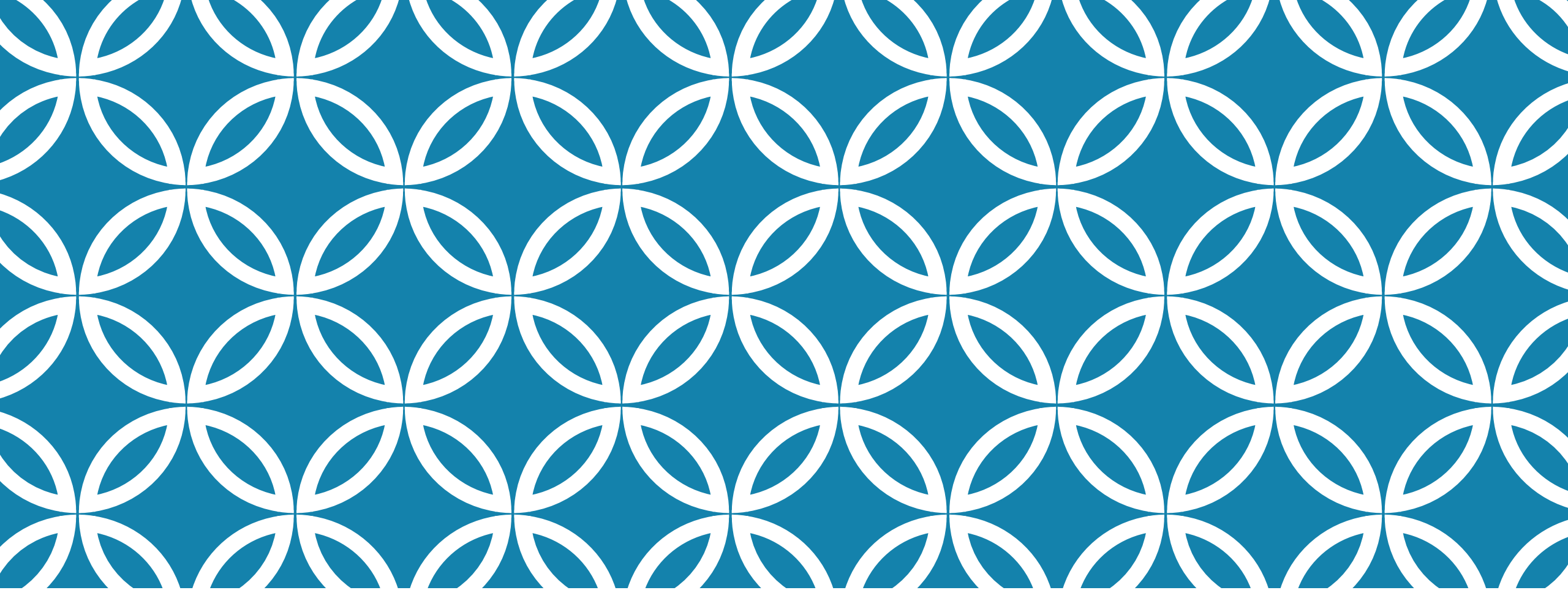


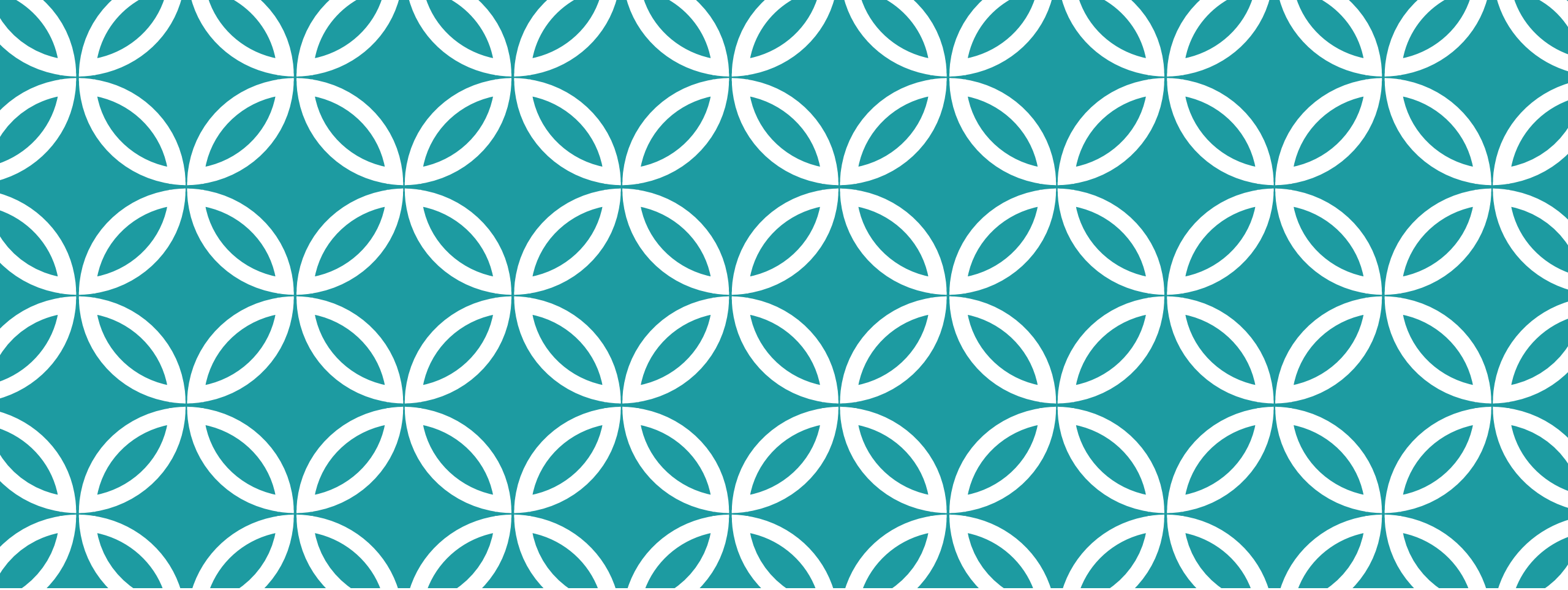
REKURSION OG JAGGED ARRAYS GENBESØGT

INTERFACES

VALUE TYPES

PRAKSIS EKSEMPEL

Grundlæggende programmering
Lektion 6



REKURSION OG JAGGED ARRAYS GENBESØGT

Et (forhåbentligt) bedre eksempel på kode der kalder sig selv

REKURSION GENBESØGT

En metode kan kalde en anden metode, men den kan også kalde sig selv. Når en metode kalder sig selv kaldes det en rekursiv metode.

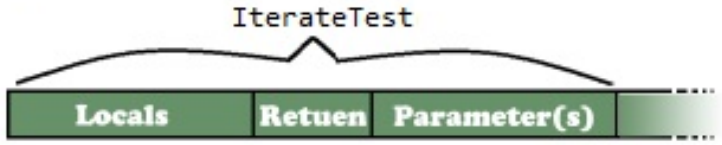
En rekursiv metode har to specifikationer:

1. En rekursiv metode kalder sig selv så mange gange indtil den er opfyldt.
2. En rekursiv metode har parameter(-re), og kalder sig selv med nye parameterværdier.

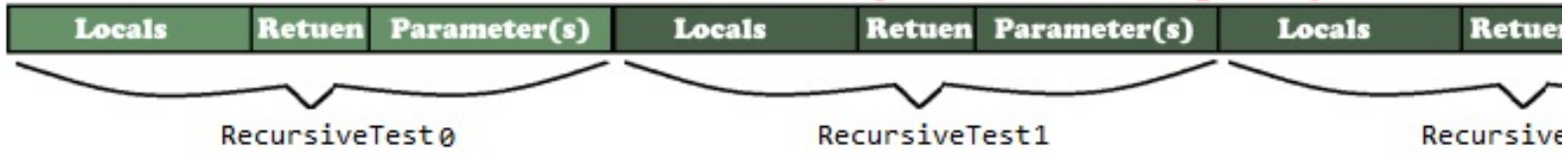
En rekursiv løsning er en kraftfuld og enkel fremgangsmåde til komplicerede løsninger, men det kan **forværre** ydeevne på grund af anvendelse af kaldstakken igen og igen.

REKURSION GENBESØGT

```
public int IterateTest(int parameter)
{
    int sum = 0;
    for (int i = 0; i <= parameter; i++)
        sum += i;
    return sum;
}
```



Recursive calls itself and it likes you are calling many methods




```
public int RecursiveTest(int parameter)
{
    return (parameter == 0) ? 0 : RecursiveTest(parameter - 1) + parameter;
}
```

REKURSION GENBESØGT

Uden rekursion

```
1   using System;
2
3   public long Factorial(int n)
4   {
5       if (n == 0)
6           return 1;
7       long value = 1;
8       for (int i = n; i > 0; i--)
9       {
10          value *= i;
11      }
12      return value;
13  }
```

Med rekursion

```
1   using System;
2
3   public long Factorial(int n)
4   {
5       if (n == 0)
6           return 1;
7       return n * Factorial(n - 1);
8   }
```

Se filerne i recursion_code.zip for et fuldt projekt der benytter rekursion (og recursion_demo.zip for en kompileret udgave af projektet).

JAGGED ARRAYS GENBESØGT

Dataene kommer i forskellige former.

Sommetider er formen er ujævn.

Med jagged arrays kan vi effektivt gemme mange rækker af varierende længder. Enhver type data, reference eller værdi kan anvendes.

Jagged arrays har forskellige egenskaber. Indeksering af jagged arrays er hurtig. Tildeling af dem er lidt langsom.

2D array code benchmarked: C#

```
// 2D array of 100 x 100 elements.
for (int a = 0; a < 100; a++)
{
    for (int x = 0; x < 100; x++)
    {
        int c = a1[a, x];
    }
}
```

Jagged array code benchmarked: C#

```
// Jagged array of 100 x 100 elements.
for (int a = 0; a < 100; a++)
{
    for (int x = 0; x < 100; x++)
    {
        int c = a2[a][x];
    }
}
```

Results

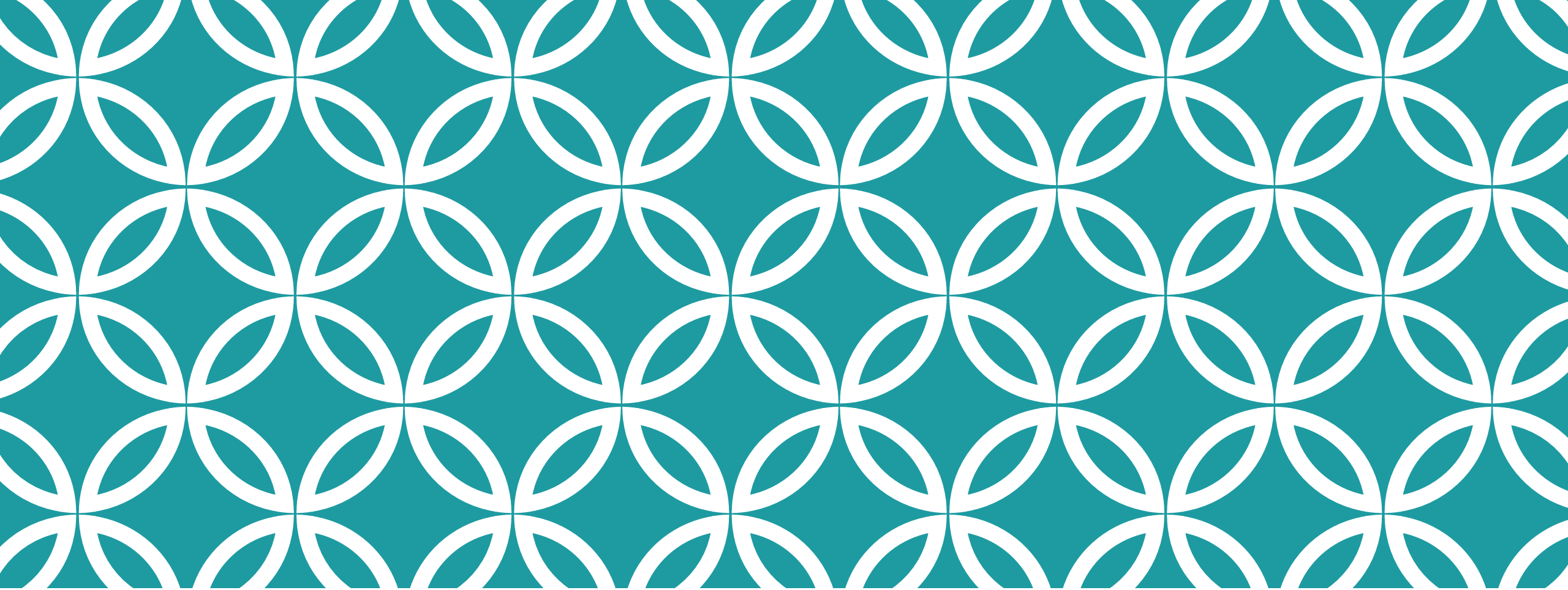
```
2D array looping:     4571 ms
Jagged array looping: 2864 ms [faster]
```

JAGGED ARRAYS GENBESØGT

```
0 references
public static void Main(string[] args)
{
    string[][] str = new string[5][];
    str[0] = new string[5];
    str[1] = new string[10];
    str[2] = new string[20];
    str[3] = new string[50];
    str[4] = new string[10];

    str[0][0] = "Pune";
    str[1][0] = "Kolkata";
    str[2][0] = "Bangalore";
    str[3][0] = "The pink city named Jaipur";
    str[4][0] = "Hyderabad";

    for (int i = 0; i < 5; i++)
        Console.WriteLine(str[i][0]);
    Console.Read();
}
```

In a jagged array, the number of rows is fixed, but the number of columns may vary.

Storing data of varying sizes in the jagged array of strings named str.



INTERFACES

Hvordan noget udføres

INTERFACES

Hvor klasser definerer *hvordan* noget skal udføres sørger interfaces for *hvad* der skal udføres.

Interfaces definerer properties, methods og events, der er medlemmer af interfacet.

Interfaces indeholder kun deklarationen af medlemmerne. Det er den afledte klasses ansvar at definere medlemmerne.

Interfacet hjælper ofte med at give en standard struktur, som de afledte klasser vil følge.

Abstrakte klasser har til en vis grad samme formål, men de bliver primært brugt når der kun skal erklæres få metoder af base-klassen og de underliggende klasser implementerer funktionerne.

INTERFACES

Interfaces erklæres med interface nøgleordet.

- Det svarer til en klasse erklæring.

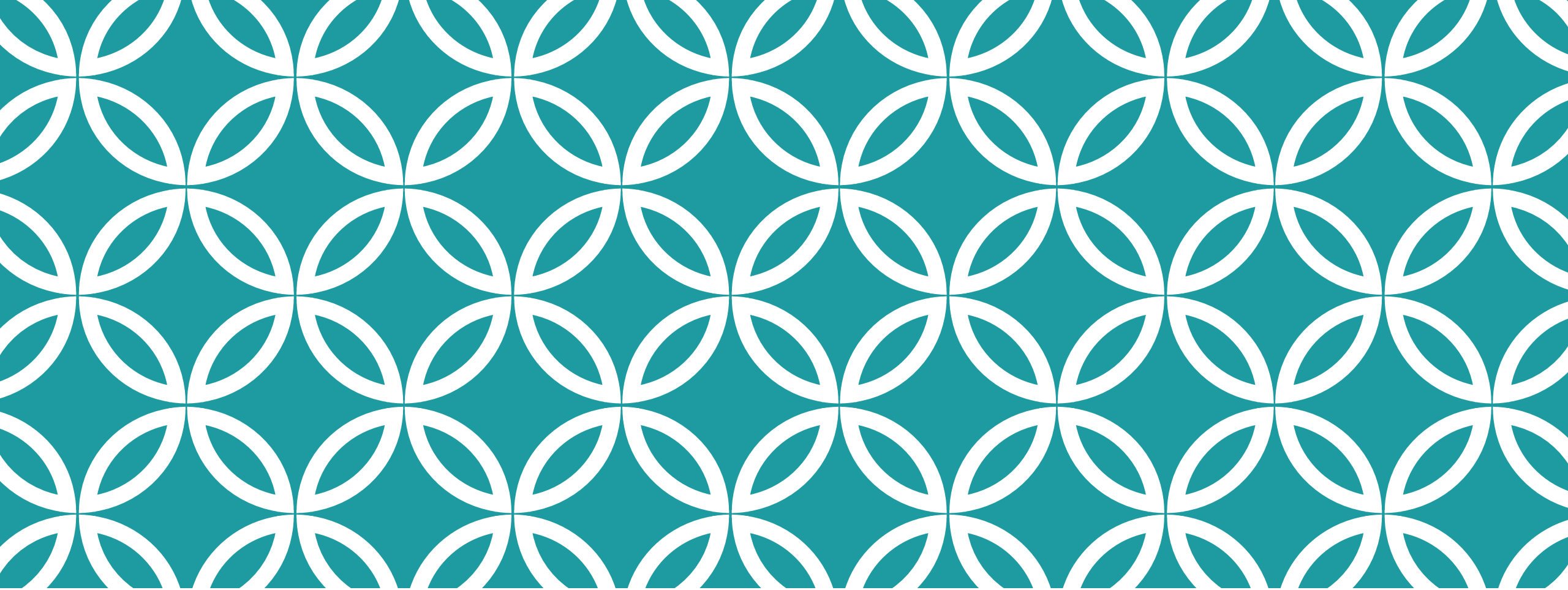
Interface-udsagn er offentlige som standard.

Følgende er et eksempel på et interface erklæring:

```
public interface ITransactions
{
    // interface members
    void showTransaction();
    double getAmount();
}
```

```
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System;

namespace InterfaceApplication
{
    1 reference
    public interface ITransactions
    {
        // interface members
        3 references
        void showTransaction();
        2 references
        double getAmount();
    }

    6 references
    public class Transaction : ITransactions
    {
        private string tCode;
        private string date;
        private double amount;
        0 references
        public Transaction()
        {
            tCode = " ";
            date = " ";
            amount = 0.0;
        }

        2 references
        public Transaction(string c, string d, double a)
        {
            tCode = c;
            date = d;
            amount = a;
        }

        2 references
        public double getAmount()
        {
            return amount;
        }

        3 references
        public void showTransaction()
        {
            Console.WriteLine("Transaction: {0}", tCode);
            Console.WriteLine("Date: {0}", date);
            Console.WriteLine("Amount: {0}", getAmount());
        }
    }
    0 references
    class Tester
    {
        0 references
        static void Main(string[] args)
        {
            Transaction t1 = new Transaction("001", "8/10/2012", 78900.00);
            Transaction t2 = new Transaction("002", "9/10/2012", 451900.00);
            t1.showTransaction();
            t2.showTransaction();
            Console.ReadKey();
        }
    }
}
```


C#
#14
INTERFACES



VALUE TYPES

Structs, boxing og enums

VALUE TYPES

En value type indeholder direkte sin værdi.

- Navnet forbindes direkte med pladsen i hukommelsen, hvor data er gemt.
- Hvis en anden variabel får værdien fra den første variabel laves der en kopi af værdien på den ny variabels plads, da to variabler aldrig kan referere til den samme plads.
  - Undtagelsen er hvis en eller begge har out eller ref parametrene, men således er de også defineret som alias for en anden variabel.

VALUE TYPES STRUCT

En structure er en værdi data type.

Den lader en enkelt variable indeholde relateret data i forskellige data typer.

Man benytter *struct* keyword til at lave en structure.

- Et struct statement definerer en ny data type, der har mere end et medlem.

```
struct Books
{
    public string title;
    public string author;
    public string subject;
    public int book_id;
};
```

VALUE TYPES

```
using System;
2 references
struct Books
{
    public string title;
    public string author;
    public string subject;
    public int book_id;
};

0 references
public class testStructure
{
    0 references
    public static void Main(string[] args)
    {

        Books Book1;   /* Declare Book1 of type Book */
        Books Book2;   /* Declare Book2 of type Book */

        /* book 1 specification */
        Book1.title = "C Programming";
        Book1.author = "Nuha Ali";
        Book1.subject = "C Programming Tutorial";
        Book1.book_id = 6495407;

        /* book 2 specification */
        Book2.title = "Telecom Billing";
        Book2.author = "Zara Ali";
        Book2.subject = "Telecom Billing Tutorial";
        Book2.book_id = 6495700;

        /* print Book1 info */
        Console.WriteLine("Book 1 title : {0}", Book1.title);
        Console.WriteLine("Book 1 author : {0}", Book1.author);
        Console.WriteLine("Book 1 subject : {0}", Book1.subject);
        Console.WriteLine("Book 1 book_id :{0}", Book1.book_id);

        /* print Book2 info */
        Console.WriteLine("Book 2 title : {0}", Book2.title);
        Console.WriteLine("Book 2 author : {0}", Book2.author);
        Console.WriteLine("Book 2 subject : {0}", Book2.subject);
        Console.WriteLine("Book 2 book_id : {0}", Book2.book_id);

        Console.ReadKey();

    }
}
```

VALUE TYPES STRUCT

Egenskaber for structures

Structures kan have methods, fields, indexers, properties, operator methods og events.

Structures kan have definerede constructors, men ikke destructors.

Modsat classes kan structures íkke arve fra andre structures eller classes.

Structures kan ikke bruges som en base for andre structures eller classes.

En structure kan implementere et eller flere interfaces.

Structure members kan ikke specifceres som abstract, virtual eller protected.

Når man opretter et struct objekt ved hjælp af New operator bliver det oprettet og den passende constructor kaldes. I modsætning til klasser, kan structs instantieres uden at bruge New operator.

Hvis der ikke anvendes New operator forbliver felterne ikke-tildelt og objektet kan ikke anvendes før alle felter er initialiseret.

VALUE TYPES STRUCT

Classes vs. structures

Classes og structures har følgende grundlæggende forskelle:

Classes er referencetyper og structs er værdi typer.

Structures understøtter ikke nedarvning.

Structures kan ikke have default constructor.

- I C# 6.0 har alle værdi typer en default constructor .


C#
PROGRAMMING
STRUCTS
RC

VALUE TYPES BOXING

Boxing og unboxing lader værdi typer blive behandlet som objekter.

Boxing af en værdi typen pakker den ind i en instans af Object henvisningens typen.

Unboxing udtrækker værditypen fra objektet.

Her tages integer variablen i som bliver boxed og tildelt objektet o

```
int i = 123;
// The following line boxes i.
object o = i;
```

Objektet o kan så unboxes og gives til integer variablen i

```
o = 123;
i = (int)o;  // unboxing
```

VALUE TYPES ENUMS

En enumeration er et sæt navngivne integer konstanter.

En enumerated type declares med enum keyword.

C# enumerations er en værdi data type. Enumeration indeholder altså dens egne værdier og kan ikke arve eller videresende arv.

Enum declares således

```
enum <enum_name>
{
   enumeration list
};
```

- *enum_name* angiver enumeration typens navn.
- *enumeration list* er en komma-separeret liste af identifiers.

Hver af symbolerne i enumeration listen står for en heltalsværdi, en større end symbolet der er angivet foran. Som standard er værdien af det første enumeration symbol 0.

```
enum Days { Sun, Mon, tue, Wed, thu, Fri, Sat };
```

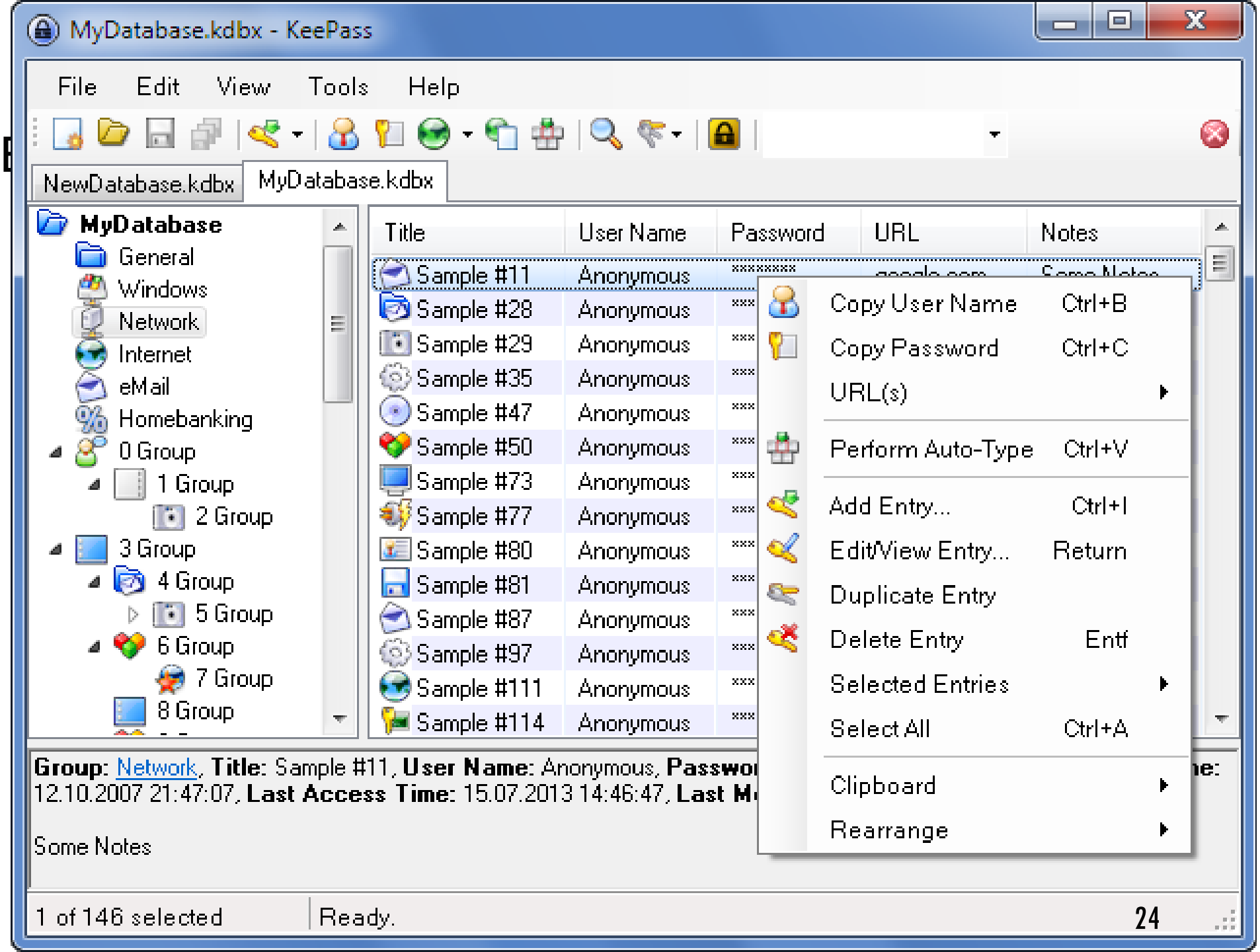
VALUE TYPES ENUMS

```
1   using System;
2   namespace EnumApplication
3   {
        0 references
4       class EnumProgram
5       {
            2 references
6           enum Days { Sun, Mon, tue, Wed, thu, Fri, Sat };
7
            0 references
8           static void Main(string[] args)
9           {
10              int WeekdayStart = (int)Days.Mon;
11              int WeekdayEnd = (int)Days.Fri;
12              Console.WriteLine("Monday: {0}", WeekdayStart);
13              Console.WriteLine("Friday: {0}", WeekdayEnd);
14              Console.ReadKey();
15          }
16      }
17  }
```

PRAKSIS EKSEMPEL

KeePass, et program der husker kodeord

PRAKSIS EKSE
KEEPASS



PRAKSIS EKSEMPEL KEEPASS

```
87      public enum AppMessage
88      {
89          Null = 0,
90          RestoreWindow = 1,
91          Exit = 2,
92          IpcByFile = 3,
93          AutoType = 4,
94          Lock = 5,
95          Unlock = 6,
96          AutoTypeSelected = 7
97      }
98
99      public static CommandLineArgs CommandLineArgs
100     {
101         get
102         {
103             if(m_cmdLineArgs == null)
104             {
105                 Debug.Assert(false);
106                 m_cmdLineArgs = new CommandLineArgs(null);
107             }
108
109             return m_cmdLineArgs;
110         }
111     }
```

Kun en lille del af den samlede kode

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 - 04

Læs:

- https://www.tutorialspoint.com/csharp/csharp_interfaces.htm

KILDER

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

KILDER

Rekursion og jagged arrays genbesøgt

https://www.codeproject.com/Articles/142292/Recursive-methods-in-Csharp

https://www.dotnetperls.com/jagged-array

Interfaces

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

https://youtu.be/lQpss9YAc4g

Value types

https://msdn.microsoft.com/en-us/library/yz2be5wk(v=vs.80).aspx

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

https://msdn.microsoft.com/da-dk/library/sbbt4032.aspx

https://www.dotnetperls.com/enum

http://csharp.net-informations.com/statements/enum.htm

KILDER

Praksis eksempel

http://keepass.info/

https://sourceforge.net/projects/keepass/files/KeePass%202.x/