DO YOU EVEN C#?

itverket

C#

- Utviklet av Microsoft
- «Cool»
- Mange fellestrekk med Java
  - Syntaks
  - Trenger en «runtime» for å kjøre
- .NET Core
  - Open-source
  - Cross-platform

NAVNEKONVENSJON

- «PascalCase»

```
public string ToString()
{
    return "C#";
}
```

PROPERTIES

- Getters / Setters

```
public class Person
{
    private string _firstName;

    public string FirstName
    {
        get { return _firstName; }
        set { _firstName = value; }
    }
}
var person = new Person();
person.FirstName = "Kato";
```

PROPERTIES

- Auto-properties

```
public class Person
{
    public string FirstName { get; set; }
}

var person = new Person();
person.FirstName = "Kato";
```

PROPERTIES

- Access Modifiers

```
public class Person
{
    public string FirstName { get; private set; }
}

var person = new Person();
person.FirstName = "Kato"; // Kompileringsfeil
```

ARV

- Kan arve bare fra en klasse

```
public class Person
{
    public Person(string firstName, string lastName)
    {
        FirstName = firstName;
        LastName = lastName;
    }

    public string FirstName { get; private set; }
    public string LastName { get; private set; }
}
```

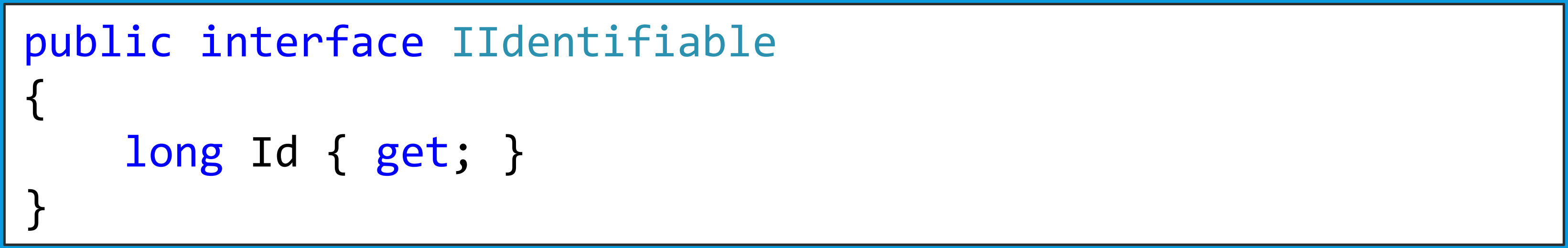
ARV

- Kan arve bare fra en klasse

```
public class Student : Person
{
    public Student(string firstName, string lastName)
        : base(firstName, lastName)
    {
    }
}
```

ARV

- Kan arve fra mange interface



```
public interface IIdentifiable
{
    long Id { get; }
}
```

ARV

- Kan arve fra mange interface

```
public class Student : Person, IIdentifiable
{
    public Student(string firstName, string lastName)
        : base(firstName, lastName)
    {
    }

    public long Id { get; set; }
}
```

VIRTUAL / OVERRIDE

- Methods / properties er ikke virtuelle som standard

```
// public class Person

public string GetFullName()
{
    return $"{LastName}, {FirstName}";
}
```

VIRTUAL / OVERRIDE

- Methods / properties er ikke virtuelle som standard

```
// public class Person

public virtual string GetFullName()
{
    return $"{LastName}, {FirstName}";
}
```

VIRTUAL / OVERRIDE

- Methods / properties er ikke virtuelle som standard

```
// public class Student : Person

public override string GetFullName()
{
    return $"{base.GetFullName()} ({Id})";
}
```

GENERICS

- Generiske klasser

```
public class List<T>
{
    private T[] _items;
}

public class KeyValuePair<TKey, TValue>
{
    private TKey _key;
    private TValue _value;
}
```

GENERICS

- Generiske metoder

```
public void WriteToConsole<T>(string message, T type)
{
    Console.WriteLine($"{message}: {type}");
}
```

GENERICS

- Constraints

```
public T CreateInstanceOf<T>() where T : class, new()
{
    return new T();
}

var person = CreateInstanceOf<Person>();
```

EXTENSION METHODS

- Statiske metoder som brukes som om de var en del av et objekt

```
public static class PersonExtensions
{
    public static string GetFullName(this Person person)
    {
        return $"{person.LastName}, {person.FirstName}";
    }
}

var person = new Person("Chuck", "Norris");
var fullName = person.GetFullName();
```

LAMBDA EXPRESSIONS

- Anonyme metoder
- Format: parameter(s) **=>** expression or statement
- Typer blir bestemt under kompilering

- x => x * x

LAMBDA EXPRESSIONS

- Eksempler

```
Func<int, int> squareNumber = x => x * x;

squareNumber(2);



// .. er det samme som
public int SquareNumber(int x)
{
    return x * x;
}
```

LINQ

- Language Integrated Query
- Rammeverk for
  - Spørringer mot objekter i minne (IEnumerable<T>)
  - Lesing / oppretting av XML (XElement)
  - Spørringer mot database (IQueryable<T>)
- To måter å bruke det på
  - Query expressions
  - Extension methods

LINQ

- Query Expressions

```
var childrenNames =
    from child in parent.Children
    select child.FirstName;
```

LINQ

- Query Expressions

```
var greatGrandchildrenNames =
    from child in greatGrandparent.Children
    from grandchild in child.Children
    from greatGrandchild in grandchild.Children
    select greatGrandchild.FirstName;
```

LINQ

- Med foreach-løkker

```
var greatGrandchildrenNames = new List<string>();

foreach (var child in greatGrandparent.Children)
{
    foreach (var grandchild in child.Children)
    {
        foreach (var greatGrandchild in grandchild.Children)
        {
            // add to list ...
        }
    }
}
```

LINQ

- Extension Methods

```
var childrenNames =
    parent.Children.Select(child => child.FirstName);
```

LINQ

- Extension Methods

```
parent.Children.First();

parent.Children.FirstOrDefault();

parent.Children.Where(child => child.FirstName == "Anders");

parent.Children.Select(child => child.GetFullName());

parent.Children.Count(child => child.FirstName == "Anders");
```

LINQ

- Extension Methods

```
IEnumerable<string> filteredChildren =
    parent.Children
        .Where(child => child.FirstName == "Anders")
        .Where(child => child.LastName == "Norris")
        .Select(child => child.GetFullName());

foreach (var childName in filteredChildren)
{
    Console.WriteLine(childName);
}
```

ITERATOR METHODS

- Lager en iterator som senere kan loopes gjennom

```
public IEnumerable<long> FibonacciNumbers()
{
    long current = 0;
    long next = 1;
    while (true)
    {
        yield return current;
        var temp = next;
        next = current + next;
        current = temp;
    }
}
```

ITERATOR METHODS

- Lager en iterator som senere kan loopes gjennom

```
IEnumerable<long> fibonacciNumbers = FibonacciNumbers();

foreach (var number in fibonacciNumbers)
{
    Console.WriteLine(number);
}

var tenFirstFibonacciNumbers =
    FibonacciNumbers().Take(10);
```

ASYNC / AWAIT

- For asynkron programmering
- Metoden markeres med nøkkelordet: **async**
- Metoden returnerer en Task evt. Task<string>
- **await** brukes dersom man har kode som skal kjøres etter operasjonen er ferdig

ASYNC / AWAIT

- Eksempel

```
var downloadTask = DownloadAsync("http://vg.no");
// Kjør kode som ikke er avhengig av downloadTask
var html = downloadTask.Result;

public async Task<string> DownloadAsync(string url)
{
    using (var httpClient = new HttpClient())
    {
        return await httpClient.GetStringAsync(url);
    }
}
```