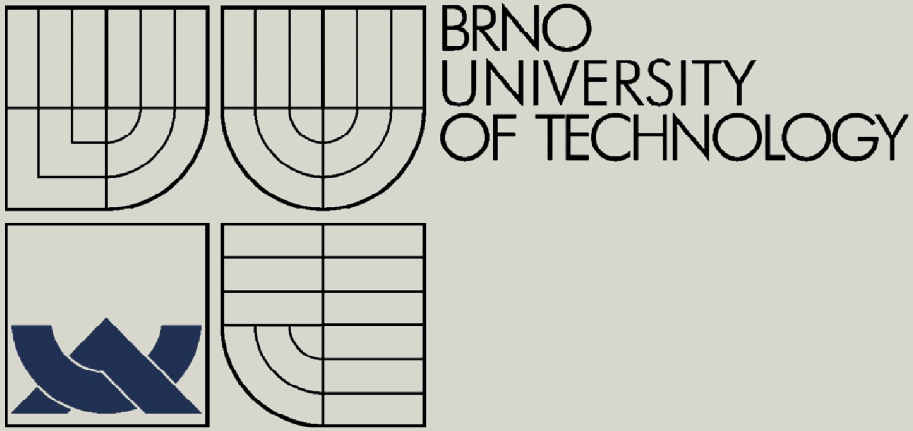


SIEMENS

05 – Bázové třídy platformy .NET

SIEMENS

Obsah přednášky

- BCL
- Správa paměti v .NETu
- IDisposable
- Kolekce
- Streamy

SIEMENS

BCL – Base Class Libary

- základní sada knihoven frameworku
- jádro je v mscorlib.dll

Obsahuje

- základní datové typy
- datové struktury
- vykonává I/O
- Informace o načtených typech
- Security
- Text
- Kolekce
- Threading
- Výčet není kompletní!

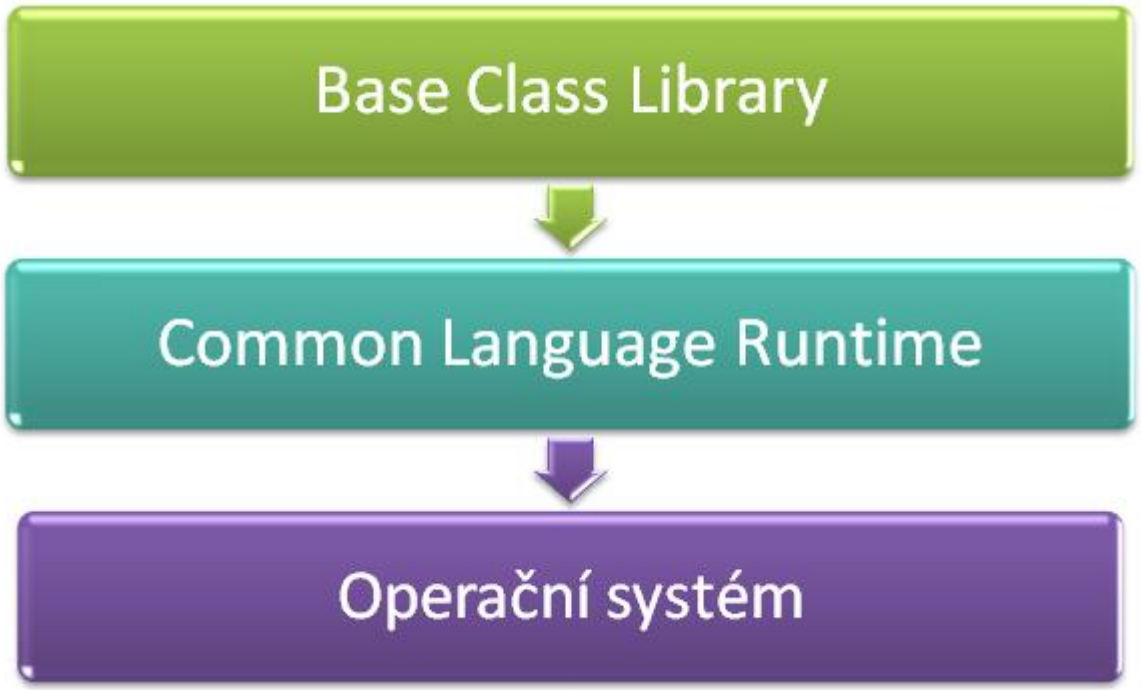
Jak je Base Class Library organizována

- Assemblies
- Organizována do jednotlivých namespaces

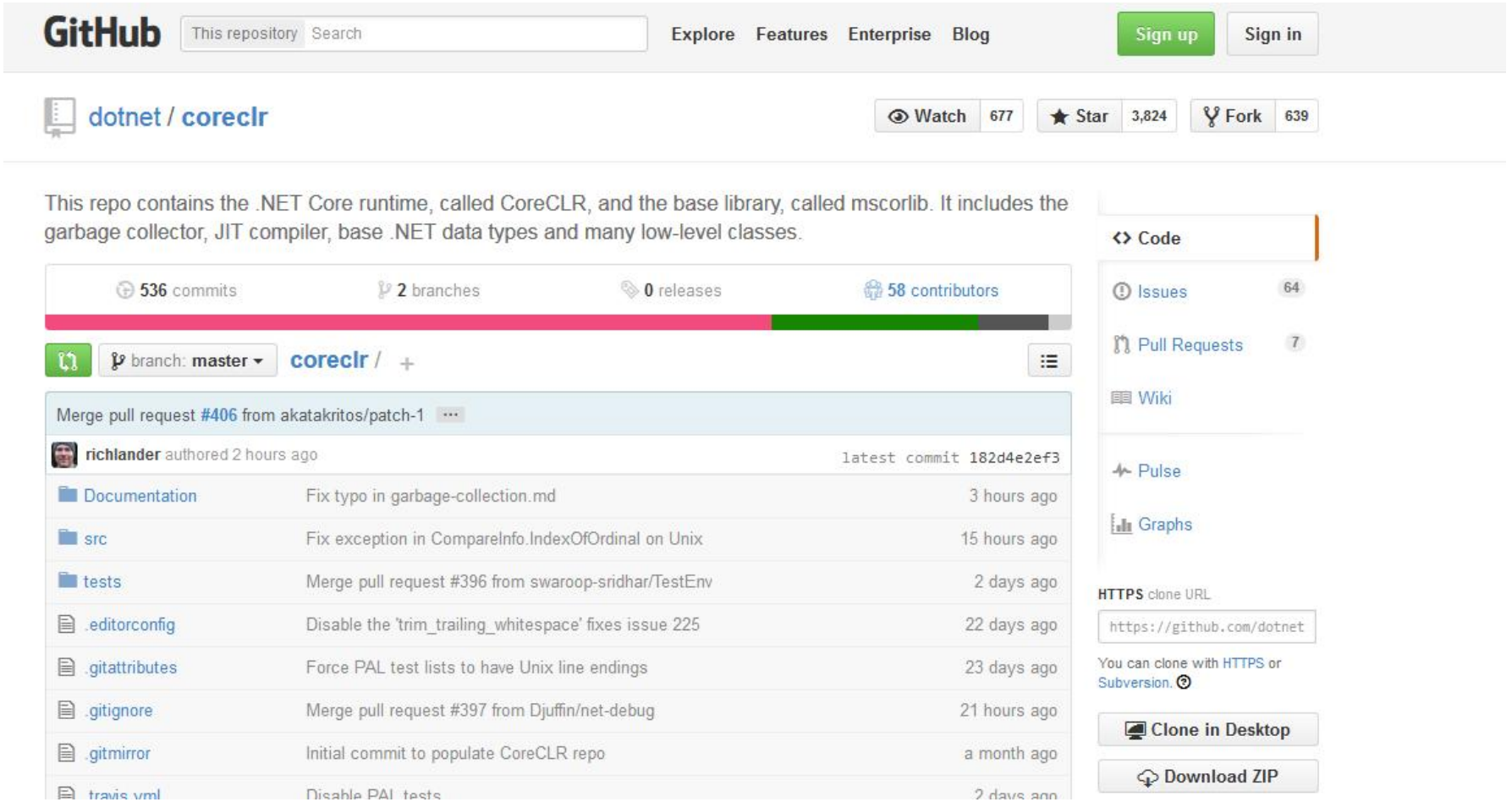
Příklady

- System
- System.Collections
- System.Diagnostics
- System.Globalization
- System.IO
- System.Security
- System.Text
- System.Threading
- Výčet není kompletní!

Směr komunikace



.NET CoreCLR


GitHub
This repository Search
Explore Features Enterprise Blog
Sign up Sign in
dotnet / coreclr
Watch 677 Star 3,824 Fork 639
This repo contains the .NET Core runtime, called CoreCLR, and the base library, called mscorlib. It includes the garbage collector, JIT compiler, base .NET data types and many low-level classes.
536 commits 2 branches 0 releases 58 contributors
branch: master coreclr /
Merge pull request #406 from akatakritos/patch-1
richlander authored 2 hours ago
latest commit 182d4e2ef3
Documentation Fix typo in garbage-collection.md 3 hours ago
src Fix exception in CompareInfo.IndexOfOrdinal on Unix 15 hours ago
tests Merge pull request #396 from swaroop-sridhar/TestEnv 2 days ago
.editorconfig Disable the 'trim_trailing_whitespace' fixes issue 225 22 days ago
.gitattributes Force PAL test lists to have Unix line endings 23 days ago
.gitignore Merge pull request #397 from Djuffin/net-debug 21 hours ago
.gitmirror Initial commit to populate CoreCLR repo a month ago
travis.yml Disable PAL tests 2 days ago
Code
Issues 64
Pull Requests 7
Wiki
Pulse
Graphs
HTTPS clone URL
https://github.com/dotnet
You can clone with HTTPS or Subversion.
Clone in Desktop
Download ZIP

SIEMENS

Obsah přednášky

- BCL
- **Správa paměti v .NETu**
- IDisposable
- Kolekce
- Streamy

SIEMENS

Správa paměti v .NET – Garbage Collector

- Alokace/dealokace v režii CLR (Common Language Runtime)

- Reprezentováno komponentou Garbage Collector
  - Třída `System.GC`
  - Periodicky odstraňuje objekty bez reference
  - Využívá systém generací
  - Kolekci je možná vynutit – `GC.Collect();`

```
private void AllocateMemory()
{
   //Memory is going to be released when execution
   //leaves the scope of method
   var wastedMemory = new string('X', 100000000);
}
```

- I v .NET je nutné řešit memory-leak (zapomenuté reference)

SIEMENS

Správa paměti v .NET - Finalizace

- Ekvivalent destruktorů v jazyce C
- Volá se před smazáním objektu

```
~ClassWithFinalizer()
{
  //Cleanup code
}
```

- Slouží jako „poslední šance“ k uklizení externích zdrojů (viz Dispose), lépe je ale uklidit explicitně!
- Složitý či blokující finalizační kód degraduje výkon
- Pozor na výjimky

- Finalizér je možné potlačit, či naplánovat znovu

```
GC.ReRegisterForFinalize(this);
GC.SuppressFinalize(this);
```

SIEMENS

Obsah přednášky

- BCL
- Správa paměti v .NETu
- **IDisposable**
- Kolekce
- Streamy

SIEMENS

IDisposable

- Rozhraní pro objekty vyžadující explicitní úklid
- Metoda `void Dispose();`
- Využívá např. `Component, Stream, SqlConnection, …`
- Pokud objekt drží jiné disposable objekty, sám by měl disposable implementovat
- Na volání metody Dispose z kontextu finalizace existuje vzor (viz dále)

SIEMENS

IDisposable – implementační vzor pro finalizaci

```
public void Dispose() // NOT virtual
{
  Dispose (true);
  GC.SuppressFinalize (this); // Prevent finalizer from running.
}

protected virtual void Dispose(bool disposing)
{
  if (disposing)
  {
    // Call Dispose() on other objects owned by this instance.
    // You can reference other finalizable objects here.
  }
  // Release unmanaged resources owned by (just) this object.
}

~DisposableClass()
{
  Dispose (false);
}
```

SIEMENS

Obsah přednášky

- BCL
- Správa paměti v .NETu
- IDisposable
- **Kolekce**
- Streamy

SIEMENS

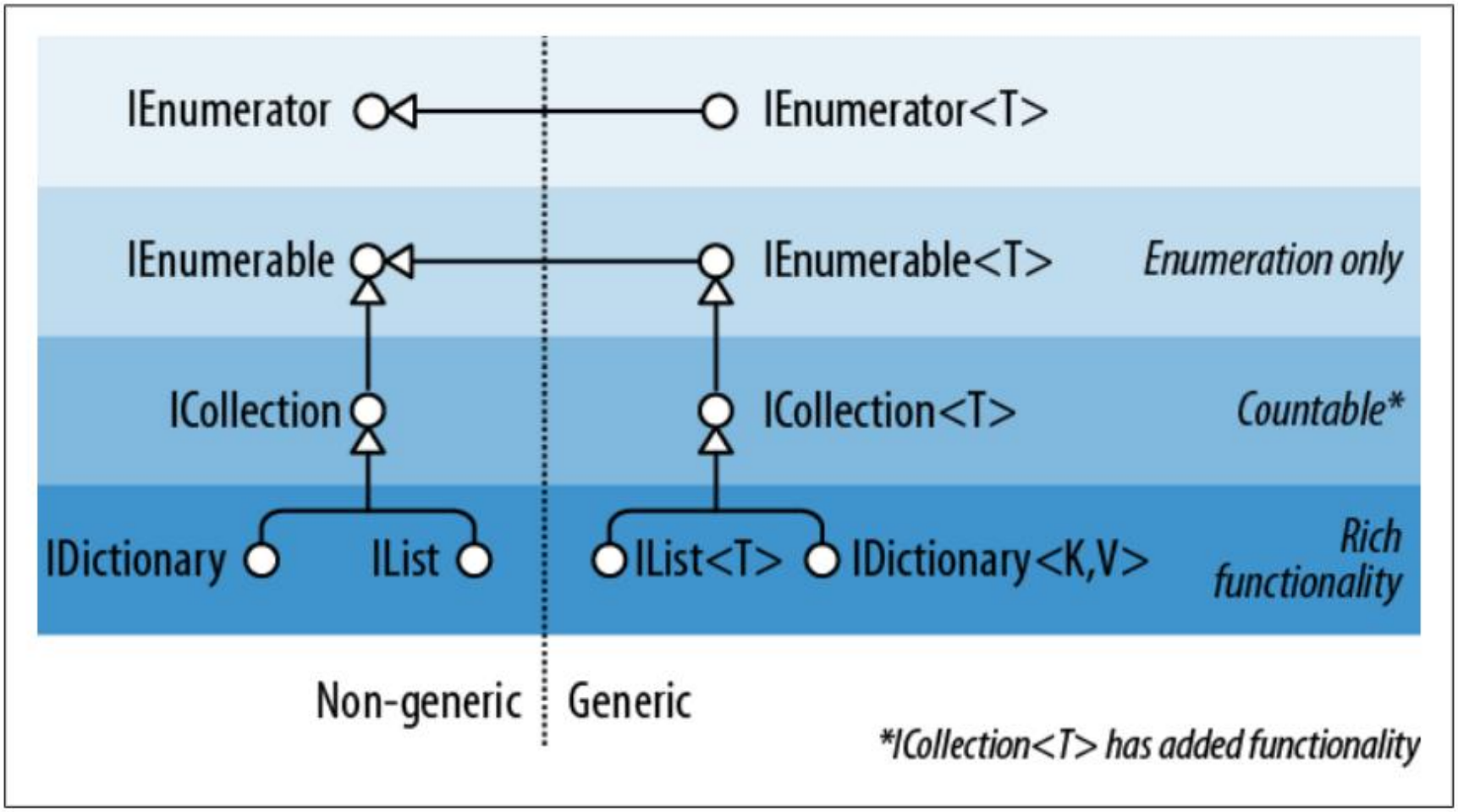
Kolekce - přehled

- **Negenerické**
  - Rozhraní - `IEnumerable, IEnumerator, IComparer, IEqualityComparer, IHashCodeProvider ICollection, IList, IDictionary`
  - Implementace - `BitArray, ArrayList, SortedList, HashTable, Queue, Stack`
  - Speciální – `NameValueCollection, OrderedDictionary, ListDictionary, HybridDictionary`
  - Bázové – `CollectionBase, DictionaryBase`

- **Generické**
  - Rozhranní – `IEnumerable<T>, IEnumerator<T>, IComparer<T>, IEqualityComparer<T> ICollection<T>, IList<T>, IDictionary<TKey,TItem>, IReadOnlyCollection<T>, IReadOnlyDictionary<TKey,TItem>, ISet<T>`
  - Implementace – `List<T>, SortedList<T>, LinkedList<T>, Dictionary<TKey,TItem>, SortedDictionary<TKey,TItem>, HashSet<T>, Queue<T>, Stack<T>`
  - Bázové – `Comparer<T>, Collection<T>, ReadOnlyCollection<T>`

  - Výčet není kompletní!

Rozhraní


IEnumerator
IEnumerator<T>
IEnumerable
IEnumerable<T>
Enumeration only
ICollection
ICollection<T>
Countable*
IDictionary
IList
IList<T>
IDictionary<K,V>
Rich functionality
Non-generic
Generic
*ICollection<T> has added functionality

SIEMENS

Kolekce - Enumerace

- Procházení kolekcí prvek za prvkem na základě `IEnumerable/IEnumerable<T>`
- Umožňuje použít klíčové slovo `foreach`

```
string[] strings = new string[] { "Hello", "World!" };
foreach (string str in strings)
{
  Console.WriteLine(str);
}
```

- Enumeraci je možné ovládat pomocí instance enumerátoru

```
IEnumerator<string> enumerator = strings.GetEnumerator();
string str = enumerator.Current;
bool success = enumerator.MoveNext();
```

- Enumeraci je možné implementovat pomocí `IEnumerator/IEnumerator<T>` nebo jednodušeji pomocí klíčového sousloví `yield return <value>;`

SIEMENS

Ukázkové použití enumerátoru

```
string sampleString = "Hello";

// Because string implements IEnumerable, we can call GetEnumerator():
IEnumerator enumerator = sampleString.GetEnumerator();

while (enumerator.MoveNext())
 {
  char sampleChar = (char)enumerator.Current;
  Console.Write(sampleChar + ".");
}

// Output: H.e.l.l.o.
```

SIEMENS

Kolekce – Implementace enumerátoru

```
public class MyGenCollection : IEnumerable<int>
 {
    int[] data = { 1, 2, 3 };

    public IEnumerator<int> GetEnumerator()
    {
        foreach (int i in data)
        yield return i;
    }

    // Explicit implementation
    IEnumerator IEnumerable.GetEnumerator()
    {
       // keeps it hidden.
       return GetEnumerator();
     }
}
```

ICollection, IObservableCollection

```
public interface ICollection<T> : IEnumerable<T>, IEnumerable
{
    void Add(T item);
    void Clear();
    bool Contains(T item);
    void CopyTo(T[] array, int arrayIndex);
    bool Remove(T item);
    int Count { get; }
    bool IsReadOnly { get; }
}
```

IObservableCollection

- Analogické INotifyPropertyChange
- Notifikuje mi změnu počtu

SIEMENS

Kolekce – praktické ukázky

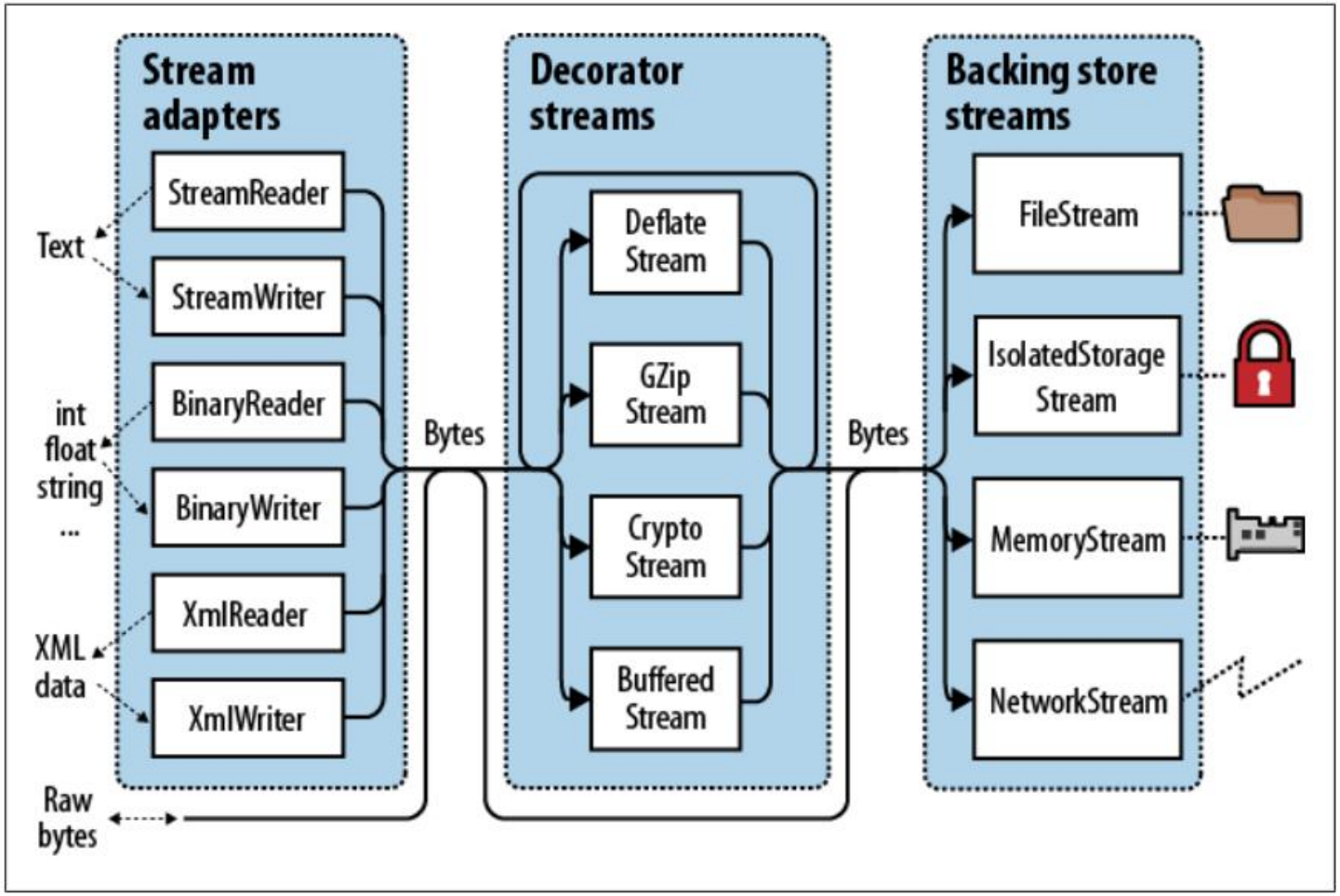
```
// Example 1 DoBasicCollectionOperations
examples.BasicCollectionOperations.DoBasicCollectionOperations();
Console.ReadKey();

//Example 2 - ArrayList, List<T>, IEnumerable, IEnumerable<T>
examples.ListEnumeration();
Console.ReadKey();

//Example 3 - Enumerator manipulation
examples.UsingEnumerator();
Console.ReadKey();

//Example 4 - Implementation of custom enumeration
examples.CustomEnumerator();
Console.ReadKey();

// ...
```

Obsah přednášky

- BCL
- Správa paměti v .NETu
- IDisposable
- Kolekce
- **Streamy**

SIEMENS

Architektura streamů



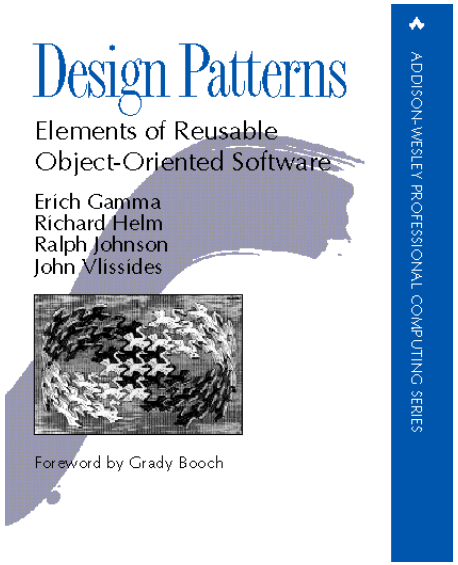
SIEMENS

Design patterns

- základní jsou v GoF.
- tři základní skupiny
  - Structural
  - Behavioral
  - Creational

- **Decorator (Wrapper)**
- Dynamicky přidává zodpovědnosti objektu v runtime.
- Vždy obsahuje původní dekorovaný typ.

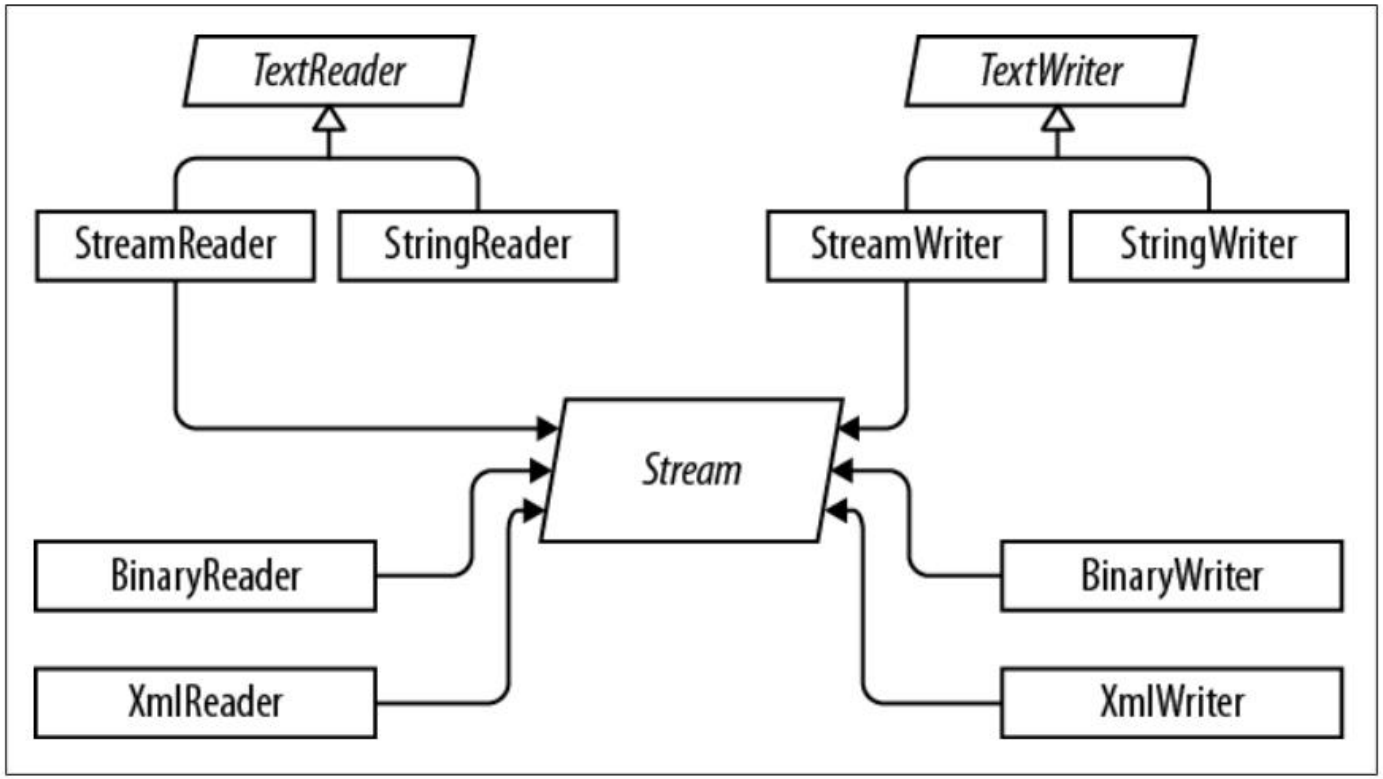
- **Adapter**
- Změna rozhraní třídy na jiné rozhraní.



Použití

```
using (Stream s = new FileStream("test.txt", FileMode.Create))
{
    Console.WriteLine(s.CanRead); // True
    Console.WriteLine(s.CanWrite); // True
    Console.WriteLine(s.CanSeek); // True
    s.WriteByte(101);
    s.WriteByte(102);
    byte[] block = { 1, 2, 3, 4, 5 };
    s.Write(block, 0, block.Length); // Write block of 5 bytes
    Console.WriteLine(s.Length); // 7
    Console.WriteLine(s.Position); // 7

// ...
```

Adaptéry


TextReader
TextWriter
StreamReader
StringReader
StreamWriter
StringWriter
Stream
BinaryReader
BinaryWriter
XmlReader
XmlWriter

SIEMENS

Compressions streams

```
using (Stream s = File.Create("compressed.bin"))
{
   // DeflateStream == decorator
   using (Stream ds = new DeflateStream(s, CompressionMode.Compress))
   {
      for (byte i = 0; i < 100; i++)
      {
         ds.WriteByte(i);
      }
   }
}
```

SIEMENS

Práce se Zip soubory

```
// create file from directory
ZipFile.CreateFromDirectory(// …);

// extract zip archive
ZipFile.ExtractToDirectory// …);

// enumerate existit entries in archive
using (ZipArchive zip = ZipFile.Open(// …)
 {
   foreach (ZipArchiveEntry entry in zip.Entries)
   {
      Console.WriteLine(entry.FullName );
   }
}
```

Operace se soubory a adresáři

Operace se soubory a adresáři

- BCL obsahuje předpřipravené třídy File, Directory, FileInfo, DirectoryInfo

IsolatedStorage

- Standardizovaná cesta pro ukládání dat desktopových aplikací
- Zabezpečený přístup
- V uživatelské složce
- Unikátní na assembly

Kontakt

Mgr. Radoslav Čáp
Siemens CT DC / Sitraff Team

Olomoucká 7/9
618 00 Brno
Česká republika

E-mail:
radoslav.cap@siemens.com

SIEMENS

Reference

BCL

- .NET Framework Class Library Overview
- .NET Core zdrojové kódy
- Base Class Libraries
- .NET Framework Class Library

Nástroje

- dotPeek Free .NET Decompiler and Assembly Browser
- LINQPad

- Design patterns

- C# in Nutshell