C# Reflection

Where to use Reflection

- Normally compiler decision of types (**strongly typed language**)
- Sometimes we need the type information at runtime => reflection
- An examples Generic framework to do something
  - Json convert
  - Entity framework

What is Reflection

<u>Metatype</u>

- Type t = obj.GetType();

<u>Kind of information</u>

- t.IsXXX (class, interface, abstract)
- t. GetProperties()   => PropertyInfo (name, type etc)
- t. GetMethods() => MethodInfo
  (name, parameters , return types etc.) => invoke methods
- t.BaseType

Call methods in Reflection

Example:

```
Type t = o.GetType();

// find
MethodInfo setIdMethod =
                    t.GetMethods().First(m => m.Name == "set_Id");

// make call
setIdMethod.Invoke(o, new object[] { 12 });

// parameters is an array of values (though here only one)
```

Extension Methods
Creating methods outside the class

Example:

Normal class:

```
public class ExtensionToType{ … }
```

Extension:

```
public class SomeExtension              // that's how Linq is implemented
{

public static string XXMethodName(this ExtensionToType t)
{ … }


}
```

Anonymous classes

General description

var newObj = new { … }

Example:

var newClass =

new {Name=“Peter”, Address = “Roskilde”};

⇒ newClass is

an object with Name + Address as ***get-properties***

Small Demo

Opgave MyJsonConverter .