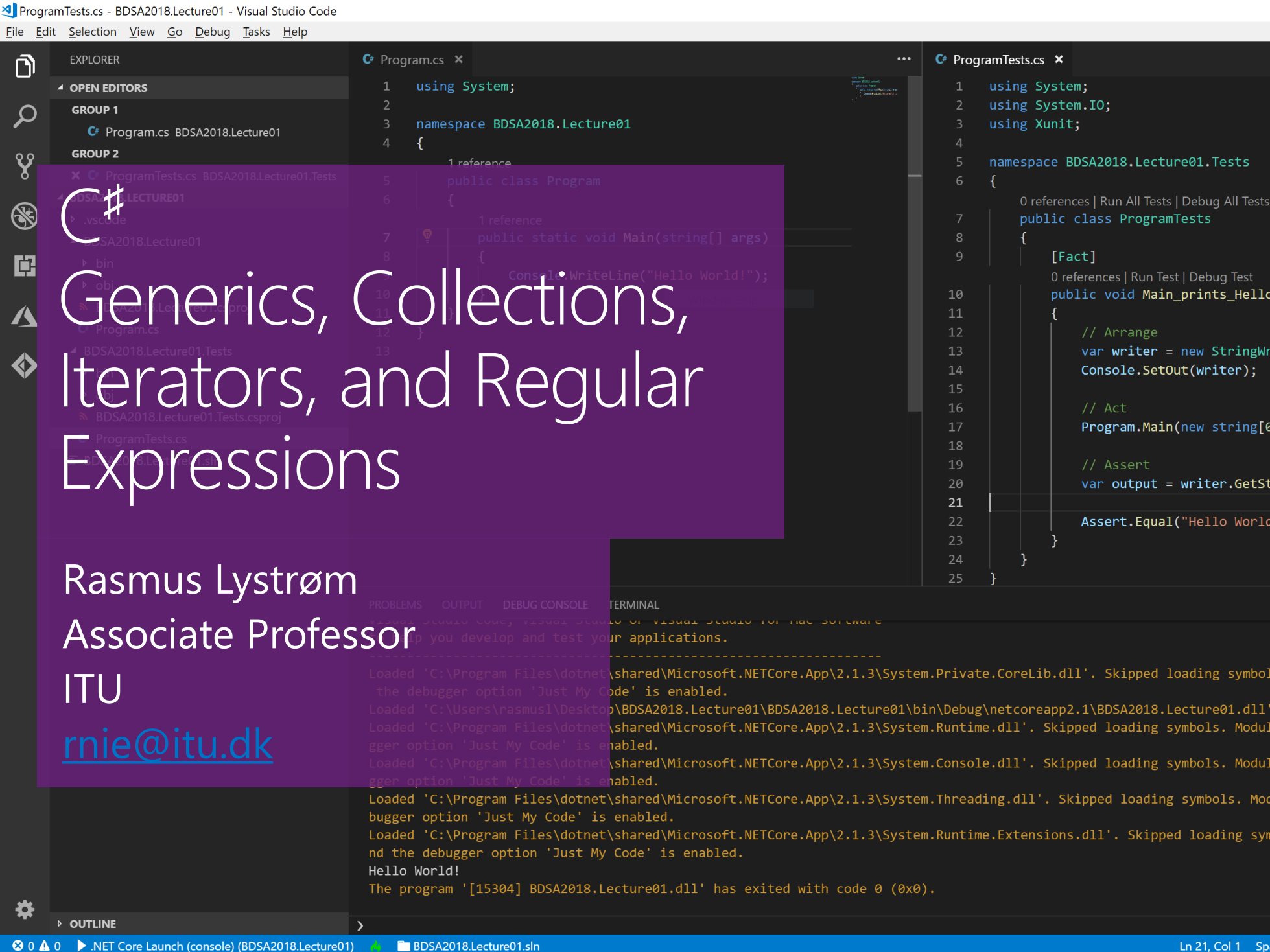


C#
Generics, Collections, Iterators, and Regular Expressions

Rasmus Lystrøm
Associate Professor
ITU
rnie@itu.dk

Agenda

Generics
Iterators, enumerators, and enumerations
Collections
Regular Expressions

Generics

"Parametric Polymorphism"
Built-in
Create your own?
Type Constraints
(Co- and contravariance)

ArrayList → List<T>

```
// Non-generic
IList list = new ArrayList();
list.Add("hello");
var s = (string)list[0];

// Generic
IList<string> list = new List<string>();
list.Add("hello");
var s = list[0];
```

Create your own generic class

```
public class MyStack<T>
{
    public void Clear() { }

    public T Peek() { }

    public T Pop() { }

    public void Push(T item) { }
}
```

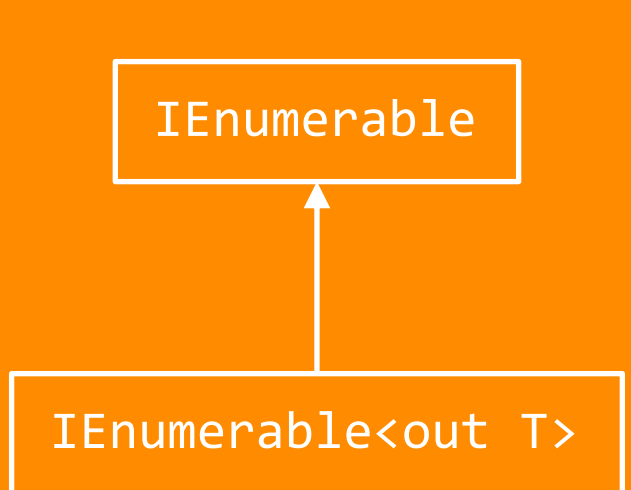
Create your own generic method

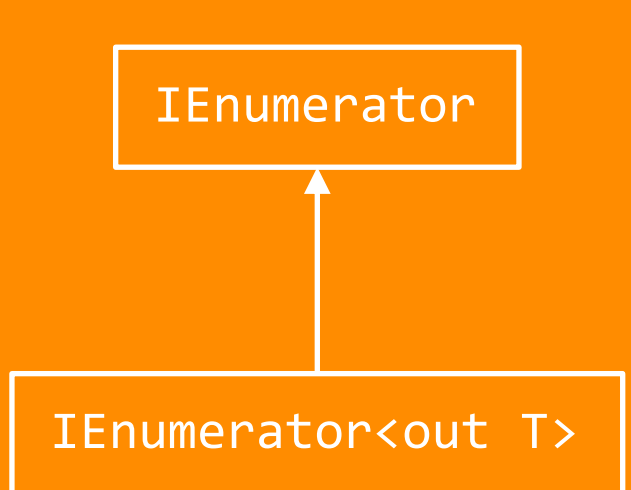
```
public string Serialize<T>(T obj) {}
```

Type constraints

```
public class MyConstrainedGenericClass<T>
    where T : class { }

public class MyConstrainedGenericClass<T>
    where T : struct { }

public class MyConstrainedGenericClass<T1, T2>
    where T1 : Foo where T2 : IBar { }

public T2 MyConstrainedMethod<T1, T2>(T1 item)
    where T1 : Foo
    where T2 : IBar {}
```

Iterators





Producer

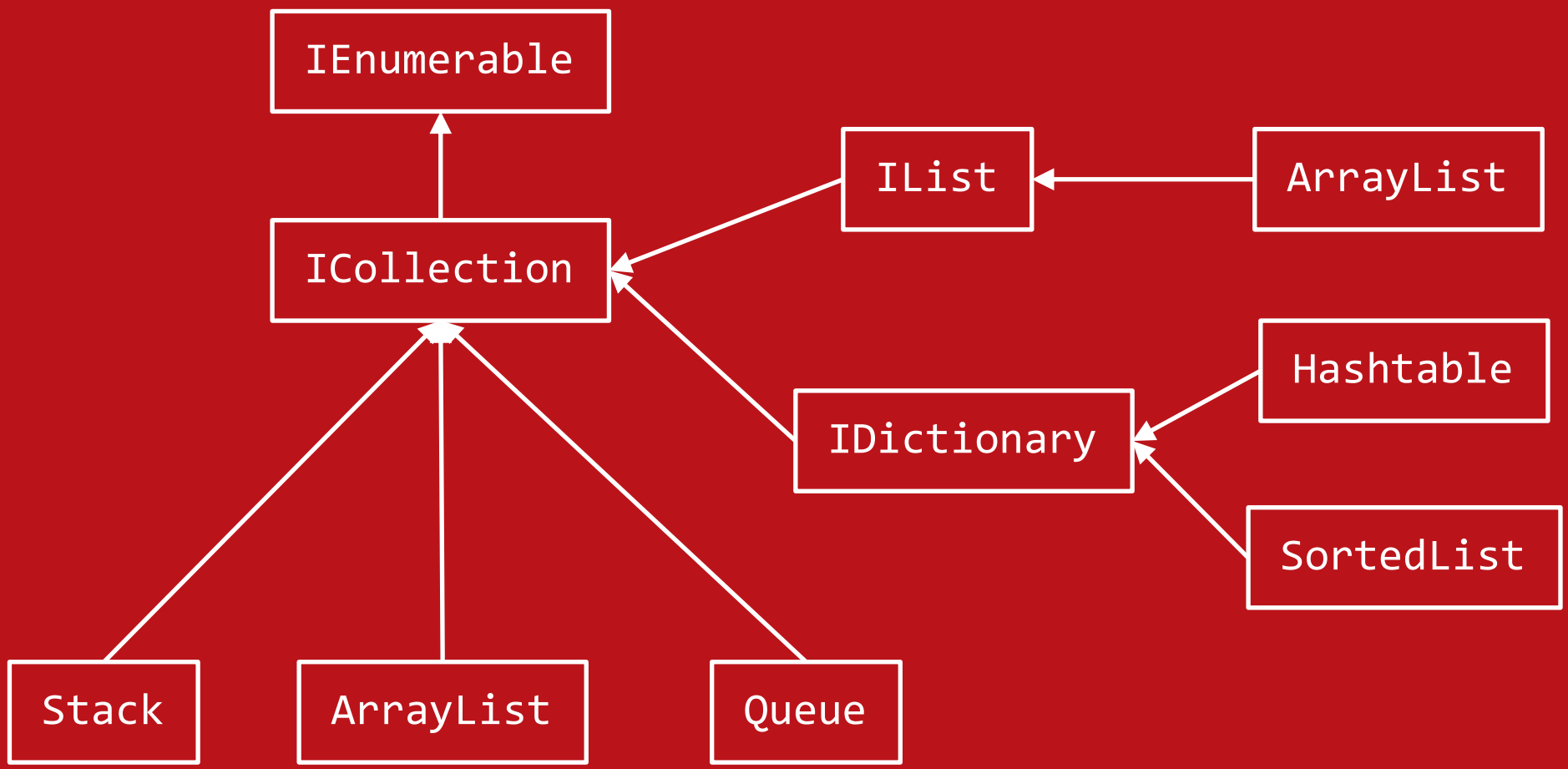
Building block for Linq

```
yield return T;
```

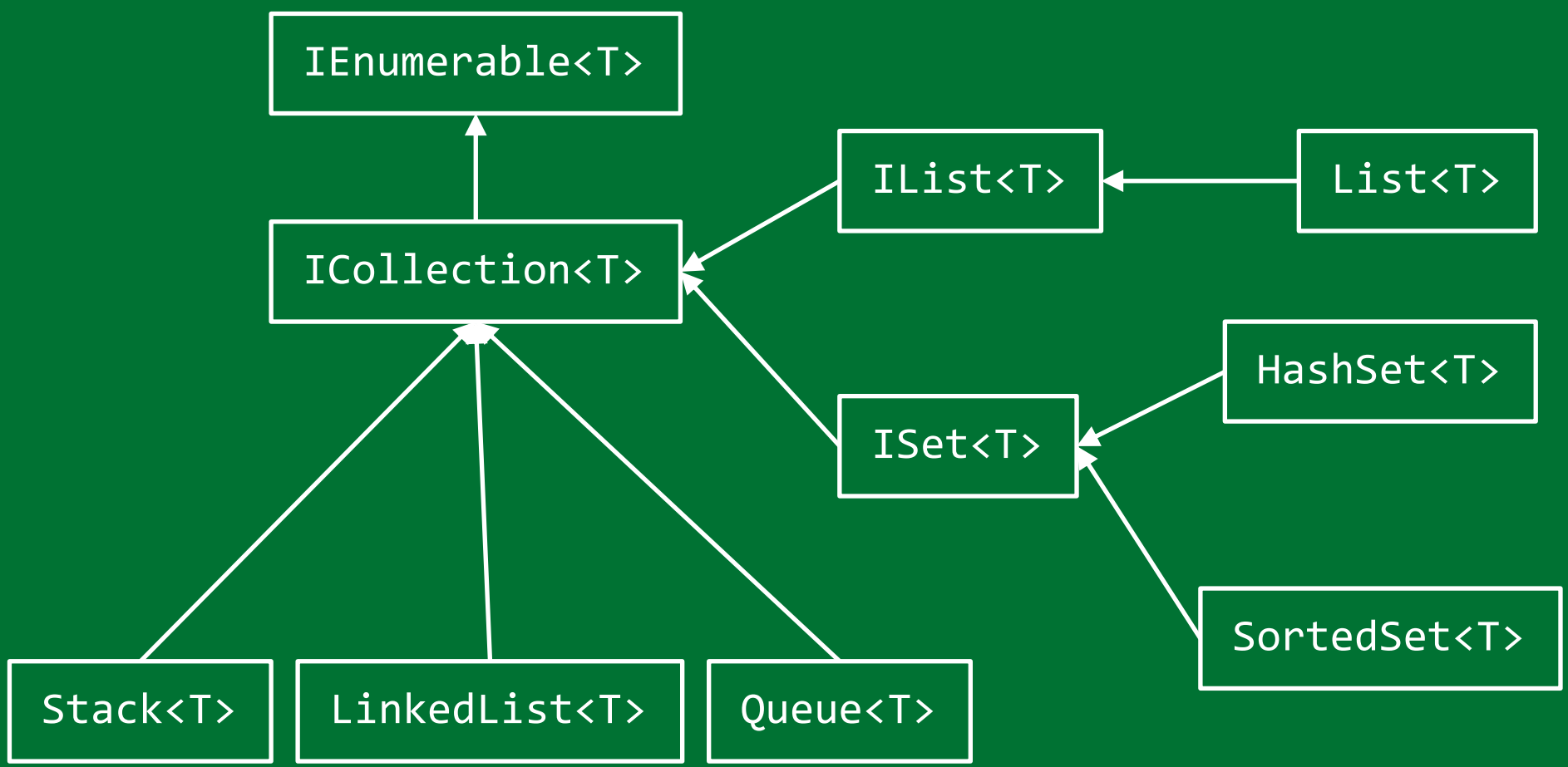
```
yield break;
```

```
foreach (var item in items)
{
}
```

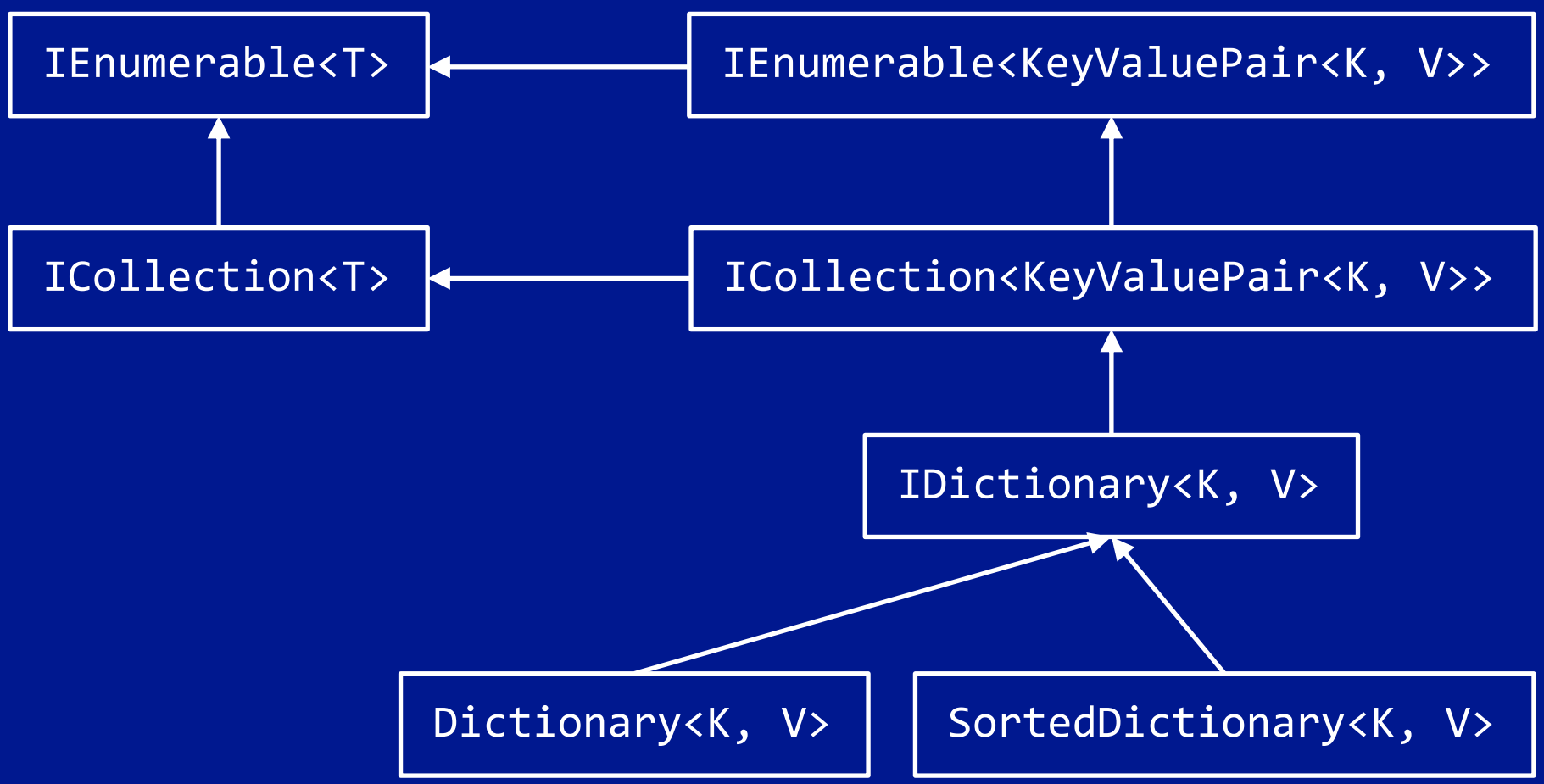
System.Collections



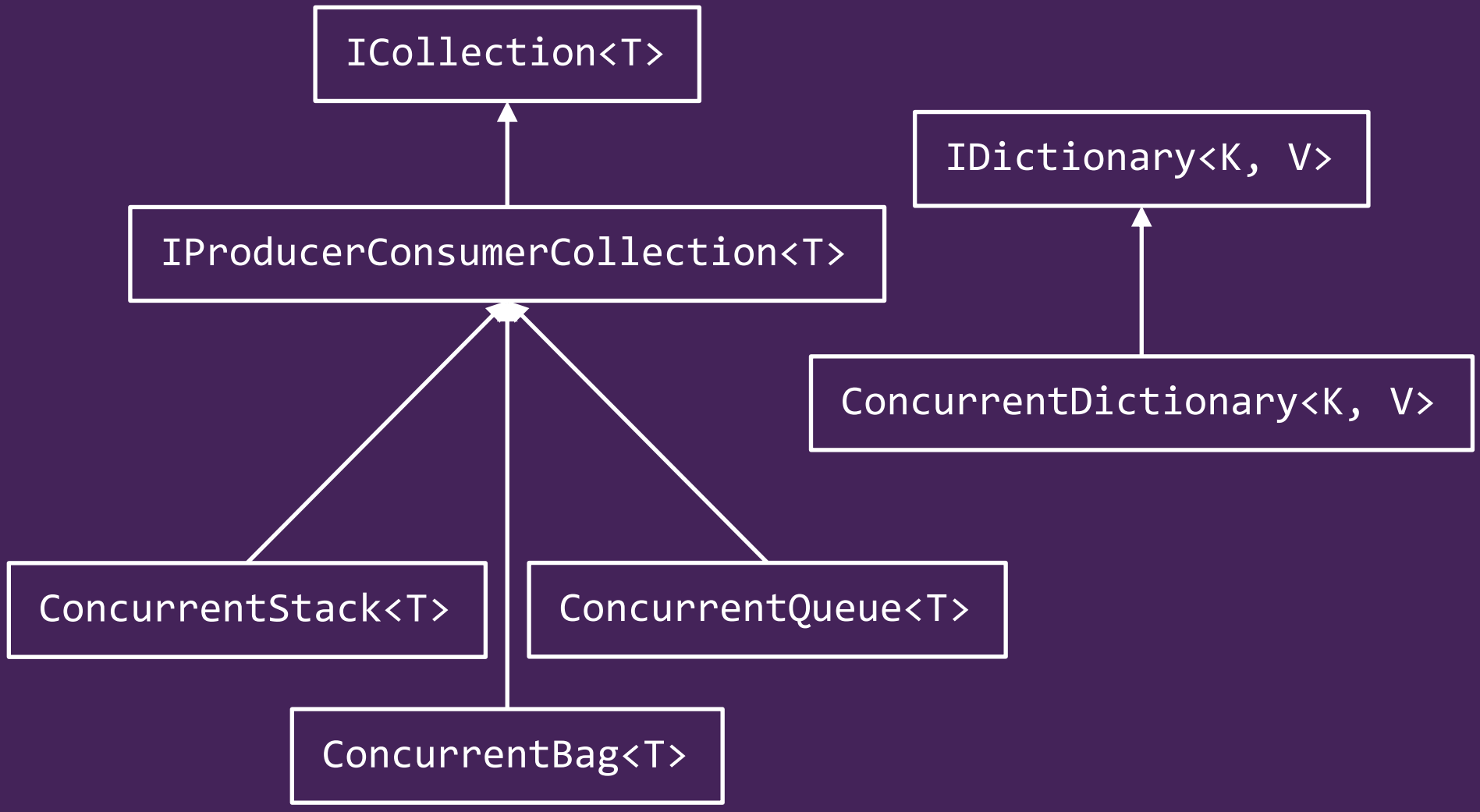
System.Collections.Generic



System.Collections.Generic 2



System.Collections.Concurrent



Regular Expressions

`*`	Zero or more times the previous character
`+`	Once or more times the previous character
`?`	Zero or one time the previous character
`.`	Any single character (not \n)
`\s`	Any whitespace character (e.g. tab)
`\S`	Any non-whitespace character
`\b`	Word boundary
`\B`	Any non-word boundary position
`\w`	Any word character (a-z, A-Z, 0-9)
`\W`	Any non-word character
`^`	Start of the input text
`$`	End of the input text

Regular Expressions

`[1c]`	matches character '1' or 'c'
`[a-z]`	matches all lower-case letters
`[a-zA-Z]`	matches all letters
`[0-9]+`	matches integer numbers
`[0-9]+\.[0-9]+`	matches a floating point
`[0-2][0-9]:[0-5][0-9]`	matches a time e.g. 12:34

http://msdn.microsoft.com/en-us/library/az24scfc.aspx