Programmeren Gevorderd

Les 1 // Klassen en objecten

Introductie

Wie ben ik

Ewout Eetesonne

Werk voornamelijk als Unity Developer (C#)

Hoe verlopen de lessen

Fysiek op Hogent

Geen verplichte aanwezigheid

Wat willen we bereiken dit semester

Werkende code ≠ goede code

Goede code = werkende code

Gereedschapskist uitbreiden

Juiste tools op juiste moment leren gebruiken

Structureel nadenken over code

Structuur semester

*Structuur kan nog wijzigen doorheen het semester

Basis programmeren

Verdieping object oriented programming

DCD

Collections

Nieuwe leerstof

Drie lagen design pattern

Unit Testing

Linq

Events

Datum: TBD

Tussentijdse Evaluatie

Eerste zit

Tweede zit

WPF

ADO.NET

Datum: TBD

Eindevaluatie

Eerste zit

Tweede zit

Herhaling Object Oriented Programming

Basis OOP

Klasse = abstract concept

Object = concrete instantie van een klasse

Access modifiers

Bepalen wat van waar toegankelijk is

Van toepassing op fields, methods, properties

- Private = enkel binnen object
- Protected = enkel binnen object + overervende klassen
- Internal = buiten object binnen project
- Public = toegankelijk buiten object

Voorbeeld: AccessModifiers

Fields

Geïnstantieerde objecten van een klasse kunnen verschillen tussen objecten

Deze verschillen duiden we aan onder andere als fields

Fields hebben altijd type, klasse zelf is ook een type

Properties

Fields ≠ properties

Property bevat altijd getter en/of setter

Interne state van object niet rechtstreeks aanpasbaar te maken van buitenaf

Methods

Klassen bevatten naast fields (beschrijven state) ook methods

Methods beschrijven wat klassen kunnen doen

Constructors

Vereiste parameters bij aanmaken en/of code uitvoeren

Een klasse heeft altijd 1 constructor, default indien niet expliciet

Meerdere constructors mogelijk (constructor overload)

Voorbeeld: ClassObject

Volgorde in klassen

Binnen een klasse of interface

1. Constant Fields
2. Fields
3. Constructors
4. Finalizers (Destructors) ← *les over events*
5. Delegates ← *les over events*
6. Events ← *les over events*
7. Enums ← *geen te kennen leerstof*
8. Interfaces (interface implementations)
9. Properties
10. Indexers ← *geen te kennen leerstof*
11. Methods
12. Structs← *geen te kennen leerstof*
13. Classes

Access modifiers volgorde

1. public
2. internal
3. protected internal
4. protected
5. private

Value & reference types

Value & reference types

Value type

Eenvoudige types: bool, int, enum

Default value

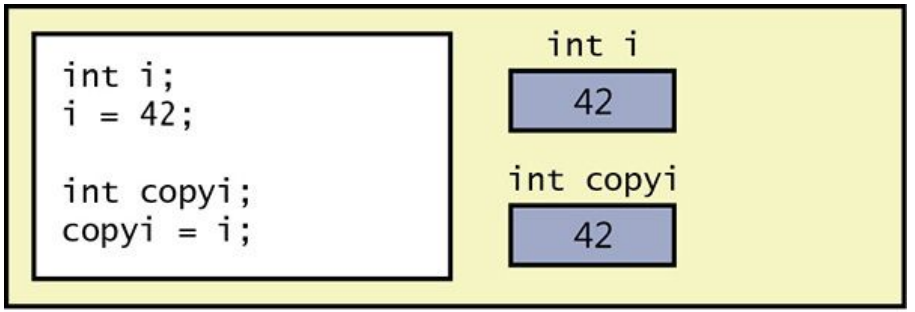
Altijd kopie van value

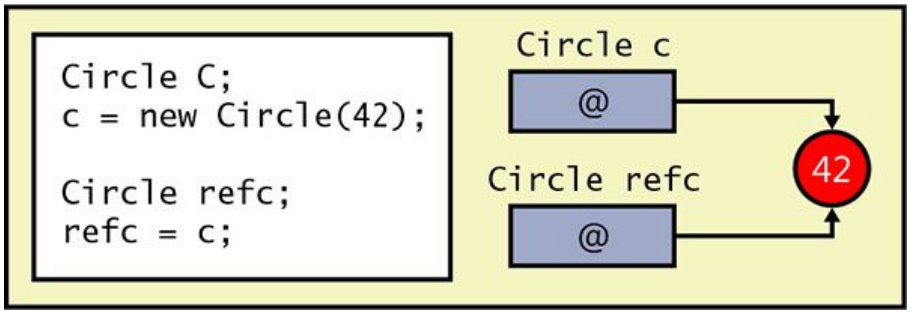
Reference type

Complexe types: class, array

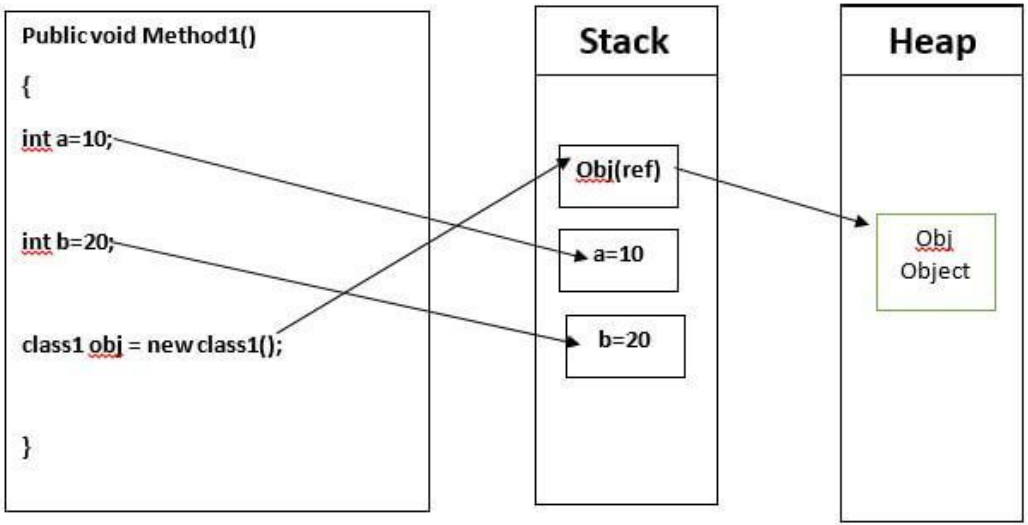
Default null

Altijd verwijzing naar elkaar





Stack & heap


Public void Method1()
{
int a=10;
int b=20;
class1 obj = new class1();
}
Stack
Obj(ref)
a=10
b=20
Heap
Obj
Object

Objecten vergelijken

Value types

Gemakkelijk te vergelijken met == operator

Reference types

Met == operator vergelijk je referentie en geen waarden

Oplossingen:

Custom vergelijking schrijven

Equals methode uit root System.Object overschrijven

Specialleke: string

Reference type : in memory

String kan tot 4GB groot zijn in memory

Stack max 4mb groot (64 bit)

Value type : in code

Voorbeeld: ValueReferenceTypes