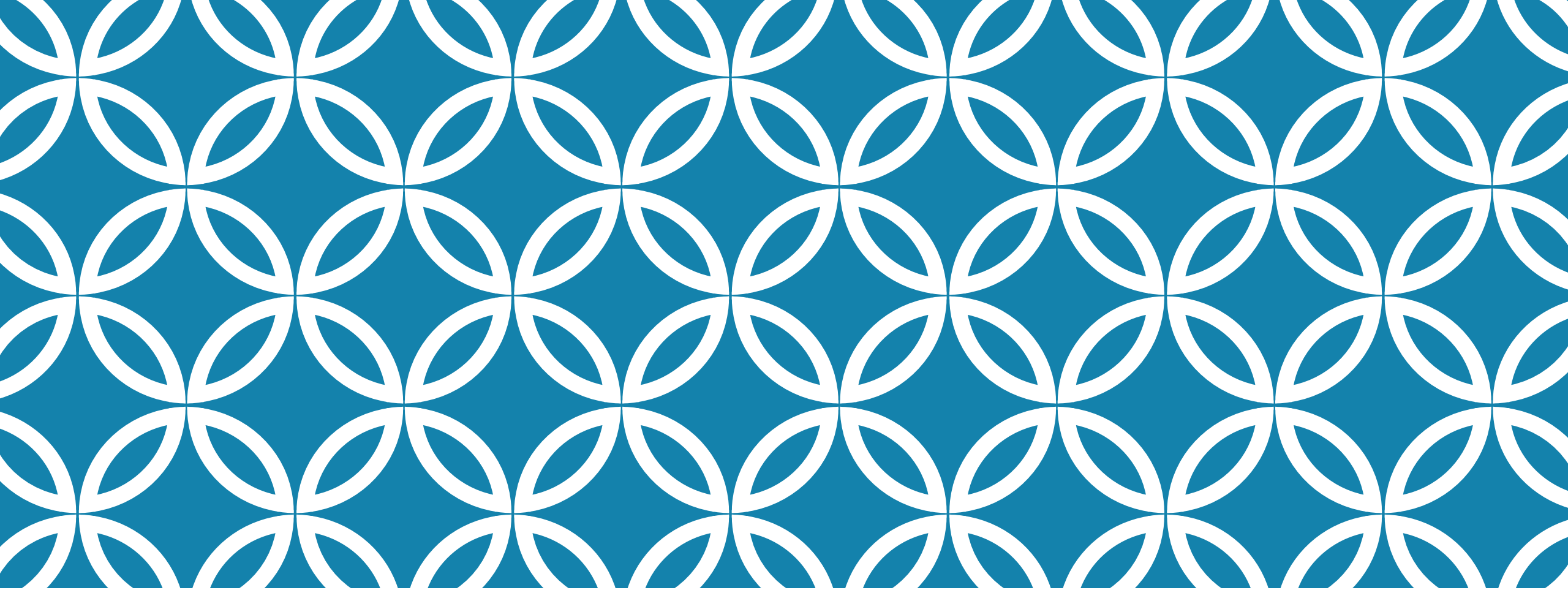
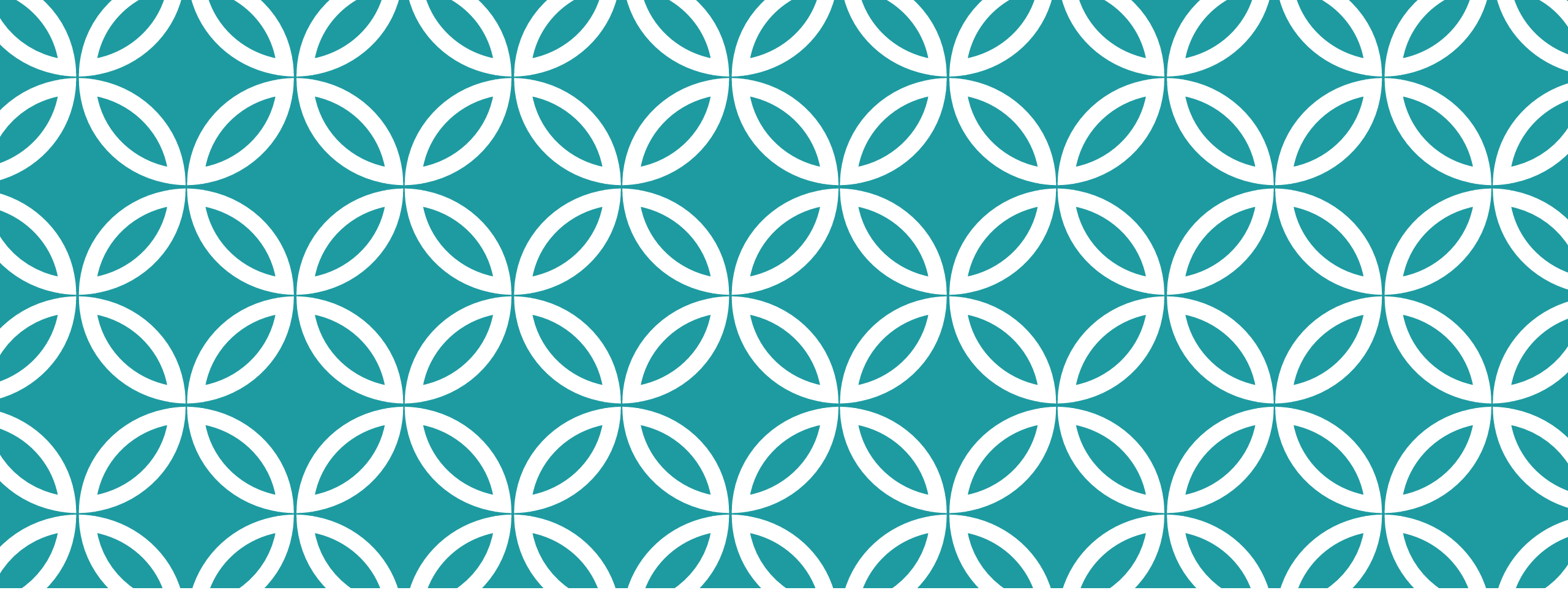


COLLECTIONS LINQ
CUSTOM COLLECTIONS

Grundlæggende programmering
Lektion 10



COLLECTIONS

Klasser til at gemme og hente data

COLLECTIONS

Collection klasser er specialiseret i data opbevaring og hentning.

Disse klasser giver support for stacks, queues, lists og hash tables.

De fleste collection klasser implementerer de samme interfaces.

Collection klasser tjener flere formal:

- At allokere hukommelse dynamisk til elementer
- At tilgå en liste af items på basis af et indeks osv.

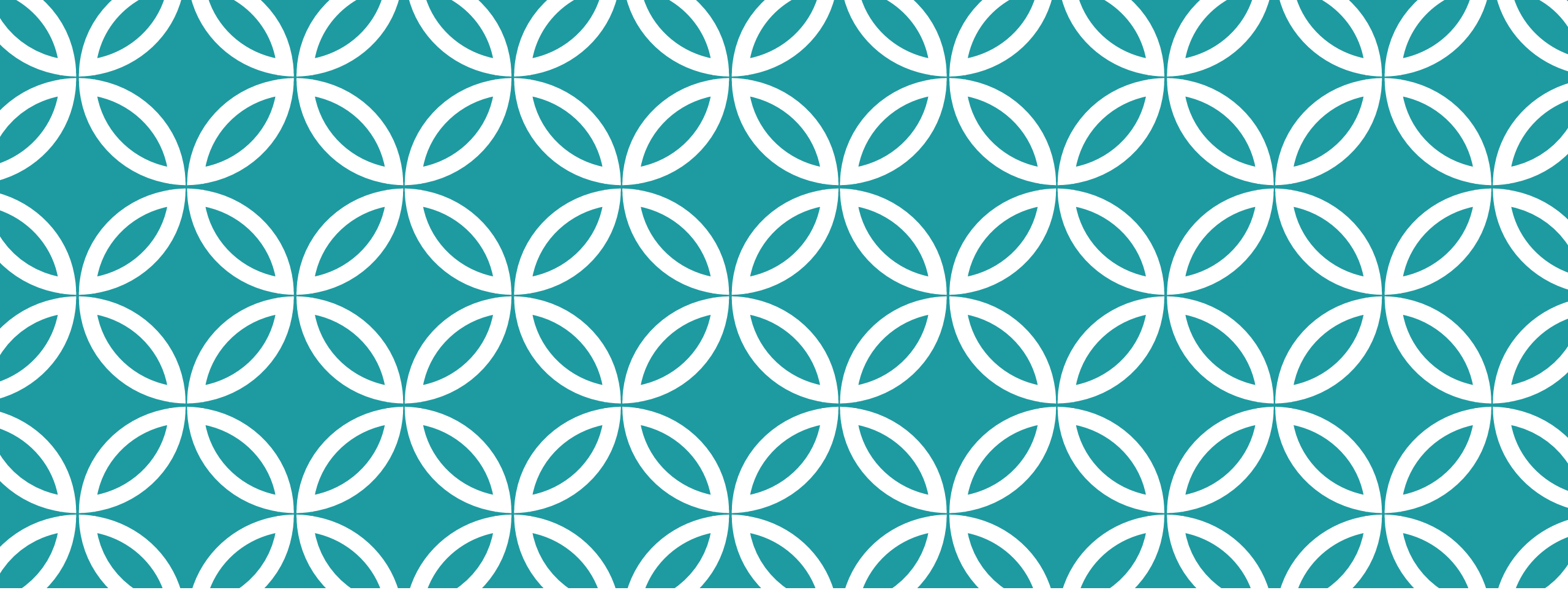
Disse klasse laver collections af objekter af Object klassen, der er basis klassen for alle data typer i C#.

Her en tabel over de mest brugte klasser i System.Collection namespace.

Class	Description and Useage
ArrayList	It represents ordered collection of an object that can be **indexed** individually. It is basically an alternative to an array. However, unlike array you can add and remove items from a list at a specified position using an **index** and the array resizes itself automatically. It also allows dynamic memory allocation, adding, searching and sorting items in the list.
Hashtable	It uses a **key** to access the elements in the collection. A hash table is used when you need to access elements by using key, and you can identify a useful key value. Each item in the hash table has a **key/value** pair. The key is used to access the items in the collection.
SortedList	It uses a **key** as well as an **index** to access the items in a list. A sorted list is a combination of an array and a hash table. It contains a list of items that can be accessed using a key or an index. If you access items using an index, it is an ArrayList, and if you access items using a key , it is a Hashtable. The collection of items is always sorted by the key value.
Stack	It represents a **last-in, first out** collection of object. It is used when you need a last-in, first-out access of items. When you add an item in the list, it is called **pushing** the item and when you remove it, it is called **popping** the item.
Queue	It represents a **first-in, first out** collection of object. It is used when you need a first-in, first-out access of items. When you add an item in the list, it is called **enqueue** and when you remove an item, it is called **deque**.
BitArray	It represents an array of the **binary representation** using the values 1 and 0. It is used when you need to store the bits but do not know the number of bits in advance. You can access items from the BitArray collection by using an **integer index**, which starts from zero.

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

10 references
public class Customer
{
    3 references
    public Customer(int id, string name)
    {
        ID = id;
        Name = name;
    }

    private int m_id;

    4 references
    public int ID
    {
        get { return m_id; }
        set { m_id = value; }
    }

    private string m_name;

    2 references
    public string Name
    {
        get { return m_name; }
        set { m_name = value; }
    }
}

0 references
class Program
{
    0 references
    static void Main(string[] args)
    {
        List<int> myInts = new List<int>();

        myInts.Add(1);
        myInts.Add(2);
        myInts.Add(3);
```

```
38
39          for (int i = 0; i < myInts.Count; i++)
40          {
41              Console.WriteLine("MyInts: {0}", myInts[i]);
42          }
43
44          Dictionary<int, Customer> customers = new Dictionary<int, Customer>();
45
46          Customer cust1 = new Customer(1, "Cust 1");
47          Customer cust2 = new Customer(2, "Cust 2");
48          Customer cust3 = new Customer(3, "Cust 3");
49
50          customers.Add(cust1.ID, cust1);
51          customers.Add(cust2.ID, cust2);
52          customers.Add(cust3.ID, cust3);
53
54          foreach (KeyValuePair<int, Customer> custKeyVal in customers)
55          {
56              Console.WriteLine(
57                  "Customer ID: {0}, Name: {1}",
58                  custKeyVal.Key,
59                  custKeyVal.Value.Name);
60          }
61
62          Console.ReadKey();
63      }
64  }
```



CUSTOM COLLECTIONS

Ens egne collections

CUSTOM COLLECTIONS

Man kan lave ens egen collection klasse ved at arve fra en af de mange.NET Framework collection klasser og føje kode til for at implementere ens egen bruger-definerede funktionalitet.

Ligegyldigt hvor begrænsede funktioner man vil give ens collection er der nogle funktioner man skal give den.

- En collection er nødt til at støtte behandlingen af alle sine poster med et For ... While loop.
- Det er meget usædvanligt at en collection ikke understøtter hentning af enkelte poster i collectionen ved en position (en indeksering).

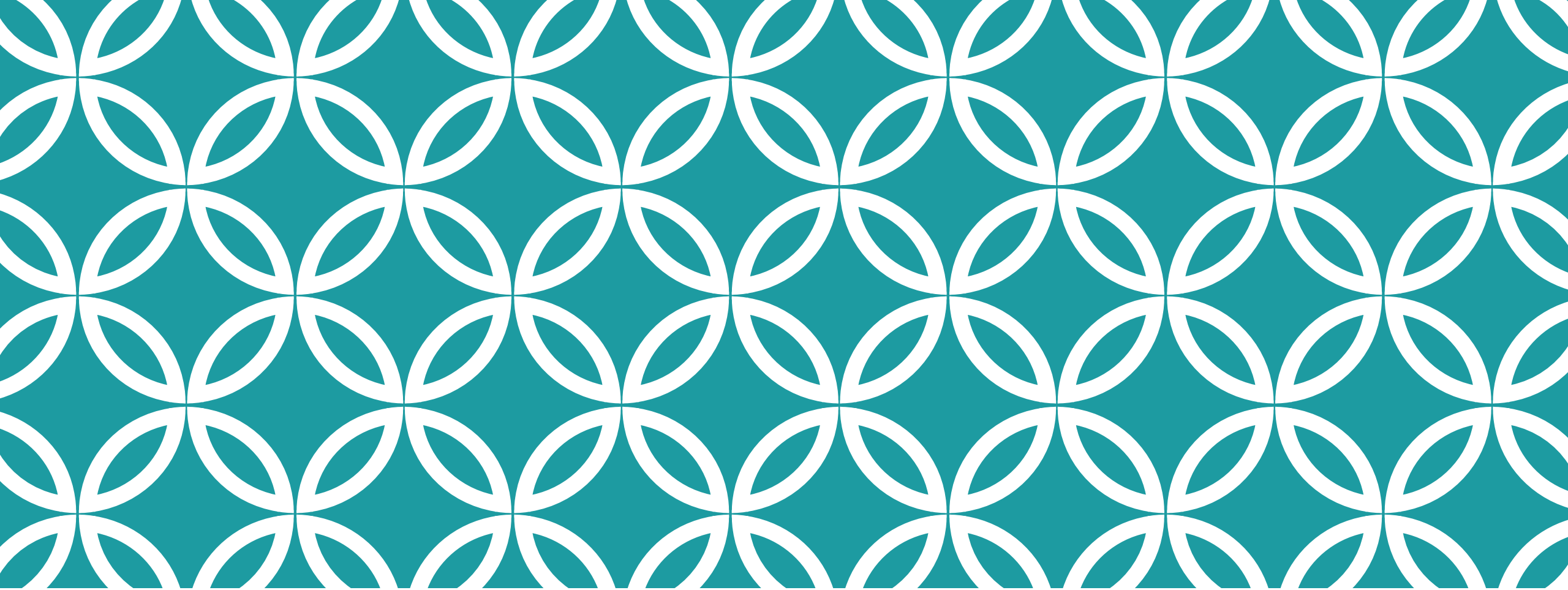
Nogle funktioner vil dog føre til andre.

- Hvis man f.eks. vil understøtte at tilføje elementer til ens collection vil man også ønske at understøtte initializers (at tilføjer flere elementer til samlingen ved hjælp af tuborg-klammer ({...}).

C# GENERICS

Part 1: Collections

Presented by Jeremy Clark
www.jeremybytes.com



LINQ

Adgang til databaser, XML og andre data kilder

LINQ

Akronymet LINQ star for Language INtegrated Query.

Gennem LINQ queries får man let data adgang til in-memory objekter, databaser, XML documenter og mere til.

LINQ blev indført i Visual Studio 2008 og er designet af Anders Hejlsberg.

LINQ tillader en at skrive queries selv uden kendskab til query sprog som SQL, XML osv.

LINQ queries kan skrives til forskellige datatyper.

```
1   using System;
2   using System.Linq;
3
    0 references
4   class Program
5   {
        0 references
6       static void Main()
7       {
8           string[] words = { "hello", "wonderful", "LINQ", "beautiful", "world" };
9           //Get only short words
10          var shortWords = from word in words
11                           where word.Length <= 5
12                           select word;
13
14          //Print each word out
15          foreach (var word in shortWords)
16          {
17              Console.WriteLine(word);
18          }
19          Console.ReadLine();
20      }
21  }
```

LINQ
SYNTAKS

Der er to syntakser I LINQ.

Lamda (Method) Syntax

```
var longWords = words.Where( w => w.length > 10);
```

Query (Comprehension) Syntax

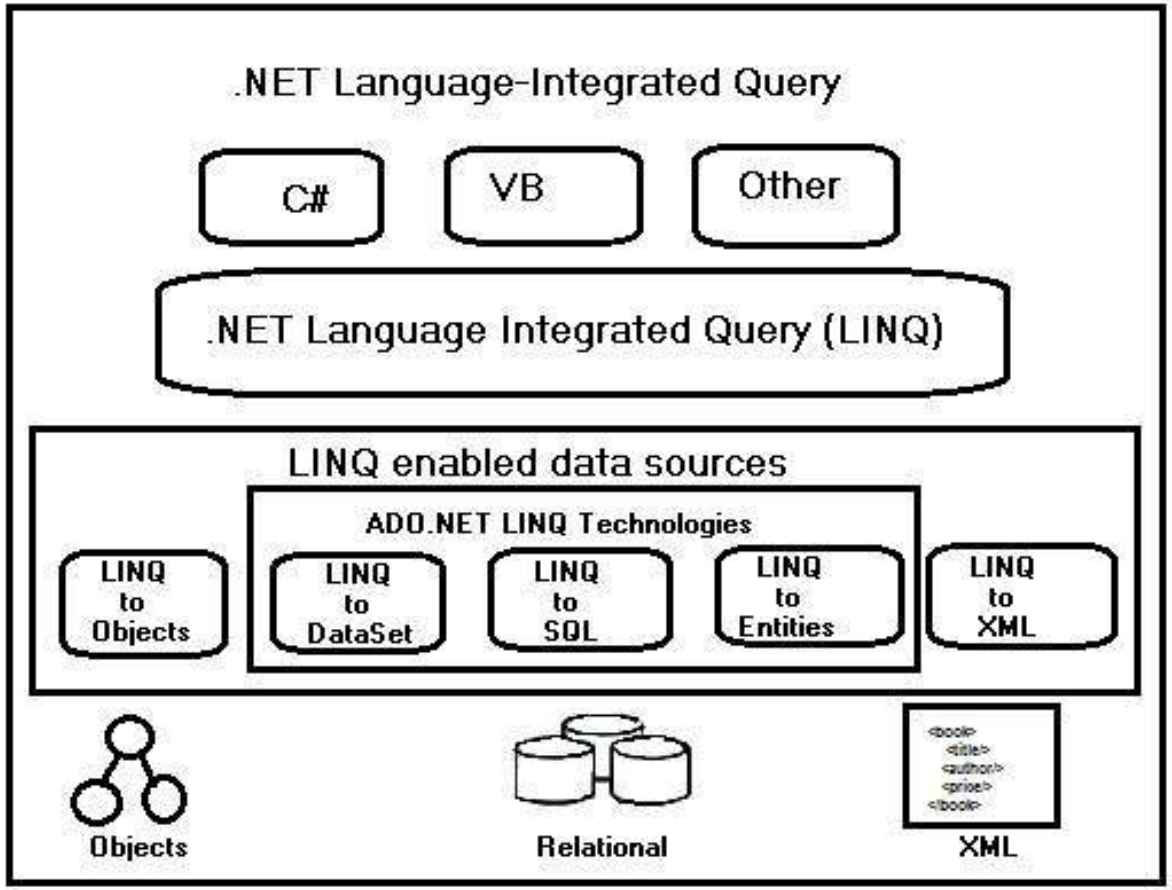
```
var longwords = from w in words where w.length > 10;
```

Query udtryk er ikke andet end en LINQ forespørgsel, udtrykt i en form, der svarer til SQL med query operatører som Select, Where og OrderBy.

Query udtryk starter som regel med nøgleordet ”From".

LINQ
I .NET

LINQ har en 3-laget arkitektur i hvilken det øverste lag består af sprog (language) extensions og det nederste lag består af data kilder, der typisk er objekter der implementerer IEnumerable<T> eller IQueryable<T> generiske interfaces.



LINQ
QUERY

```
1   using System;
2   using System.Collections.Generic;
3   using System.Linq;
4   using System.Text;
5
6   namespace Operators
7   {
        0 references
8       class LINQQueryExpressions
9       {
            0 references
10          static void Main()
11          {
12              // Specify the data source.
13              int[] scores = new int[] { 97, 92, 81, 60 };
14
15              // Define the query expression.
16              IEnumerable<int> scoreQuery = from score in scores
17                                            where score > 80
18                                            select score;
19
20              // Execute the query.
21              foreach (int i in scoreQuery)
22              {
23                  Console.Write(i + " ");
24              }
25              Console.ReadKey();
26          }
27      }
28  }
```

LINQ
VS STORED PROCEDURES

Der er en række forskelle på LINQ og Stored procedures.

Stored procedures er meget hurtigere end en LINQ forespørgsel da de følger en forventet udførelses plan.

Det er nemmere at undgå run-time fejl under udførelse af en LINQ query i forhold til en stored procedure.

- LINQ har Visual Studio Intellisense support samt full-type checking ved kompilerings-tid.

LINQ tillader debugging igennem .NET debugger hvilket ikke er tilfældet for stored procedures.

LINQ tilbyder understøttelse af flere databaser i modsætning til stored procedures, hvor det er vigtigt at omskrive koden til forskellige typer af databaser.

Implementering af LINQ baserede løsninger er nemt og enkelt i forhold til indsættelsen af en række af stored procedures.

LINQ
UDEN OG MED

Førhen uden LINQ

```
SqlConnection sqlConnection = new SqlConnection(connectString);
SqlConnection.Open();
System.Data.SqlClient.SqlCommand sqlCommand = new SqlCommand();
sqlCommand.Connection = sqlConnection;
sqlCommand.CommandText = "Select * from Customer";
return sqlCommand.ExecuteReader (CommandBehavior.CloseConnection)
```

Med LINQ

```
Northwind db = new Northwind(@"C:\Data\Northwnd.mdf");
var query = from c in db.Customers
            select c;
```

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 - 07

Læs:

- https://www.tutorialspoint.com/csharp/csharp_events.htm
- https://www.tutorialspoint.com/csharp/csharp_file_io.htm

Opgave

- Lav et program hvor brugeren angiver en tekst-streng der så gemmes som en fil

KILDER

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

KILDER

Collections

- https://www.tutorialspoint.com/csharp/csharp_collections.htm
- http://csharp-station.com/Tutorial/CSharp/Lesson20
- https://youtu.be/J9Cwi45UtZU

Custom collections

- https://support.microsoft.com/da-dk/kb/307484
- https://msdn.microsoft.com/en-us/library/xth2y6ft(v=vs.71).aspx
- http://www.c-sharpcorner.com/uploadfile/skumaar_mca/custom-collection-class-in-c-sharp/

KILDER

Custom collections

- https://www.codeproject.com/articles/265692/having-fun-with-custom-collections
- https://visualstudiomagazine.com/articles/2015/02/01/creating-a-simple-collection-class.aspx

LINQ

- http://www.tutorialspoint.com/linq/
- https://kevinlawry.wordpress.com/2012/08/07/why-i-avoid-stored-procedures-and-you-should-too/
- https://youtu.be/1UqjUgcdq6g