Generiska typer

Introduktion - C# in depth

Del 1 - Kap 1&2 - (C# 1-5) Resan, (C# 1) Kärnan

Del 2 - Kap 3-7 - (C# 2) ‘Generic types’, ‘Nullables’, ‘Delegates’, ‘Iterators’, Övrigt

Del 3 - Kap 8-12 - (C# 3) ‘Compiler’, ‘Lambda expressions’, ‘extension methods’, ‘Query’, ‘LINQ’

Introduktion - C# in depth

Del 4 - Kap 13&14 - (C# 4) ‘COM’, ‘variance’, ‘locking’, ‘dynamic’ (bindning, typning, mm)

Del 5 - Kap 15-16 - (C#5) ‘Async/await’, ‘asynchrony’

Appendix A-C

Idag? (fm/em)

- Generiska typer
- Iterator (IEnumerator)
- Generiska reflection
- Nullable<T>
- Generiska - övrigt
- Databasinriktad exempel
- Övning

Man kan ladda ner presentationen med övningskoden.

Generiska typer repetition

Kapitel 3 (s.59-104)

Klass<T1,T2> {}
Klass<T> : Superclass {}

synlighet returtyp Metod (paramtyp param) {}
private T Method (T t) { return t; }
public List<T> ListMethod<T> () {return new List<T>();}

Generiska typer repetition

Varför? -> typsäkerhet

ArrayList arrayList = new ArrayList();
arrayList.add(5);
arrayList.add(“Hej”); Nej

Fungerar i kompilatorn men INTE runtime.

Generiska typer repetition

Samma problem med typomvandling...

```
ArrayList arrayList = new ArrayList();
arrayList.add(“Hej”);

int x = (int)arrayList.get(0); Nej
```

Generiska typer repetition

Andra fördelar:

- Mindre ‘overhead’ / mindre att kolla innan körning (snabbare)
- JIT har sätt att köra det utan ‘boxing/unboxing’
- Lättare att läsa och förstå t.ex List<string> än ArrayList? i andras kod

Generiska typer repetition

Exempel:

List<T>

Dictionary<TKey,TValue>

IEnumerable<T>, IEnumerator<T>

IComparable<T>

IEquatable<T>

IObservable<T>, IObserver<T>

Generiska typer - constraints

Constraints (s.71-76)

struct<T> : Where T class

-> Accepterar bara klasser som T (ie. int[] och string men inte int)

class<T> : Where T struct

-> Accepterar bara struct som T (ie. int men inte string)

Generiska typer - constraints

Som arv: hur många ‘interfaces’ du vill men bara en klass

class Example<T> where T : MyClass, IEnumerable<string>, IComparable<int> Ja

Men inte : ArrayList, List<int> Nej

eller värdetyper (som int), string, object, enum, delegate

ser s.75 för några bra exempel

Generiska metoder - Type inference

```
static List<T> MakeList<T>(T first, T second)
{ … }

List<string> list = MakeList<string>(“hej”, “du”);
även:
List<string> list = MakeList(“hej”, “du”);
men inte: = MakeList(“hej”, minObjekt);
    även om man har T first, U second
```

Generiska metoder - Type inference

```
Public static class Pair {
    public static Pair<T1,T2> Of<T1,T2>(T1 first,T2 second)
    {
        return new Pair<T1,T2>(first,second);
    }
} ...
Pair<int,string> pair = Pair.Of(10, “value”);
```

Generiska metoder

```
public static double Sqrt<TNumber>(TNumber value)
{
    return Math.Sqrt(Convert.ToDouble(value)); Ja
}
```

public static add<T,U>(T t, U u) { t + u; } Nej

(men med ‘dynamic’ nyckelordet kan man kom undan med det här)

https://github.com/robinos/GenericsAndIteratorConsoleExamples
Google
ÅTKOMST VALV
DELA VIA FACEBOOK

robinos / GenericsAndIteratorConsoleExamples

Unwatch 1 | Star 0 | Fork 0

Examples with generic classes/methods and iterators in C# — Edit

5 commits | 1 branch | 0 releases | 1 contributor

branch: master | GenericsAndIteratorConsoleExamples / +

license info

robinos authored 4 days ago — latest commit b8c3c7acc4

GenericListIteratorConsoleExample	license info	4 days ago
.gitattributes	Initial commit to add default .gitIgnore and .gitAttribute files.	10 days ago
.gitignore	Initial commit to add default .gitIgnore and .gitAttribute files.	10 days ago
Changelog.txt	Added Changelog and README	10 days ago
GenericListIteratorConsoleExampl...	Added License text	4 days ago
License.txt	Added License text	4 days ago
README.txt	Added Changelog and README	10 days ago

README.txt

```
Version 0.1.0
This Solution was created to house examples of generic classes/methods and
iterators in C#.
```

Code
Issues 0
Pull Requests 0
Wiki
Pulse
Graphs
Settings

HTTPS clone URL

https://github.cc

You can clone with HTTPS, SSH, or Subversion.

Clone in Desktop

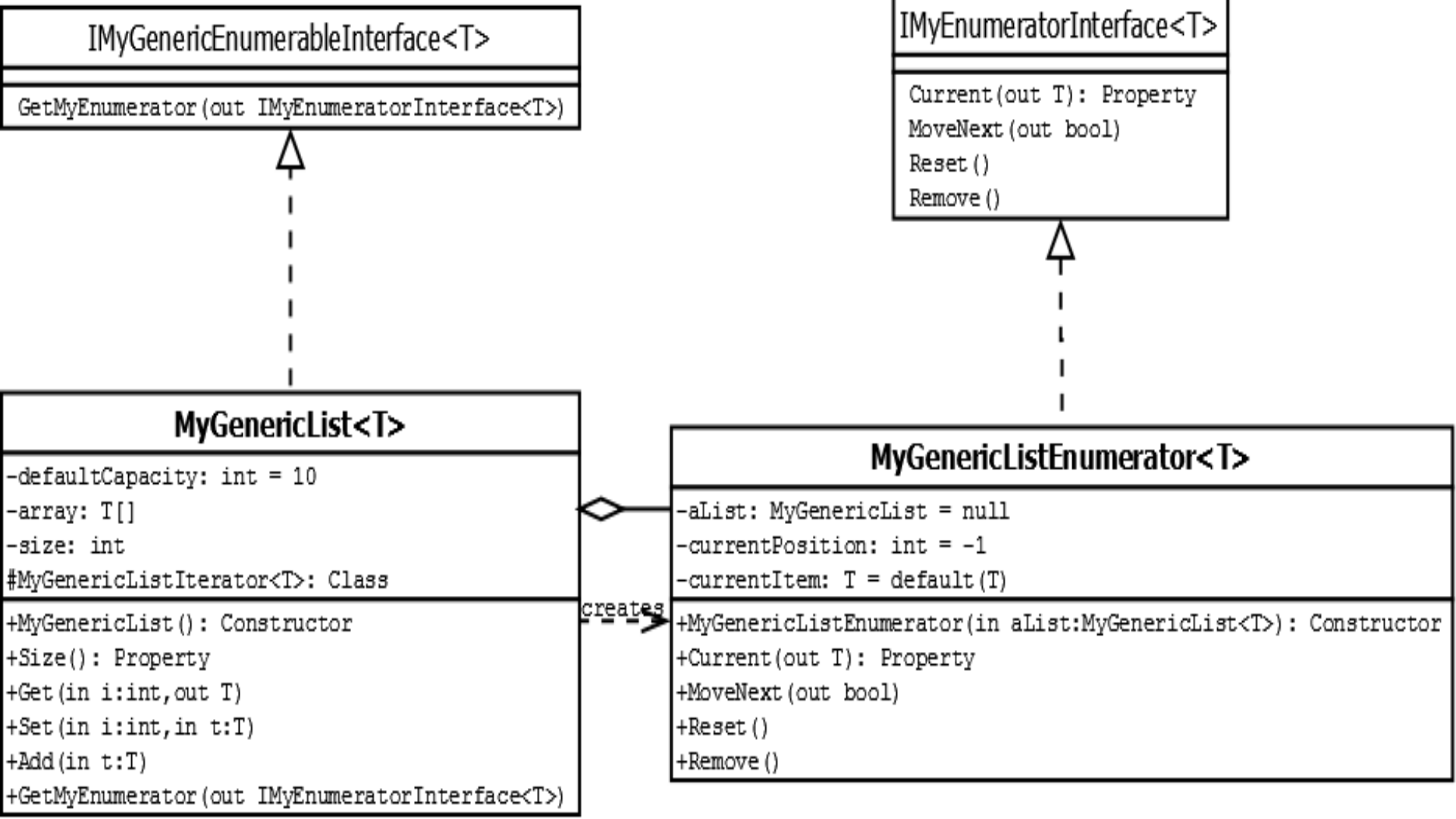
Download ZIP

Exempel: generisk lista / iterator

- Tillämpning med egenbyggd generisk lista och iterator i ‘liknande’ stil som C#.

Projekt: GenericListIteratorConsoleExample

https://github.com/robinos/GenericsAndIteratorConsoleExamples


IMyGenericEnumerableInterface<T>
GetMyEnumerator(out IMyEnumeratorInterface<T>)
IMyEnumeratorInterface<T>
Current(out T): Property
MoveNext(out bool)
Reset()
Remove()
MyGenericList<T>
-defaultCapacity: int = 10
-array: T[]
-size: int
#MyGenericListIterator<T>: Class
+MyGenericList(): Constructor
+Size(): Property
+Get(in i:int,out T)
+Set(in i:int,in t:T)
+Add(in t:T)
+GetMyEnumerator(out IMyEnumeratorInterface<T>)
creates
MyGenericListEnumerator<T>
-aList: MyGenericList = null
-currentPosition: int = -1
-currentItem: T = default(T)
+MyGenericListEnumerator(in aList:MyGenericList<T>): Constructor
+Current(out T): Property
+MoveNext(out bool)
+Reset()
+Remove()

IEnumerable<out T> : IEnumerable

Från sida 87-90

IEnumerable<T> ligger till grund för itererbar klasser som List och Dictionary bland annat.

Kräver att man implementera:

IEnumerator<T> GetEnumerator()

IEnumerator<T> : IDisposable, ...

Iterator klassen som tillåter Foreach loopar.

Kräver att man implementera:
egenskap: T Current { get; set; }

metoder: bool MoveNext(),
Reset() eller throw NotSupportedException (COM interop),
IDisposable.Dispose() som kan vara tom

IEnumerator<T> : …, IEnumerator

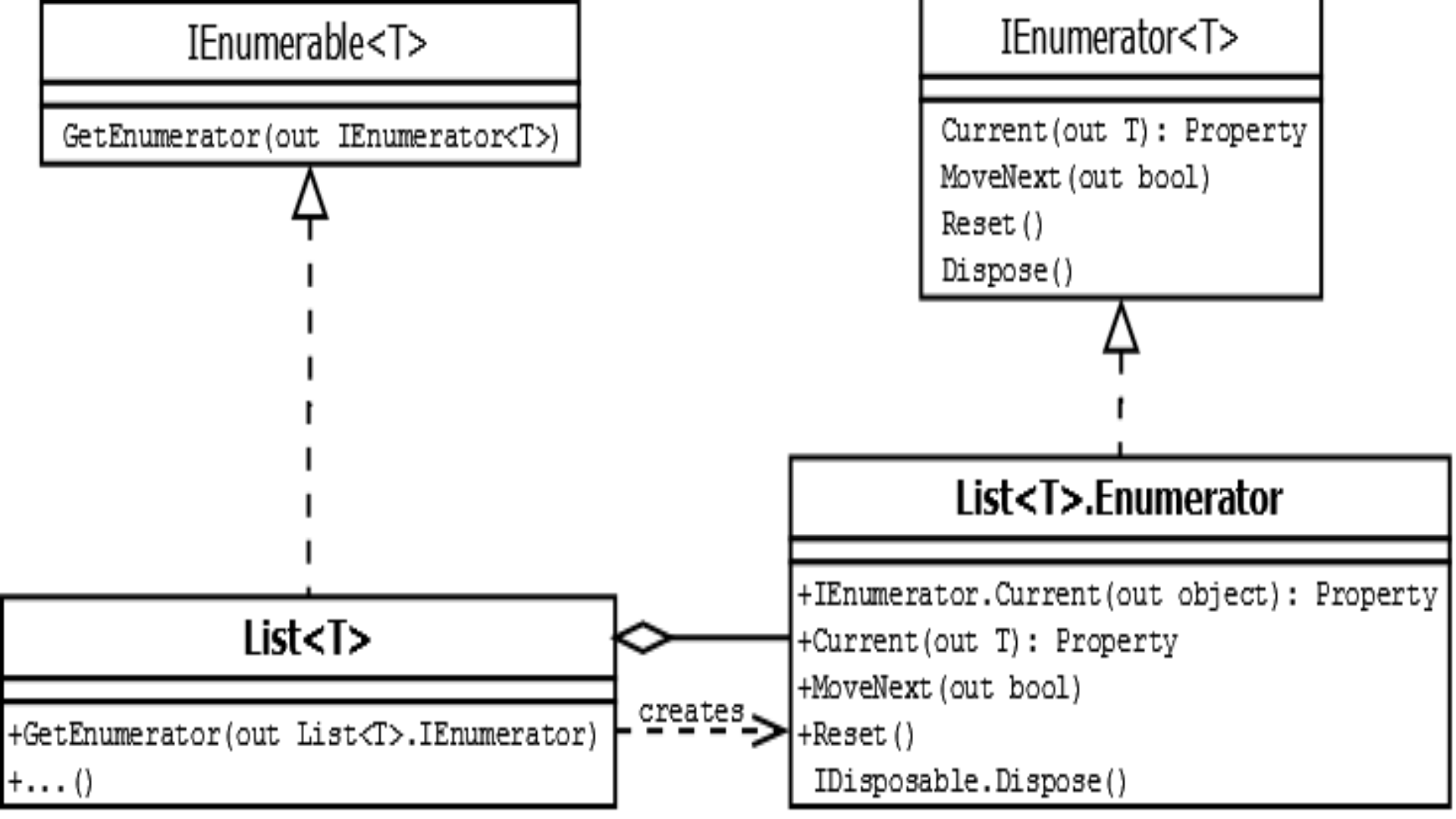
Extra komplexitet: IEnumerator<T> kräver även implementering av IEnumerator.

egenskap: object IEnumerator.Current { get; set; }

Bra exempel av Microsoft själv med Box klassen

(sök c# IEnumerator<T>)

*Notera: Inte trådsäkert!


IEnumerable<T>
GetEnumerator(out IEnumerator<T>)
IEnumerator<T>
Current(out T): Property
MoveNext(out bool)
Reset()
Dispose()
List<T>.Enumerator
+IEnumerator.Current(out object): Property
+Current(out T): Property
+MoveNext(out bool)
+Reset()
IDisposable.Dispose()
List<T>
+GetEnumerator(out List<T>.IEnumerator)
+...()
creates

Factory metod designmönstret

IEnumerable kräver att man tillämpa IEnumerator, samma som IMyEnumerable kräver IMyEnumerator i egenbyggd exemplet.

Både är exempel av ‘Factory method’ designmönstret

- en class/interface som kräver att man implementera en annan class/interface för att garantera viss funktionalitet (i det här fallet iteration för foreach) men inte exakt hur man väljer att implementera det.

Iterator med yield (lätt version C# 2)

```
public IEnumerator GetEnumerator() {
for(int index=0; index<values.Length,index++)
    {
        yield return values[(index + startingPoint) %
            values.Length]
    }
}
```

Exempel: generisk lista / iterator

- Tilllämpning med IEnumerable<T> med en GetEnumerator metod som använder ‘yield return’

Projekt: GenericListEnumeratorConsoleExample

https://github.com/robinos/GenericsAndIteratorConsoleExamples

Reflection repetition

- Anropa sig själv
- Kör till -slut tillstånd- och/eller -absolut slut-
- Måste kunna ta slut i **varje** fall - största problemet brukar vara oändliga loopar

ie. Söker under alla undermappar för filnamn ‘temp.txt’ och avsluta när det hittas ELLER om det inte hittades.

Reflection med generiska typer

sidor 90-94

Generiskt typdefinition

- typeof(List<>) ger List’1[T]
- typeof(Dictionary<,>) ger Dictionary’2[TKey,TValue]

Konstruktuerad typdefinition

- typeof(T) ger **t.ex.** System.Int32
-typeof(List<T>) ger **t.ex.** List’1[System.Int32]

Reflection (generisk utan dynamic)

```
public static void PrintTypeParam<T>()
{ Console.WriteLine(typeof(T)); }
...
Type type = typeof(Snippet);
MethodInfo definition = type.GetMethod("PrintTypeParam");
MethodInfo constructed = definition.MakeGenericMethod
(typeof(string));
constructed.Invoke(null, null);
```

Exempel: Reflection (generisk)

- Tillämpning med Labyrint-exemplet

https://github.com/robinos/LabyrinthWindowsForms

Generiska typer: Övrigt

sidor 94-104

Invariant!

List<Animal> animals = new List<Cat>(); Nej, kompilerar inte

animals.Add(new Cat()); Trots att det här är okej, varför?

*Kom ihåg att MyGenericList använde en array egentligen.

Generiska typer: Övrigt

Arrays är covariant.

Även med en array fungerar animals[0] = new Cat(); Ja

Animals animals[] = new Cat[] kompilerar men körs inte. Den egentligen kräver att *samlingen* Cat[] = new Cat[] även om *individuella objekt kan vara subklasser*.

Skillnaden: Den generiska klassen berättar om problemet redan vid kompilering. En array håller tyst.

Generiska typer: Övrigt

Sida 97 och 98 visar hur man kan kom omkring problemet i en exempel fall

- Cat cat = (Cat)animal(0); är minst önskvärt
- List<T>.ConvertAll
- implementera egen interface
- Enumerable.Cast<T> & Enumable.OfType<T> i LINQ
- class helper<TBase,TDerived> skickar vidare superklassen

struct Nullable<T> / class Nullable

Snabb genomgång! Läsa för djupare kunskap.

Sidor 105-132

Nullable<T> gör en override på GetHashCode, ToString och Equals.

Nullable klassen ger override metoderna.

struct Nullable<T> / class Nullable

T kan konverteras (boxing) till Nullable<T> (som generad kod ofta gör åt en) och från Nullable<T> till T som unboxing.

GetValueOrDefault() - värdet eller 0

int? nullable -är samma som- Nullable<int> nullable

int c = a ?? b; b blir default värdet om a är null

struct Nullable<T> / class Nullable

Nullable integers (s.120) : int? nullInt = null, five = 5;

-nullInt -> null

nullInt + five -> null

nullInt==nullInt -> true

nullInt<=nullInt -> false

logical operators (s.122) med bool?

Exempel med databasinriktning

- Dictionary<string,Film> till FilmBox combobox.

- Innehåll av label-objekt: Årtal, Längd och FilmBild ändrar sig beroende på vilken film man har valt.

Exempelkod: https://github.com/robinos/FilmListaWindowsForms

Övning: Databas till Dictionary

- Dictionary<int,T> där int är tabell ID och T är en objekt som håller resterande rader.

- Fyll en ComboBox och några Label-objekt med data manuellt

Övning: Databas till Dictionary

- Databas tabell och kodkopplingen är redan gjort, fyll i en ComboBox själv från en Dictionary<int, Atom>

- Exempelkoden är Windows Forms men WPF går bra

Övningskod: https://github.com/robinos/AtomerWindowsForms

Där finns även AtomerWindowsForms.pdf med beskrivningen och koden.