# C# Almarnak

!!! = Skal rettes / indsættes

Historik

|  |  |
| --- | --- |
| 12-12-2018 | Dokument oprettet |
| 13-12-2018 | Tilføjet identifiers og keywords Kommentarer |
| 14-12-2018 | Felter/fields tilføjet  Constructor  this |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |

# Conventions used in this book

# A first c# program

# Syntax

## Keywords

Keywords I C# er predefinerede, reserverede identifiers som har en mening for compileren. Dvs. Det er ord man ikke selv kan benytte.

Første tabel her er keywords, eller identifiers som ikke kan benyttes noget sted i dit program da de er reserverede.

|  |  |  |  |
| --- | --- | --- | --- |
| abstract | as | base | Bool |
| break | byte | case | Catch |
| char | checked | class | Const |
| continue | decimal | default | Delegate |
| do | double | else | Enum |
| event | explicit | extern | False |
| finally | fixed | float | For |
| foreach | goto | if | Implicit |
| in | int | interface | Internal |
| is | lock | long | Namespace |
| New | Null | Object | Operator |
| out | override | params | Private |
| protected | public | readonly | ref |
| return | sbyte | sealed | Short |
| sizeof | stackalloc | static | String |
| struct | switch | this | throw |
| true | try | typeof | Uint |
| ulong | unchecked | unsafe | Ushort |
| using | Using static | virtual | Void |
| volatile | while |  |  |

## Contextual keywords

I C# findes der også noget der kaldes contextual keywords. Det er keywords som har en speciel betydning, men ikke er reserveret til compileren. Det kan fx være ordet ”add”. Det har i visse tilfælde en speciel betydning, men du kan staddig godt selv lave en metode der hedder add.

Nogle af ordene har betydning mere en ét sted

|  |  |  |
| --- | --- | --- |
| add | alias | Ascending |
| async | await | By |
| descending | dynamic | Equals |
| from | get | Global |
| group | into | Join |
| let | nameof | On |
| orderby | Partial (type) | Partial (metode) |
| remove | select | set |
| value | var | When (Filter condition) |
| Where (generic type constraint | Where (query clause) | yield |

## Kommentarer

Der findes to måder at lave kommentarer på.

// Dette på en kommentar på 1 linje

/\* Dette er  
 en kommentar  
 på flere linjer \*/

XML Dokumentation

I C# har man mulighed for at gøre brug af XML dokumentation.  
Der er flere fordele ved at bruge XML dokumentationen. Bl.a muligheden for automatisk at generere dokumentation for sin kode. Lad os se på et eksempel:

!!! Indsæt eksempel fra kodeeksemplerne

# Type basic

# Numeric types

# Boolean type and operator

# Strings and characters

# Arrays

# Variables and parameters

# Expressions and operators

# Null operators

# Statements

# Namespaces

# Classes

## Felter

Et felt er en variabel der tilhører klassen

!!! Indsæt eksempel

Laves altid private og siddes vha properties (!!!)

Et felt kan også være readonly. Det gør at det ikke kan ændres efter det er blevet sat.  
Du kan kun sidde værdien af en readonly når den bliver deklereret eller i constructoren til den klasse der indeholder variabelen.  
Man behøver ikke at sidde et felt fra starten. Så vil det have default værdierne (0,null, false osv).

Du kan også deklererer flere felter af gangen så længe det er samme type

!!! indsæt eksempel static readonly int legs = 2, eyes = 2

## Constructors

En constructor bliver brugt til at initialisere kode I en klasse eller struct. Den bliver defineret som en metode, men modsat en ”almindelig” metode har den ingen retur-type og navnet er det samme som den klasse den ligger i.

!!! Eksempel

En constructor kan fra C# 7.0 og frem skrives lidt mere forenklet

!!! public Panda (string n) => name = n;

En klasse og struct kan overloade deres constructors. Det kræver bare at parametrene er forskellige eller i en anden rækkefølge.   
Så følgende to constructors vil IKKE virke  
!!! Ensemepl

En constructor kan også kalde en anden constructor  
!!! public wine(decimal price, int year) : this(price) {….}

Når man fra en constructor kalder en anden constructor vil den kaldte constructor blive kørt først.

## Implicit parameterless constructor

Hvis du opretter en klasse uden en constructor vil compileren automatisk oprette en poarameterløs constructor. Men lige så snart du opretter en constructor laver compileren ikke længere en parameterløs constructor

## Metoder

### Beskrivelse Signatur Metodekald Parameter Return værdier

## This

This keywordet refererer til instancen af objektet.  
public class Plante{  
Urtepotte urtepotten;  
public void PlantIUrtepotte(Urtepotte urtepotten){  
  
urtepotten = urtgepotten  
urtepotte.Plante = this;

}

}

Det ses bl.a når man sidder felter for at skelne dem fra parameter hvis de hedder det samme

!!! eksempel

Public Elev(string navn. Int alder){  
 this.navn = navn;  
 this.alder = alder

}

# Inheritance

# The object type

# Structs

# Access modifiers

# Interfaces

# Enums

# Nested types

# Genrics

# Delegates

# Events

# Lambda expressions

# Anonymous methods

# Try statement and exceptions

# Enumeration and iterators

# Nullable types

# Extension methods

# Anonymous types

# Tuples

# Linq

# Dynamic binding

# Operator overloading

# Attributes

# Caller info attributes

# Asynchronous functions

# Unsafe code and pointers

# Preprocessor directives

# Xml documentation