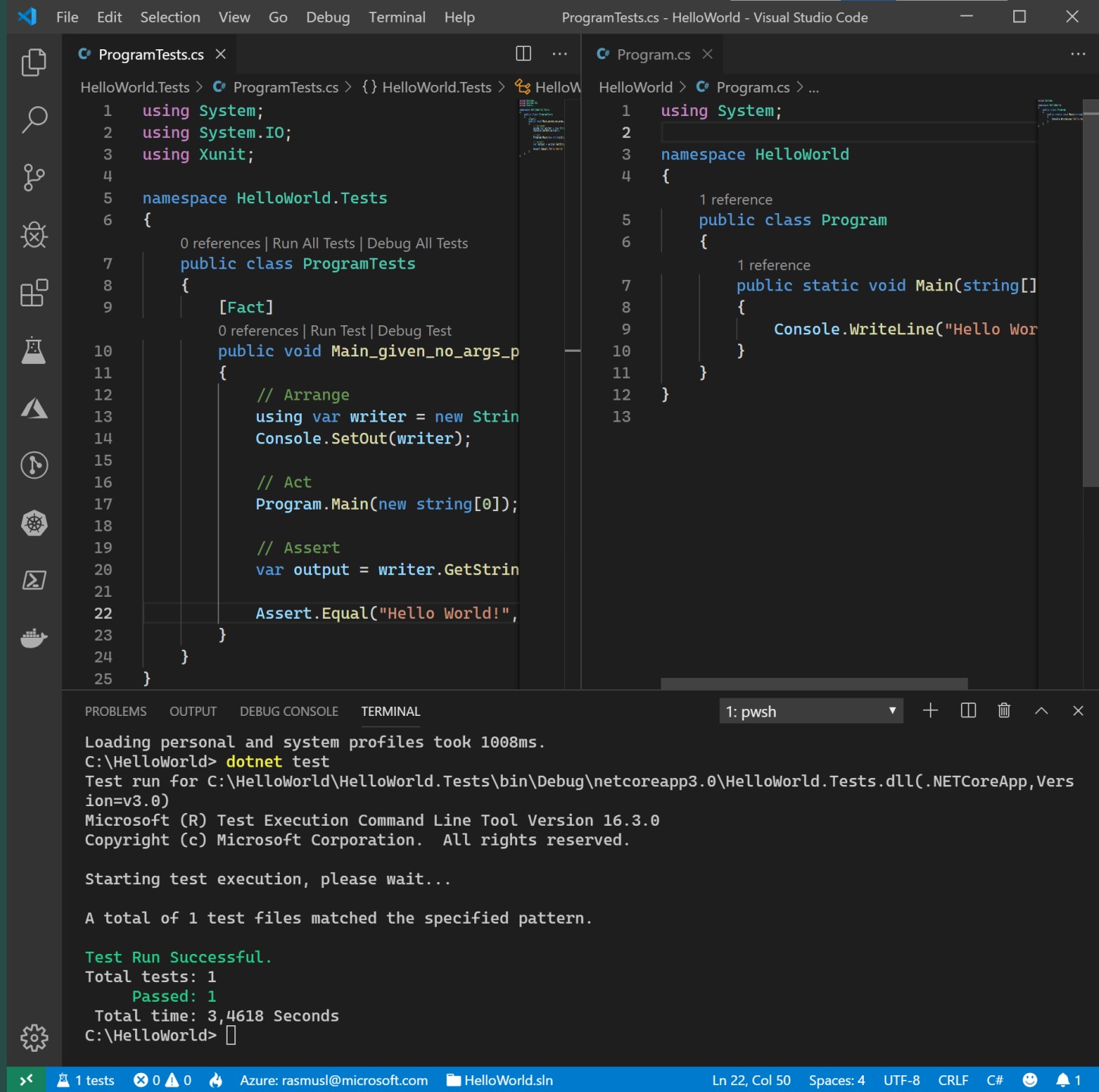
C♯
Lambdas and LINQ

Rasmus Lystrøm
Associate Professor
ITU
rnie@itu.dk



Agenda

Properties
Anonymous methods
Delegates
Lambda expressions
Local functions
Anonymous types
Tuples
Extension methods
LINQ

Properties

Properties 1/3

```
public class City
{
    public int Id { get; set; }
    public string Name { get; private set; }
}
```

Properties 2/3

```
public class City
{
    private int _id;
    public int Id { get => _id; set => _id = value; }

    private string _name;
    public string Name
    {
        get { return _name; }
        set
        {
            _name = value;
        }
    }
}
```

Properties 3/3

```
public class City
{
    private int _id;
    public int Id
    {
        get => _id;
        set
        {
            // Place setter validation logic here if required
            _id = value;
        }
    }
}
```

Delegates

Delegates – Building block for Higher-order functions

```
public delegate int BinaryOperation(int x, int y);

static void Main(string[] args)
{
    var add = new BinaryOperation(
        delegate (int x, int y)
        {
            return x + y;
        }
    );
}
```

Delegates demo

Lambda expressions

```
Action<string> write = s => Console.WriteLine(s);

Predicate<City> b = c => c.Name.StartsWith("B");

Func<int, int> square = a => a * a;
```

(Local functions)

```
static void Main(string[] args)
{
    int square(int a) { return a * a; };

    Console.WriteLine(square(16));
}
```

Anonymous types

```
var question = new
{
    Title = "The answer...?",
    Answer = 42
};
```

(Tuples)

```
var s = Tuple.Create("Clark Kent", "Superman");

var b = ("Bruce Wayne", "Batman");

var f = (name: "Barry Allen", alterEgo: "The Flash");

IEnumerable<(float x, float y)> GenerateCoordinates()
{
    yield return (1.3f, 23.45f);
}
```

Extension methods 1/2

```
IEnumerable<City> cities = new[]
{
    new City { Id = 1, Name = "Berlin" },
    new City { Id = 2, Name = "Hamburg" },
    new City { Id = 3, Name = "Frankfurt" }
};

var count = cities.Count();

var sorted = cities.OrderBy(c => c.Name);

var filtered = cities.Where(c => c.Name.Contains("i"));

var pick = cities.FirstOrDefault(c => c.Id == 2);
```

Extension methods 2/2

```
public static class Extensions
{
    public static int WordCount(this string str)
    {
        return str.Split(
            new char[] { ' ', '.', '?' },
            StringSplitOptions.RemoveEmptyEntries).Length;
    }
}
```

Data

```
IEnumerable<City> cities = new[]
{
    new City { Id = 1, Name = "Berlin" },
    new City { Id = 2, Name = "Hamburg" },
    new City { Id = 3, Name = "Frankfurt" }
};
```

LINQ

```
var sorted = from c in cities
             where c.Name.Contains("i")
             orderby c.Name descending
             select new { Name = c.Name };
```

LINQ demo

Extension methods

Extension Methods

```
var sorted = cities.Where(c => c.Name.Contains("i"))
                   .OrderByDescending(c => c.Name)
                   .Select(c => new { Name = c.Name });
```