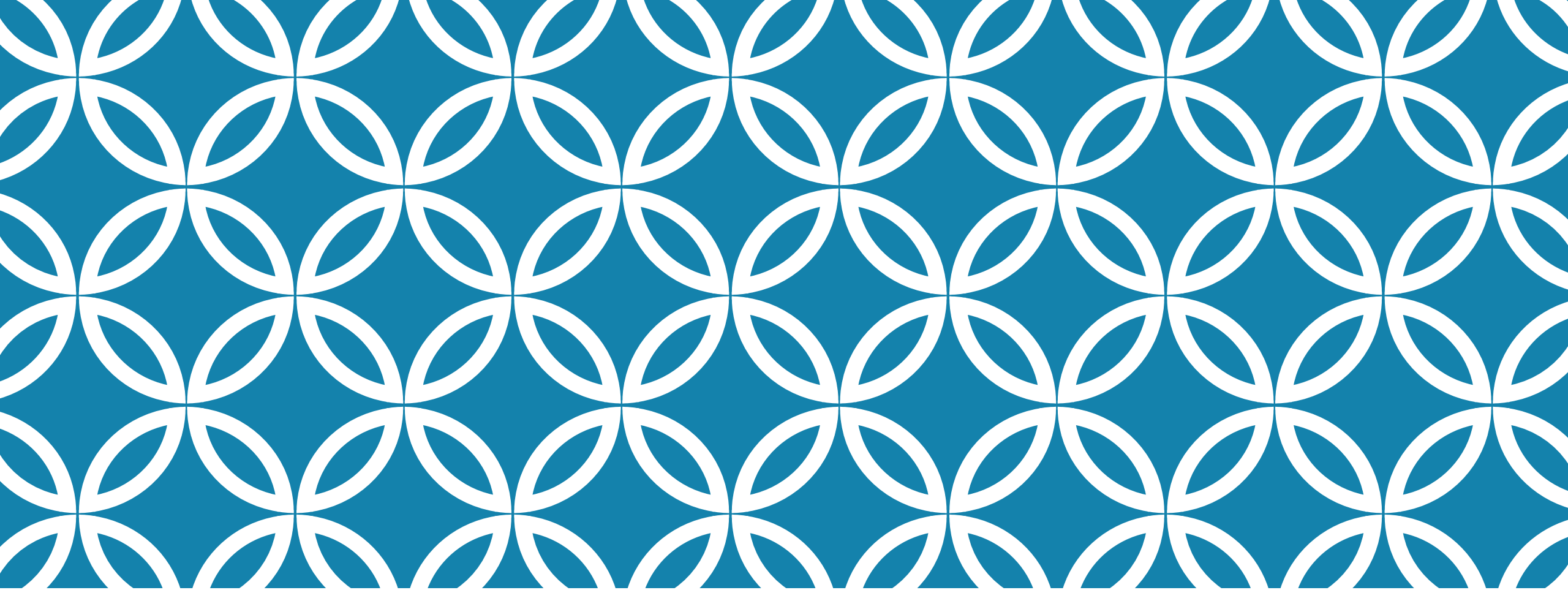
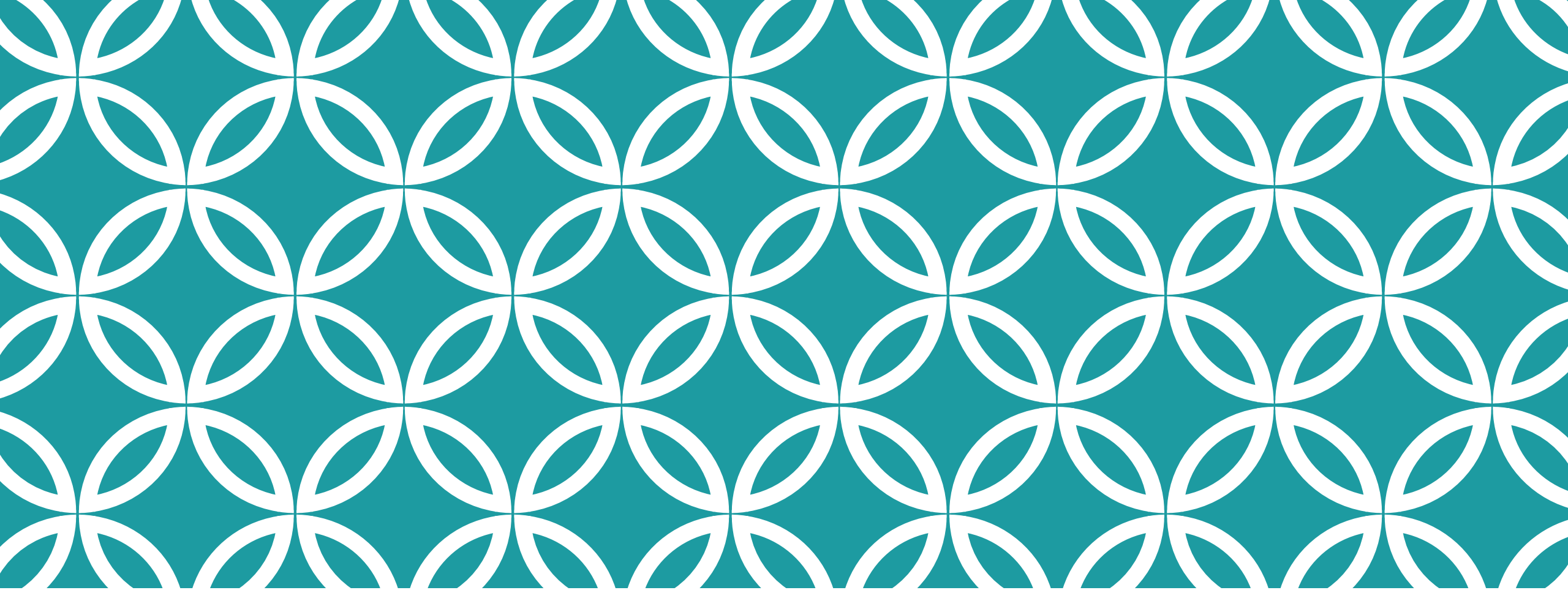


# EXCEPTION HANDLING
# VALIDATION
# GENERICS
# PRAKSIS EKSEMPEL

Grundlæggende programmering
Lektion 7



# EXCEPTION HANDLING

Problemer opstår så der må gøres noget

# EXCEPTION HANDLING

En undtagelse (exception) er et problem, der opstår under udførelsen af et program.

I C# er en undtagelse er en reaktion på en usædvanlig hændelse, der opstår, mens et program kører, f.eks. et forsøg på at dividere med nul.

Exceptions sørger for en måde at transfer kontrol fra en del af et program til en anden.

C# exception handling bygger på fire keywords: **try**, **catch**, **finally** og **throw**.

- **try**: En try blok identificerer en blok af kode, for hvilken særlige undtagelser er aktiveret. Det efterfølges af en eller flere catch blokke.
- **catch**: Et program fanger en undtagelse med en exception handler det sted i programmet, hvor du ønsker at håndtere problemet. Catch keyword angiver fangst af en undtagelse.
- **finally**: Finally blokken anvendes til at udføre et givet sæt udsagn, om en undtagelse er thrown eller ikke thrown. For eksempel, hvis du åbner en fil, skal det være lukket om en undtagelse hæves eller ej.
- **throw**: Et program kaster en undtagelse, når et problem dukker op. Dette gøres ved hjælp af throw keyword.

# EXCEPTION HANDLING SYNTAKS

## Syntaks

Hvis en blok udløser en exception fanger en metode en exception gennem en kombination af try og catch keywords.

En try/catch blok placers omrking koden, der kan generere en exception. Kode i en try/catch omtales som protected (beskyttet) kode.

Du kan liste flere catch statements til at fange forskellige typer af undtagelser i tilfælde af at en blok trigger mere end en enkelt undtagelse i forskellige situationer.

```
try
{
   // statements causing exception
}
catch( ExceptionName e1 )
{
   // error handling code
}
catch( ExceptionName e2 )
{
   // error handling code
}
catch( ExceptionName eN )
{
   // error handling code
}
finally
{
   // statements to be executed
}
```

# EXCEPTION HANDLING KLASSER

## Exception klasser i C#

C# exceptions repræsenteres af classes.

Exception classes i C# er primært direkte eller indirekte afledt af **System.Exception** klassen.

- Nogle exception classes afledt af System.Exception klassen er **System.ApplicationException** og **System.SystemException**.

- **System.ApplicationException** klassen understøtter exceptions genereret af application programmer. Derfor bør exceptions, som defineres af programmørerne, stamme fra denne klasse.
- **System.SystemException** klassen er base klassen for alle prædefinerede system exceptions.

- Nogle af de prædefinerede exception klasser afledt af System.SystemException klassen kan ses i skemaet på næste slide.

# EXCEPTION HANDLING KLASSER

| Exception Class | Description |
|---|---|
| System.IO.IOException | Handles I/O errors. |
| System.IndexOutOfRangeException | Handles errors generated when a method refers to an array index out of range. |
| System.ArrayTypeMismatchException | Handles errors generated when type is mismatched with the array type. |
| System.NullReferenceException | Handles errors generated from deferencing a null object. |
| System.DivideByZeroException | Handles errors generated from dividing a dividend with zero. |
| System.InvalidCastException | Handles errors generated during typecasting. |
| System.OutOfMemoryException | Handles errors generated from insufficient free memory. |
| System.StackOverflowException | Handles errors generated from stack overflow. |

# EXCEPTION HANDLING HÅNDTERING

## Håndtering af exceptions

C # giver en struktureret løsning på håndtering af exceptions i form af try og catch blokke. Ved hjælp af disse blokke bliver kerne-program statements adskilt fra fejl håndterings statements.

Disse fejl håndterings blokke implementers med **try, catch,** og **finally** keywords.

# EXCEPTION HANDLING

```
using System;
namespace ErrorHandlingApplication
{
    3 references
    class DivNumbers
    {
        int result;
        1 reference
        DivNumbers()
        {
            result = 0;
        }
        1 reference
        public void division(int num1, int num2)
        {
            try
            {
                result = num1 / num2;
            }
            catch (DivideByZeroException e)
            {
                Console.WriteLine("Exception caught: {0}", e);
            }
            finally
            {
                Console.WriteLine("Result: {0}", result);
            }
        }
        0 references
        static void Main(string[] args)
        {
            DivNumbers d = new DivNumbers();
            d.division(25, 0);
            Console.ReadKey();
        }
    }
}
```

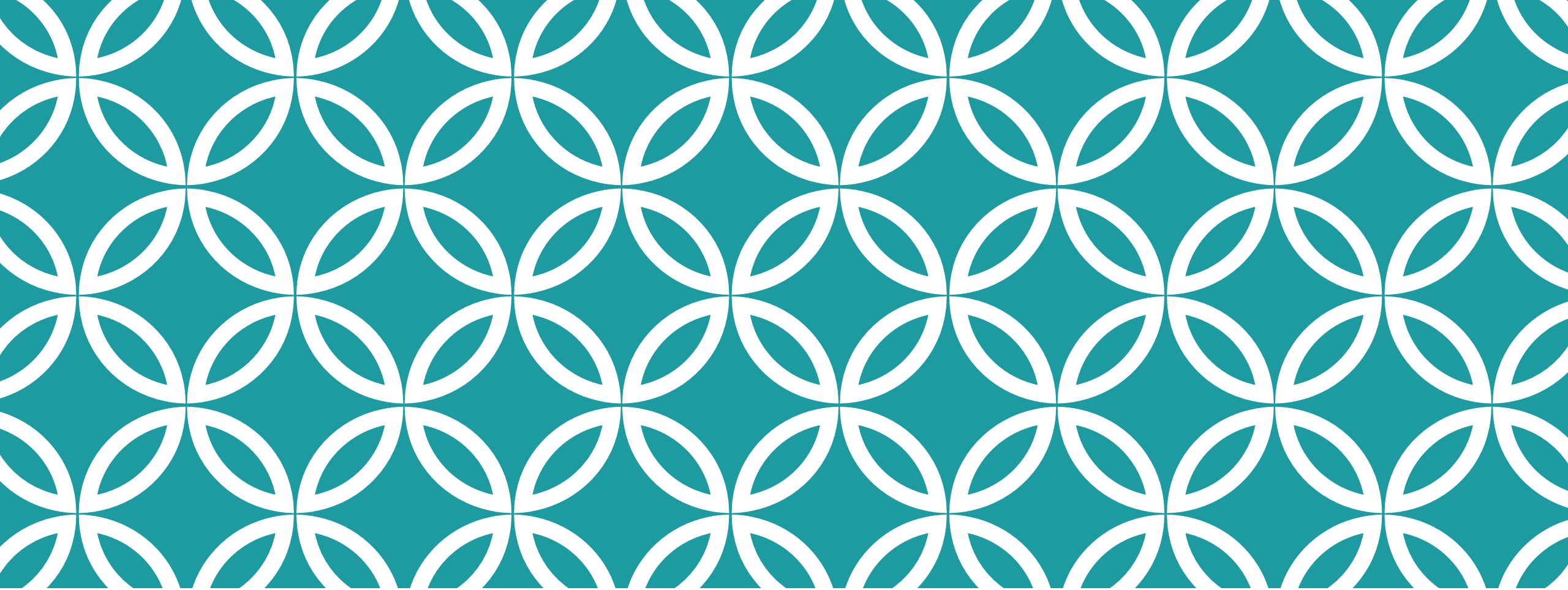
# EXCEPTION HANDLING BRUGERDEFINEREDE

## Brugerdefinerede exceptions

Man kan også definer ens egne exceptions.

Bruger definerede exception klasser er afledt af **Exception** klassen.

```
1  using System;
2  namespace UserDefinedException
3  {
       0 references
4      class TestTemperature
5      {
           0 references
6          static void Main(string[] args)
7          {
8              Temperature temp = new Temperature();
9              try
10             {
11                 temp.showTemp();
12             }
13             catch (TempIsZeroException e)
14             {
15                 Console.WriteLine("TempIsZeroException: {0}", e.Message);
16             }
17             Console.ReadKey();
18         }
19     }
20 }
21
       3 references
22 public class TempIsZeroException : Exception
23 {
       1 reference
24     public TempIsZeroException(string message) : base(message)
25     {
26     }
27 }
28
       2 references
29 public class Temperature
30 {
31     int temperature = 0;
           1 reference
32     public void showTemp()
33     {
34         if (temperature == 0)
35         {
36             throw (new TempIsZeroException("Zero Temperature found"));
37         }
38         else
39         {
40             Console.WriteLine("Temperature: {0}", temperature);
41         }
42     }
43 }
```



# VALIDATION

Tjek af programmet

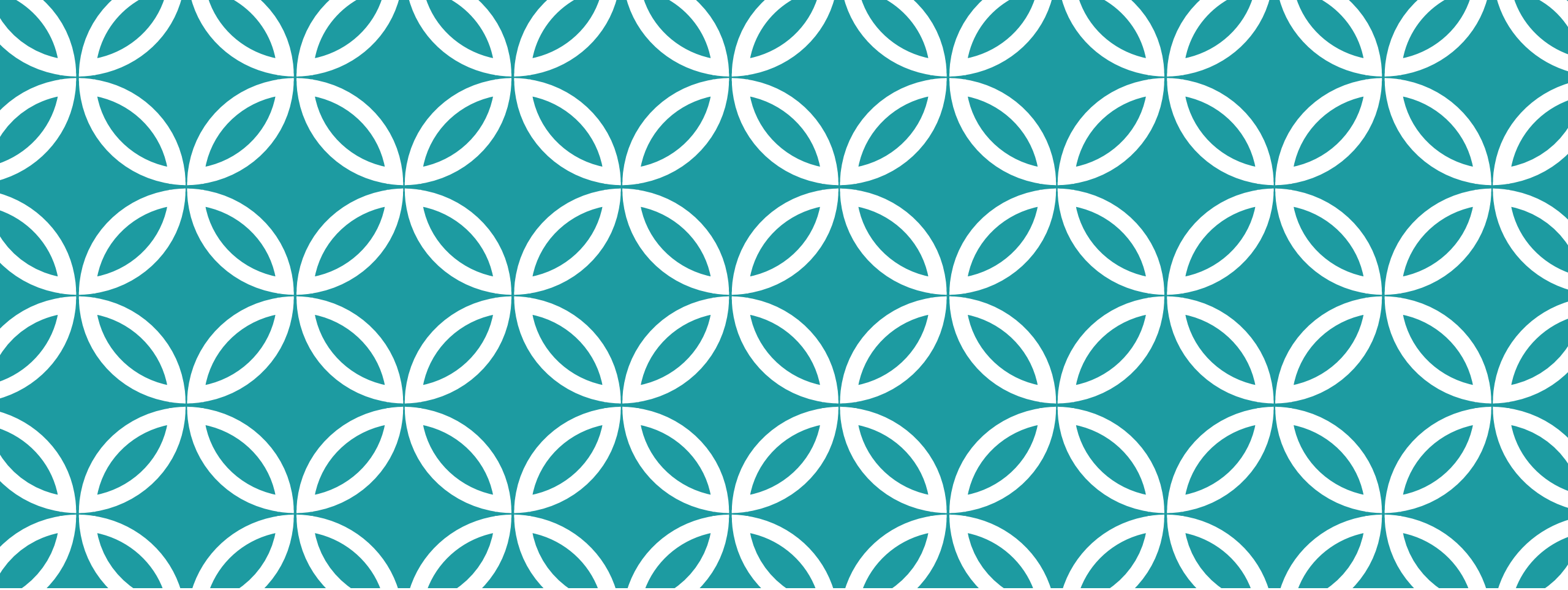
# VALIDATION

Datavalidering er test-værdier introduceret til en app (via en tjeneste, fil eller indtastning af data) mod forventede værdier og ranges.

Man gør det for at:

- Undgå overflow.
- Undgå forkerte resultater.
- Undgå uønskede bivirkninger
- Vejlede systemer eller brugere.
- Forebyg sikkerheds indtrængen.

Compileren validerer, at objekttypen er korrekt, den validerer ikke objektets værdi.



# GENERICS

Lad programmet bestemme når det skal bruges

# GENERICS

Generics gør det muligt at forsinke specifikation af datatypen for program elementer i en klasse eller en metode, indtil de faktisk bliver brugt i programmet.

- Med generics kan man altså skrive en klasse eller metode, der kan arbejde med alle datatyper.

Man skriver specifikationerne for klassen eller den metode, med alternative parametre for datatyper.

- Når compileren møder en constructor for klassen eller et funktionskald for metoden, genererer den kode til at håndtere den specifikke datatype.

```
1  using System;
2  using System.Collections.Generic;
3
4  namespace GenericApplication
5  {
       5 references
6      public class MyGenericArray<T>
7      {
8          private T[] array;
               2 references
9          public MyGenericArray(int size)
10         {
11             array = new T[size + 1];
12         }
13
               2 references
14         public T getItem(int index)
15         {
16             return array[index];
17         }
18
               2 references
19         public void setItem(int index, T value)
20         {
21             array[index] = value;
22         }
23     }
24
       0 references
25     class Tester
26     {
           0 references
27         static void Main(string[] args)
28         {
29
30             //declaring an int array
31             MyGenericArray<int> intArray = new MyGenericArray<int>(5);
32
33             //setting values
34             for (int c = 0; c < 5; c++)
35             {
36                 intArray.setItem(c, c * 5);
37             }
38
39             //retrieving the values
40             for (int c = 0; c < 5; c++)
41             {
42                 Console.Write(intArray.getItem(c) + " ");
43             }
44
```

*Fortsættes på næste slide med linje 45-66*

# GENERICS

```
45              Console.WriteLine();
46
47              //declaring a character array
48              MyGenericArray<char> charArray = new MyGenericArray<char>(5);
49
50              //setting values
51              for (int c = 0; c < 5; c++)
52              {
53                  charArray.setItem(c, (char)(c + 97));
54              }
55
56              //retrieving the values
57              for (int c = 0; c < 5; c++)
58              {
59                  Console.Write(charArray.getItem(c) + " ");
60              }
61              Console.WriteLine();
62
63              Console.ReadKey();
64          }
65      }
66  }
```

*Forsat fra linje 1-44 på sidste slide*

# GENERICS EGENSKABER

## Generics egenskaber

Generics kan forbedre ens programmer på følgende måder:

De hjælper med at maksimere genbrug af kode, type sikkerhed og ydeevne.

Du kan oprette generic collection klasser.

- .NET Framework klasse bibliotek indeholder en række nye generic collection classes i System.Collections.Generic namespace. Man kan bruge disse generic collection classes i stedet for collection classes i System.Collections namespace.

Man kan lave egne generic interfaces, classes, methods, events og delegates.

Man kan lave generic classes tvunget til at give adgang til metoder for bestemte datatyper.

Man kan få information om de typer, der anvendes i en generic datatype ved run-time ved hjælp af reflection.

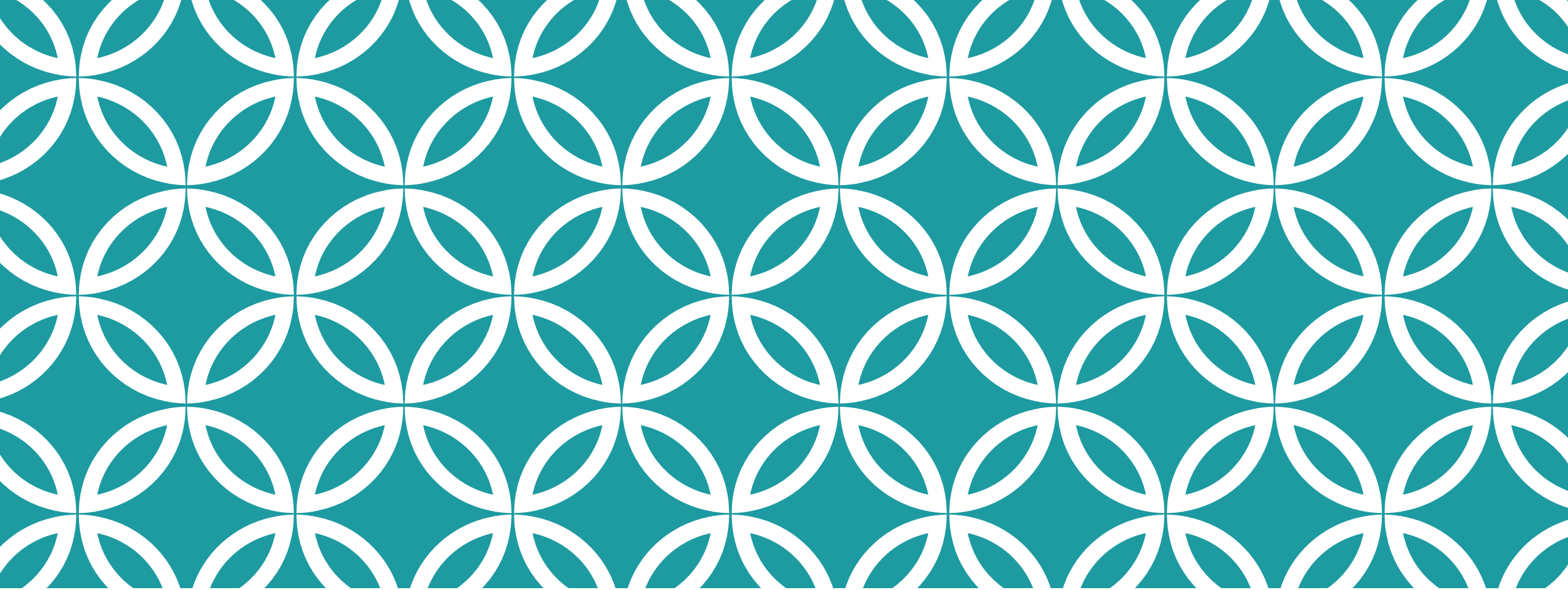
# GENERICS METODER

## Generic methods

Man kan declare en generic method med et type parameter.

```
using System;
using System.Collections.Generic;

namespace GenericMethodAppl
{
    0 references
    class Program
    {
        2 references
        static void Swap<T>(ref T lhs, ref T rhs)
        {
            T temp;
            temp = lhs;
            lhs = rhs;
            rhs = temp;
        }
        0 references
        static void Main(string[] args)
        {
            int a, b;
            char c, d;
            a = 10;
            b = 20;
            c = 'I';
            d = 'V';

            //display values before swap:
            Console.WriteLine("Int values before calling swap:");
            Console.WriteLine("a = {0}, b = {1}", a, b);
            Console.WriteLine("Char values before calling swap:");
            Console.WriteLine("c = {0}, d = {1}", c, d);

            //call swap
            Swap<int>(ref a, ref b);
            Swap<char>(ref c, ref d);

            //display values after swap:
            Console.WriteLine("Int values after calling swap:");
            Console.WriteLine("a = {0}, b = {1}", a, b);
            Console.WriteLine("Char values after calling swap:");
            Console.WriteLine("c = {0}, d = {1}", c, d);

            Console.ReadKey();
        }
    }
}
```


C#
#15
GENERICS



# PRAKSIS EKSEMPEL

Et program der holder styr på bøger

# PRAKSIS EKSEMPEL

Hent bogdb.cs filen fra denne lektions mappe på Fronter.

Dette eksempel benytter de emner vi har gennemgået hidtil til at lave et samlet biblioteks-administrerings program.

I skal ikke blive forvirret om at det omtaler en database, data gemmes reelt internt i programmet og forsvinder ved lukning, men det vi laver i arrays er en database-lignende struktur og brug, så derfor betegnelsen.

# LEKTIE

Kig på dette til næste gang

# LEKTIE

Se https://mva.microsoft.com/en-US/training-courses/programming-in-c-jump-start-14254 - 05

Læs:

- https://www.tutorialspoint.com/csharp/csharp_anonymous_methods.htm,
- https://msdn.microsoft.com/en-us/library/bb397687.aspx og
- https://msdn.microsoft.com/da-dk/library/bb882516.aspx

# KILDER

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

# KILDER

## Exception handling

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

https://youtu.be/gOtZSaLPu-E

https://youtu.be/El8rIaE3LI8

## Validation

https://mva.microsoft.com/en-US/training-courses/programming-in-c-jump-start-14254?l=KkOpp1SfB_8200115888

## Generics

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

https://youtu.be/ZrjCG0Fu5Ew