Programmeren Gevorderd

Les 2 // Overerving en polymorfisme

Inheritance/overerving

Overerving

Altijd basisklasse die ‘geïmplementeerd’ wordt

3 Methodes:

1. ‘Standaard’ overerving
2. Abstracte basisklasse
3. Interface

‘Standaard’ overerving

Een basisklasse met functionaliteit

Wordt geïmplementeerd om functionaliteit toe te voegen

Of aan te passen via virtual/override keywords

Telkens maar één basisklasse

Bv. GSM -> Smartphone

Voorbeeld: RegularInheritance

Abstracte klasse

Lijkt op ‘gewone’ implementatie

Verschil: abstracte klasse kan niet geïnstantieerd worden, moet altijd geïmplementeerd worden

Kan abstracte methodes & properties bevatten

Kan ook functionaliteit bevatten

Telkens maar één abstracte basisklasse

Bv. voertuig -> auto, zeilboot

Voorbeeld: AbstractInheritance

Interfaces

Afspraak over publieke properties & methods

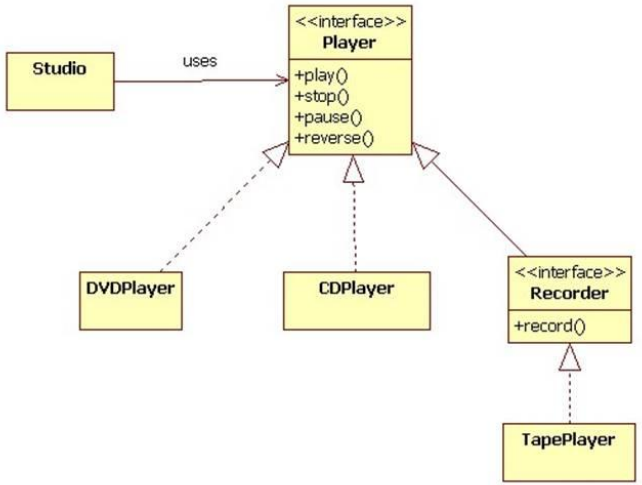
Geen fields of constructoren

Sinds C# 8 wel default implementaties voor properties & methods (over te erven zonder override)

Meerdere interfaces kunnen geimplementeerd worden

Beperkingen:

- Geen fields
- Geen access: alles is public
- Kan niet overerven van een klasse: wel van andere interfaces



Voorbeeld: Interface

Is & as operatoren

Is

Wordt gebruikt om te kijken of iets een bepaald type heeft

Returned true/false

As

Wordt gebruikt om iets om te zetten naar een bepaald type

Returned de variabele als nieuw type of returned null indien onmogelijk

Returned cast value of null, nooit exception

Is operator kan sinds C# 7 gebruikt worden om cast te controleren alsook door te voeren

4 kernprincipes OOP

1. Abstractie - relevantie attributen en gedragingen van een entiteit omzetten naar klassen
2. Encapsulatie - afschermen welke functionaliteit intern blijft en welke extern toegankelijk is
3. Overerving - klassen kunnen functionaliteiten overerven van andere klassen, herbruikbaarheid
4. Polymorfisme - klassen kunnen meerdere vormen aannemen

Keywords

Static

Static = zonder instantiëring

Static klassen kunnen niet geïnstantieerd worden, kunnen enkel static methods, properties, etc bevatten

Static functies, properties & fields kunnen enkel aangeroepen worden op het type, niet op instanties

Static functies, properties & fields kunnen in niet static klassen gebruikt worden door niet statics

Sealed

Klasse kan niet overgeërfd kan worden door andere klassen

Sealed klasse zelf kan wel klassen overerven

Geeft aan dat deze klasse altijd bovenaan moet liggen in overerving hiërarchie

Niet zo nuttig om zelf te gebruiken,

Vooral gebruikt in standaard C# klassen, dus wel goed om te begrijpen

ReadOnly

Meestal gebruikt voor fields

Duidt aan dat een value van een field enkel kan aangepast worden bij de declaratie van het field & in de constructor

Bij value types = value kan daarna niet meer veranderen

Bij reference types = reference naar object kan niet meer veranderen, object zelf wel

Voorbeeld: Keywords

Objecten vergelijken

Equals()

Wordt gebruikt om te bepalen of twee objecten gelijk zijn aan elkaar

Equals returned boolean

Wordt intern door veel methodes gebruikt

Vergelijk op properties & fields naar keuze

GetHashCode()

GetHashCode returned numerieke waarde

Wordt gebruikt in hash-based collecties zoals dictionary & hashset

Maak gebruik van GetHashCode() methode op properties

Combineer hashcode van meerdere properties met XOR (^) operator of HashCode.Combine()

Gelijke objecten geven gelijke hashcodes

Verschillende objecten niet altijd verschillende hash codes

Voorbeeld: ObjectMethods

Comparison

IComparable

Interface om objecten van hetzelfde type vergelijkbaar te maken

Wordt gebruikt om te ordenen & sorteren

Wordt geimplementeerd door alle numerieke types, string, char & datetime

CompareTo() methode returned waardes kleiner dan 0, groter dan 0 of 0 om volgorde in gesorteerde set te bepalen

Comparer<T>

Implementatie van interface IComparer<T>, heeft ook niet generieke variant

Heeft ook CompareTo() methode waarvan return values gelijkaardig systeem volgen

Wordt gebruikt om alternatieve sortering te doen, voor standaard sortering gebruik je IComparable interface

Voorbeeld: Comparison