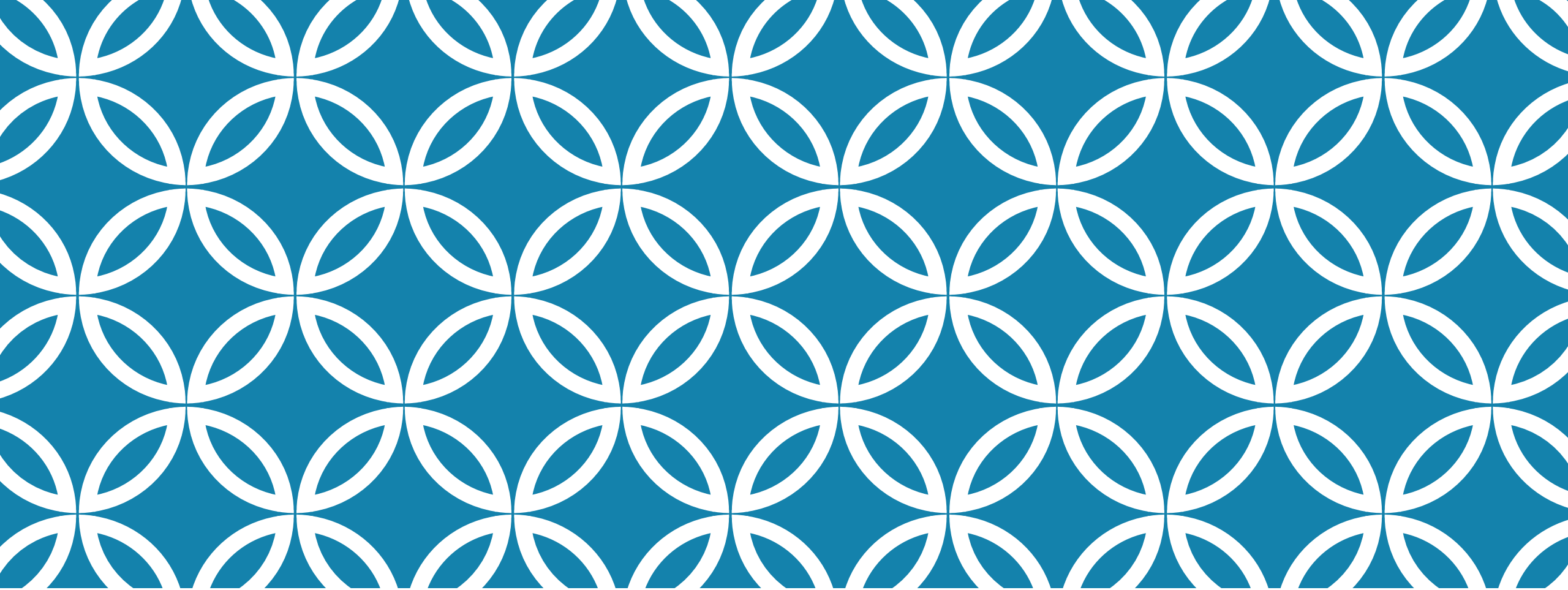
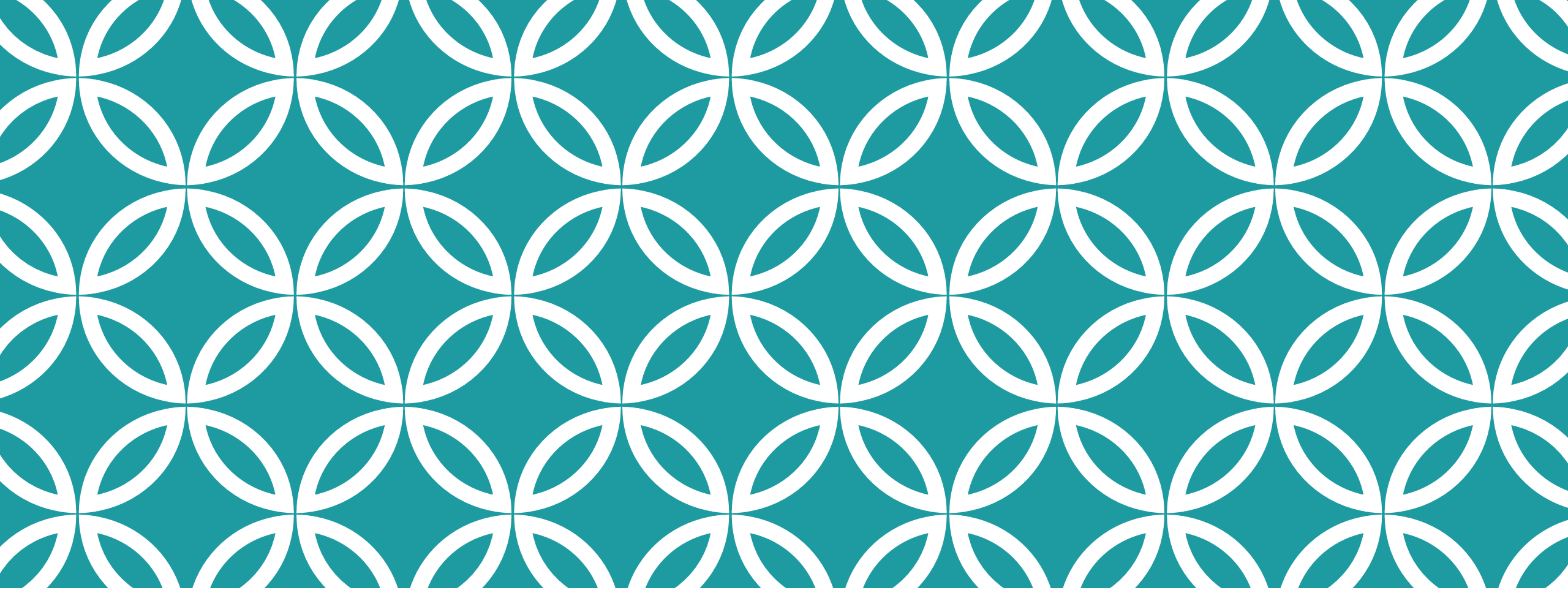


DELEGATES
LAMBDA EXPRESSIONS
GAMMEL EKSAMENSOPGAVE
C# OG UNITY

Grundlæggende programmering
Lektion 8



DELEGATES

Referencer til metoder

DELEGATES

En delegate er en reference type variabel, der indeholder en reference til en metode. Referencen kan ændres ved runtime.

Delegates er specielt brugt til at gennemføre events og call-back metoder.

Alle delegates implicit afledt af System.Delegate klassen.

Delegate deklarationen bestemmer metoderne der kan henvises til af delegate, da den kun kan referere til en metode der har den same signatur.

```
public delegate int MyDelegate (string s);
```

kan således kun bruges til at referere metoder der har en enkelt *string* parameter og returnerer en *int* type variabel.

Den generelle declaration er således:

```
delegate <return type> <delegate-name>
<parameter list>
```

DELEGATES INSTANTIERING

Instantiering af delegates

Når en delegate type erklæres skal et delegate objekt oprettes med den new keyword og være forbundet med en særlig metode.

Når en delegate laves angives argumentet videre til nye udtryk på en sådan måde at det ligner et metode kald, men uden argumenterne til metoden.

```
using System;

delegate int NumberChanger(int n);
namespace DelegateAppl
{
    0 references
    class TestDelegate
    {
        static int num = 10;
        1 reference
        public static int AddNum(int p)
        {
            num += p;
            return num;
        }

        1 reference
        public static int MultNum(int q)
        {
            num *= q;
            return num;
        }
        2 references
        public static int getNum()
        {
            return num;
        }

        0 references
        static void Main(string[] args)
        {
            //create delegate instances
            NumberChanger nc1 = new NumberChanger(AddNum);
            NumberChanger nc2 = new NumberChanger(MultNum);

            //calling the methods using the delegate objects
            nc1(25);
            Console.WriteLine("Value of Num: {0}", getNum());
            nc2(5);
            Console.WriteLine("Value of Num: {0}", getNum());
            Console.ReadKey();
        }
    }
}
```

DELEGATES MULTICASTING

Multicasting en delegate

Delegate objekter kan sammensættes ved hjælp af "+" operator.

En sammensat delegeret kalder de to delegates den blev sammensat af.

Kun delegates af samme type kan være sammensat.

"-" operatoren kan bruges til at fjerne en komponent delegate fra en sammensat delegate.

Ved hjælp af denne egenskab kan man oprette en invocation (påkalde) liste over metoder, der vil blive kaldt, når en delegate kaldes. Dette kaldes multicasting af en delegeret.

```
using System;

delegate int NumberChanger(int n);
namespace DelegateAppl
{
    0 references
    class TestDelegate
    {
        static int num = 10;
        1 reference
        public static int AddNum(int p)
        {
            num += p;
            return num;
        }

        1 reference
        public static int MultNum(int q)
        {
            num *= q;
            return num;
        }

        1 reference
        public static int getNum()
        {
            return num;
        }

        0 references
        static void Main(string[] args)
        {
            //create delegate instances
            NumberChanger nc;
            NumberChanger nc1 = new NumberChanger(AddNum);
            NumberChanger nc2 = new NumberChanger(MultNum);
            nc = nc1;
            nc += nc2;

            //calling multicast
            nc(5);
            Console.WriteLine("Value of Num: {0}", getNum());
            Console.ReadKey();
        }
    }
}
```

DELEGATES BRUG

Multicasting en delegate

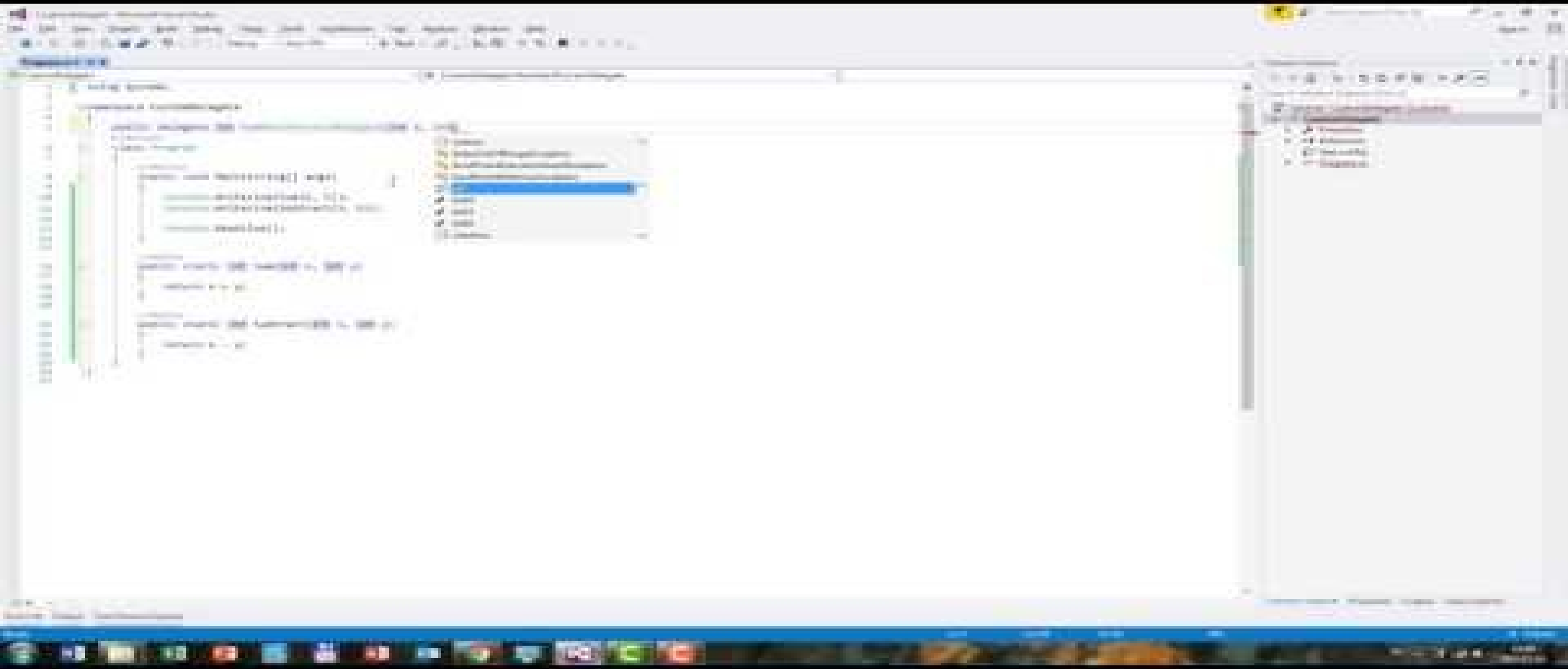
Delegate objekter kan sammensættes ved hjælp af "+" operator.

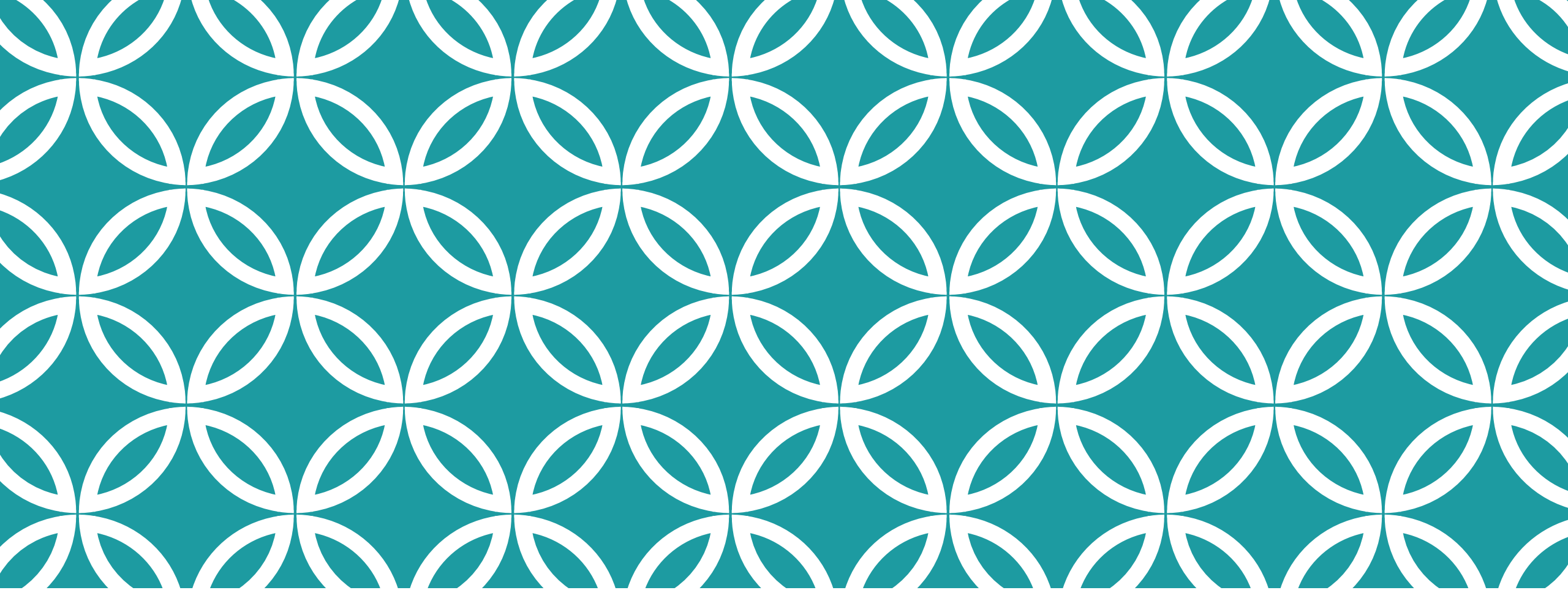
En sammensat delegeret kalder de to delegates den blev sammensat af.

Kun delegates af samme type kan være sammensat.

"-" operatoren kan bruges til at fjerne en komponent delegate fra en sammensat delegate.

Ved hjælp af denne egenskab kan man oprette en invocation (påkalde) liste over metoder, der vil blive kaldt, når en delegate kaldes. Dette kaldes multicasting af en delegeret.





LAMBDA EXPRESSIONS

En anonym funktion

LAMBDA EXPRESSIONS

Et lambda udtryk er en anonym funktion, som kan bruges til at oprette delegates eller udtryk træ typer.

Ved at bruge lambda udtryk kan man skrive lokale funktioner, der kan overføres som argumenter eller returneres som værdien af funktionskald.

Lambda udtryk er særligt nyttigt for at skrive LINQ query udtryk.

For at oprette et lambda udtryk angiver man inputparametre (hvis nogen) på venstre side af lambda operatoren =>, og man sætter udtryk eller udsagn blokken på den anden side

- Lambda udtrykket `x => x * x` angiver en parameter, x og giver den værdien af x potens. Du kan tildele dette udtryk til en delegeret type.

LAMBDA EXPRESSIONS

```
1   using System;
2   using System.Collections.Generic;
3
        0 references
4   class Program
5   {
            0 references
6       static void Main()
7       {
8           List<int> elements = new List<int>() { 10, 20, 31, 40 };
9           // ... Find index of first odd element.
10          int oddIndex = elements.FindIndex(x => x % 2 != 0);
11          Console.WriteLine(oddIndex);
12          Console.ReadKey();
13      }
14  }
```

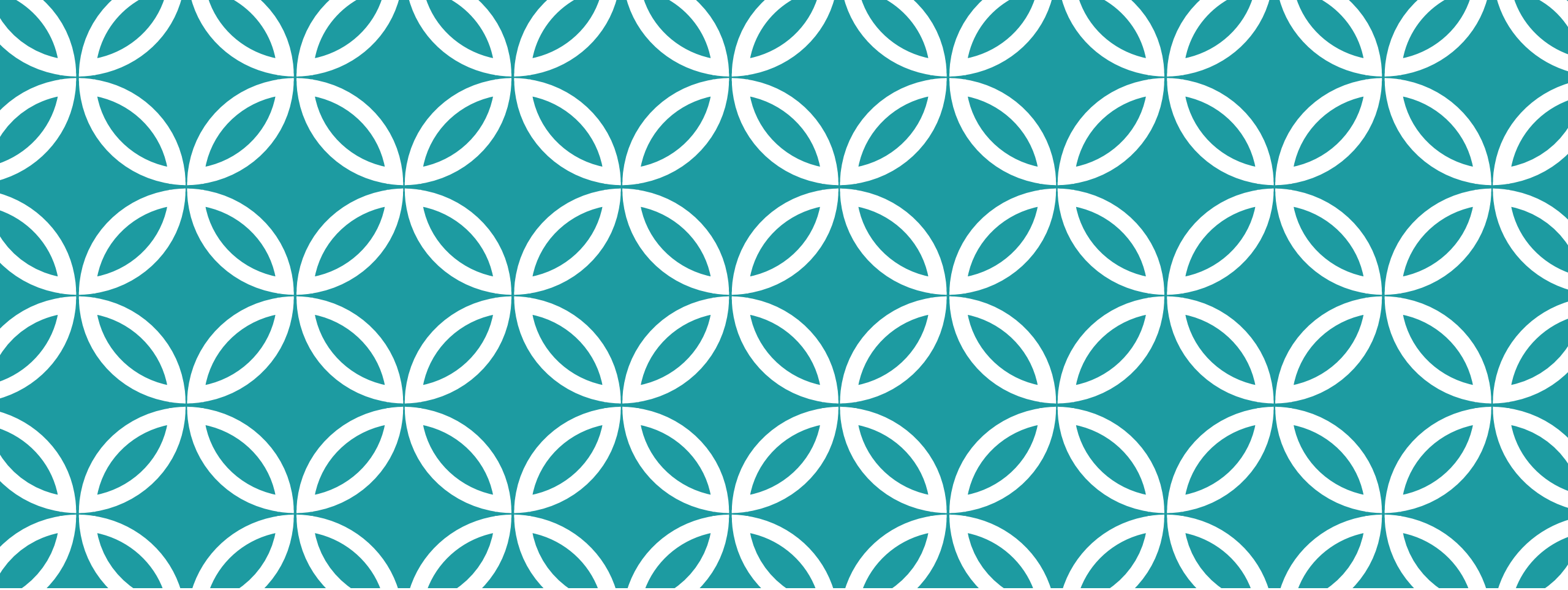
Getting Started with

Lambda Expressions

//c0deporn;

EKSAMENSOPGAVERNE SIDSTE HOLD

Gennemgang i fællesskab



C# OG UNITY

Vises i andet slideshow

LEKTIE

Kig på dette til næste gang

LEKTIE

Se og løs opgaverne til https://mva.microsoft.com/en-US/training-courses/programming-in-c-jump-start-14254 - 06

Læs:

- https://www.tutorialspoint.com/csharp/csharp_exception_handling.htm
- https://www.tutorialspoint.com/csharp/csharp_generics.htm

KILDER

Materiale benyttet i denne lektion
Noget af det er udover pensum-listen!

KILDER

Delegates

https://www.tutorialspoint.com/csharp/csharp_delegates.htm

https://youtu.be/mlzYgppHXXM

Lambda expressions

https://msdn.microsoft.com/en-us/library/bb397687.aspx

https://msdn.microsoft.com/en-us/library/bb397675.aspx

https://www.dotnetperls.com/lambda

https://youtu.be/xev-kNmz_a0