.Net Essentials

## Hoofdstuk 1: De achtergronden van C#

## Hoofdstuk 2: De C#-ontwikkelomgeving

IDE (Integrated Development Enviroment):

Dit is een front-end voor enkele tools (bijvoorbeeld debugger, code editor, comiler, UI..)

Event-Handler:

Zorgt dat elke control kan reageren op events, het is een speciale methoden die uitgevoerd wordt.

## Hoofdstuk 3: Inleiding tot het gebruik van graphics

Overloaded methodes: Methodes met dezelfde naam maar andere of ander aantal parameters

Image toevoegen:

BitmapImage demoBitmapImage = new BitmapImage();

demoBitmapImage.BeginInit();

demoBitmapImage.UriSource = new Uri("imagedemo.jpg", UriKind.RelativeOrAbsolute);

//demoBitmapImage.UriSource = new Uri(@"C:\Users\Kris\Pictures\imagedemo.jpg",

// UriKind.RelativeOrAbsolute);

demoBitmapImage.EndInit();

Image rightImage = new Image();

rightImage.Source = demoBitmapImage;

rightImage.Margin = new Thickness(120, 10, 0, 0);

rightImage.Width = 150;

rightImage.Height = 150;

paperCanvas.Children.Add(rightImage);

## Hoofdstuk 4: Variabelen en berekeningen

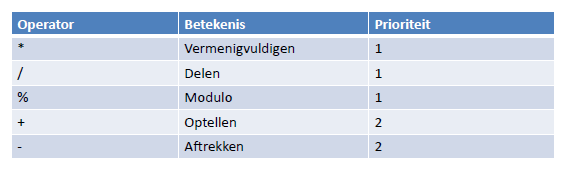
Belangrijke soorten variabelen:

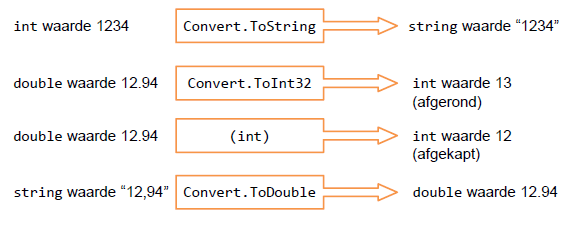
* Int: geheel getal van 4 bytes
* Double: kommagetal van 8 bytes
* Long: zeer lange getallen
* Float: komma getallen (Kleiner bereik dan double)
* Decimal: voor grote precisie, zoals financiele berekeningen
* String: voor teksten

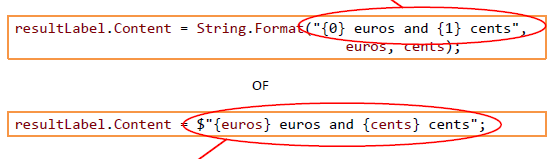
Regels voor declaratie van variabelen:

* Begin met een letter of \_
* Geen spaties, symbolen of keywords
* Max 255 tekens
* camelCasing

Soorten belangrijke operatoren:



Conversiemogelijkheden:

Formatteren gaat op 2 manieren:

## Hoofdstuk 5: Methoden en argumenten

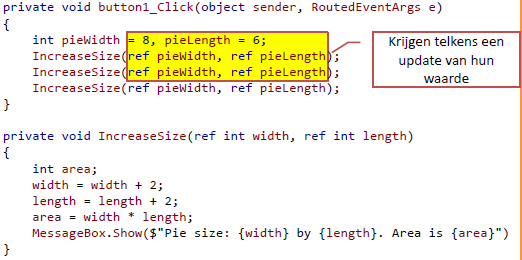
Lokale variabelen:

* Hulpmiddel voor berekeningen
* In de methode body gedeclareerd
* Ze bestaan enkel binnen de methode (lokaal of in de scope)
* Enkel zichtbaar binnen de methode body
* 2 methodes met lokale variabelen met dezelfde naam zijn toegestaan

Het verschil tussen out en ref

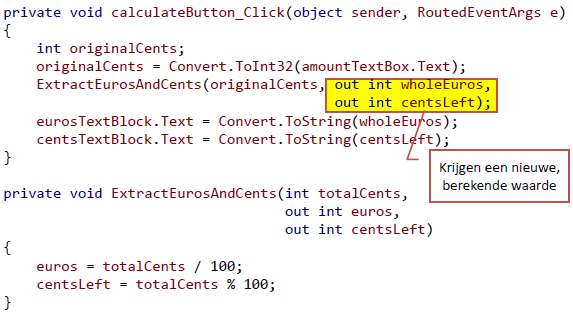
REF

* Refereert naar een bestaand argument met een waarde
* De methode verandert deze waarde
* Tweerichtingsverkeer



OUT

* Refereerd naar een bestaand of nietbestaand argument zonder waarde
* De methode geeft een initiele waarde
* Eenrichtingsverkeer





## Hoofdstuk 6: Objecten

Using 🡪 een klasse van een bepaalde namespace gebruiken (bv using vanboven het programma)

Namespace 🡪 hetgeen achter de using bv string of windows.forms etc

Delen van een klasse:

* Members
  + Voeren een methoden uit
  + Properties geven de toestand weer van de members
* Methodes en properties op objectniveau
  + Niet “de klasse label heeft een hoogte van...”
  + Maar “een label instantie label1 heeft een hoogte van... “
* Properties die je instelt of ophaalt

Klasse random:

Genereert een random variabelen beginnend met 5 tot maar niet met 110

Declaratie:

private Random ageGuesser = new Random();

int value = ageGuesser.Next(5, 110);

Klasse DispatcherTimer

Declaratie (met constructor):

private DispatcherTimer timer = new DispatcherTimer();

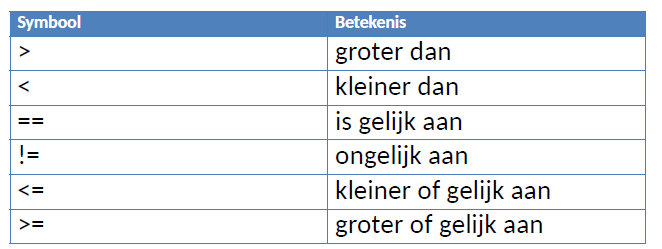
Namespace: System.Windows.Threading

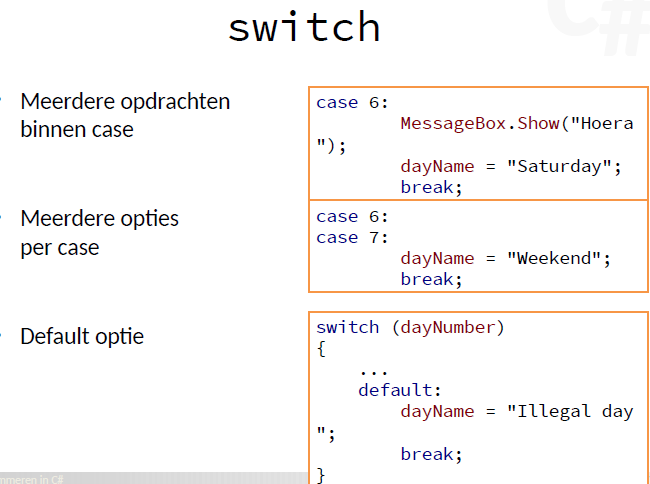
## Hoofdstuk 7: Beslissingen

Er zijn 2 manieren om beslissingen te maken:

* If statement
* Switch statement

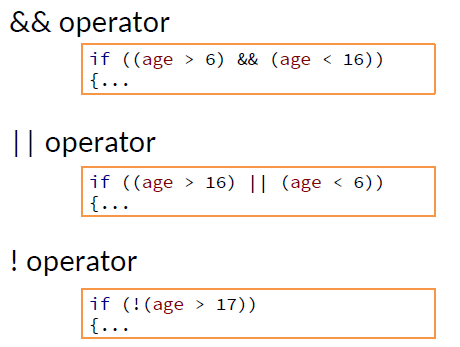
Operatoren:



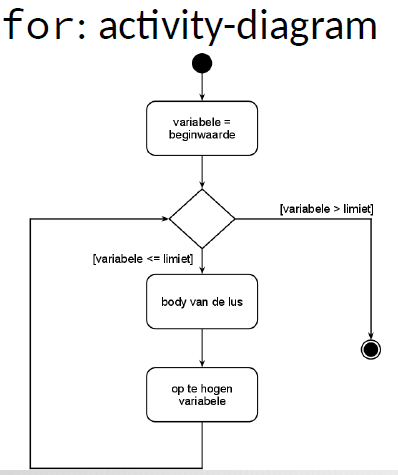


Ternaire operator:

*Resultaatvariabele* = *voorwaarde* ? *eerste\_expressie* : *tweede\_expressie*;



## Hoofdstuk 8: Herhalingen

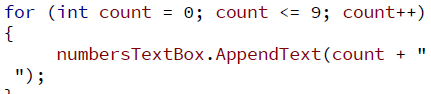
Soorten herhalingen:

* For loop
* While loop
* While do loop

**For loop:**

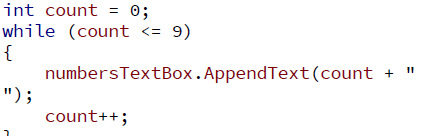
De body van de lus wordt gerhaaldelijk uitgevoerd

**Belangrijk:**

* De lusvariabelen mag niet veranderen in de body zelf
* Gebruik dan de while loop

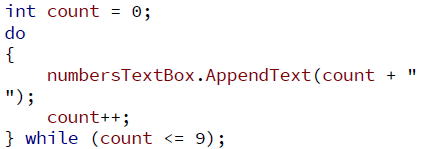
While loop:

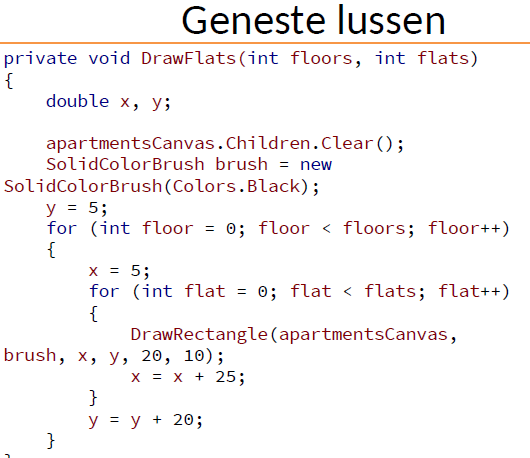
Deze gebruik je voor herhalingen waarbij de lusvoorwaarde vrijer te bepalen is. Het aantal iteraties is niet op voorhand bepaald

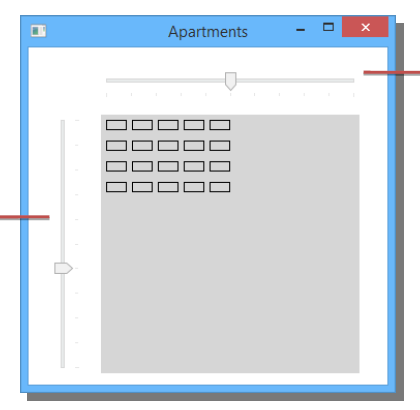


Do ... while loop

Atlernatieve herhaling. Op het einde van de herhaling wordt er pas getest en elke herhaling gebeurd minstens 1 keer.



Geneste lus



## Hoofdstuk 9: Debuggen

Wat is een bug?

Een fout in een programma.

Deze kan je opsporen door bijvoorbeeld een message.show() te gebruiken

Maar een debugger of het lezen van de broncoden door iemand anders kan ook helpen.

Ze ontstaan door syntacfouten. Deze zijn het makkelijkst op te sporen want de copiler geeft het aan.

generatedNumber = random?.Next(5,100);

?. = Save Navigation-operator = Save Navigation

Als *random == null*  *generatedNumber* wordt *null*

private int? generatedNumber;

## Hoofdstuk 10: Klassen schrijven

Indeling van een klasse:

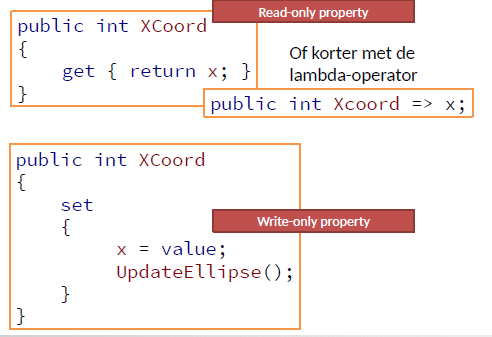
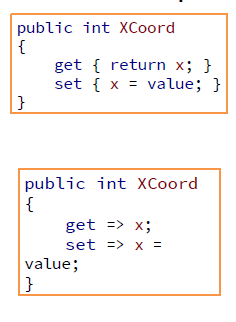
* Public methoden (aanspreekbaar door extern)
* Properties (toegang tot de toestand van het klasse object)
* Constructor methoden (zelfde naam als klassen)
* Private methoden (niet aanspreekbaar door extern)
* Private member variabelen (toestand van klasse object niet aanspreekbaar)

Klassediagram betekenissen:

* - = private
* + = public
* # = protected

Encapsulation?

Informatie verbergen. Elk obejct heeft een toestand die verborgen blijft voor de buitenwereld, en vertoont een bepaald gedrag dat deze toestand beinvloed. Enkel de buitenwereld kan het object manipuleren via public methoden en properties. Je gebruikt best altijd private properties en public methoden om deze te manipuleren.

Private properties definieren in getters en setters

Wat is een constructor?

* Speciale methode om een object te creeren deze heeft dezelfde naam als de klasse
* Met of zonder parameters
* Als je er geen maakt is er een basis constructor
* Als je er zelf schrijft vervalt deze basis ‘default’ constructor.
* Ctor tab tab

Overloading:

Een methode met dezelfde naam maar andere parameters (kan ook bij een constructor zijn)

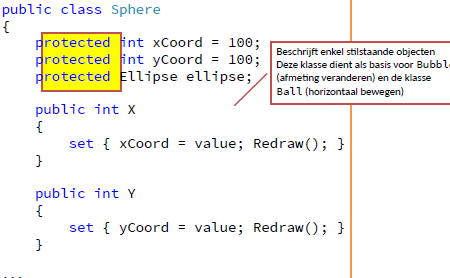
Static klassen en methoden:

Als er geen objecten van de klasse gemaakt moeten worden.

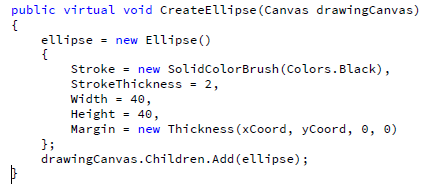
## Hoofdstuk 11: Overerving

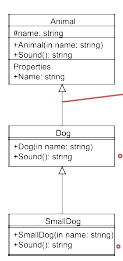
Wat is overerving?

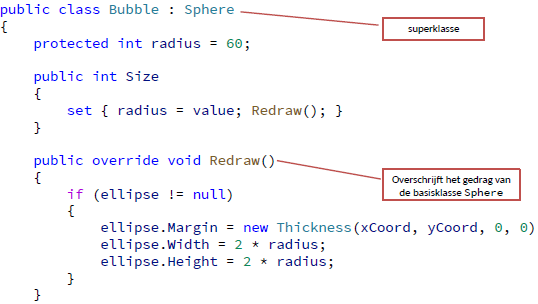
Als een klasse overerft van een andere klasse betekend dit dat de klasse alle items van zijn superklasse over erft, berhalve de constructoren. Deze heeft ook toegang tot ALLE items. Zowel private als publlic.

Protected variabelen:

Enkel toegankelijk voor de subklasse, dus de properties van de superklasse staan op protected. Het doel hierval is het beschikbaar stellen van deze properties.

Superklasse:



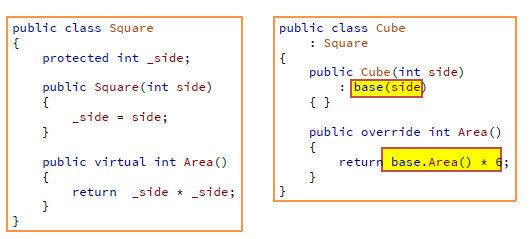
Subklasse

Van welke superklasse erft elk object over?

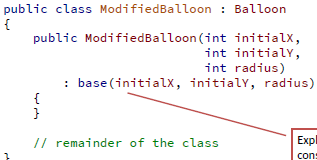
System.Object

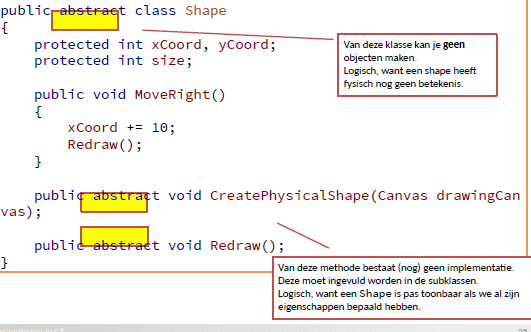
Wat is “Base”?:

Een referentie naar het onmiddelijke supergedeelte van het object. Het wordt gebruikt om een methode, proprtie, etc op te roepen van de superklasse. Wordt veel gebruikt als de methode hetzelfde doet als die van de superklasse en net iets extra erbij.



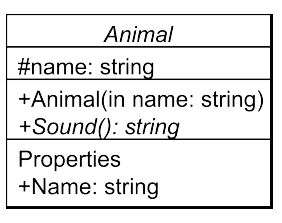
Hoe de constructor overgeerfdt kan worden:



Wat zijn abstracte klassen?

Base klassen waar je geen instanties van kan maken. Dit zijn klassen waar alle gemeenschappelijke kenmerken in staan voor ALLE subklassen. De methoden en properties worden zonder implementatie opgenomen, en dan ingevuld in de subklassen.

UML van een abstracte klasse:



Overerving niet mogelijk maken?

Dan maak je een klasse sealed. Dit kan je doen als veiligheidsoverweging.

public sealed class SmallDog : Dog

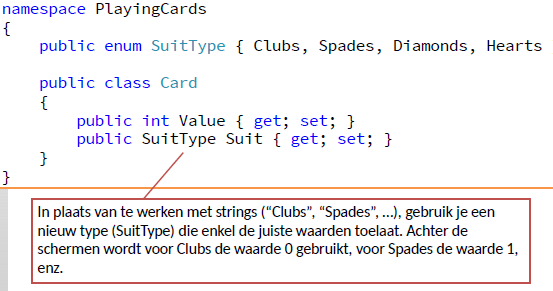
{

...

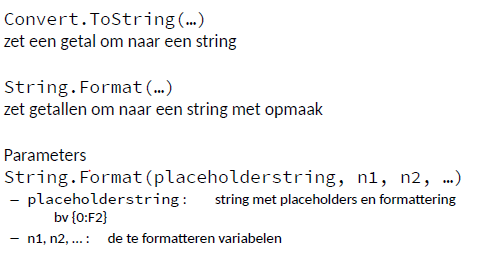
}

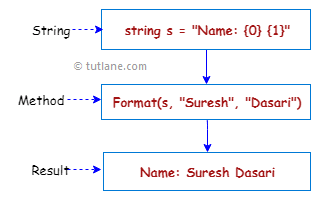
Wat is een enum?

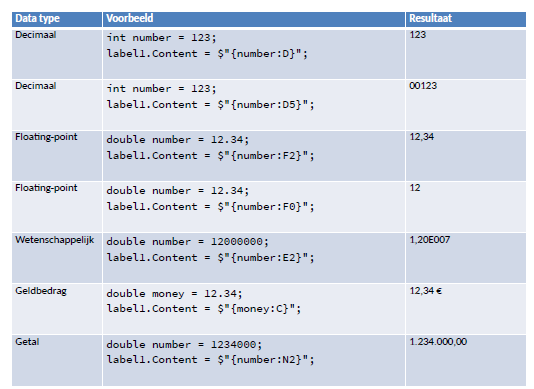
Een type dat een opsomming voorsteld, bijvoorbeeld de dagen van de week, maanden van het jaar, etc. De eerste waarde van de enum heeft altijd 0, buiten als je dit aanpast.

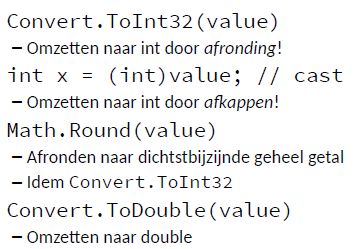


## Hoofdstuk 12: Bewerkingen

Literals? Getallen met een letterlijke waarde die je rechtstreeks in een programma schrijft.





Wat zijn constanten?

Onveranderlijke waarden. Om de leesbaarheid te verhogen.

const double InchesToCm = 2.54;

double d1 = 4.2;

double d2 = 4.8;

string line = $"afgerond: {Math.Round(d1)} en {Convert.ToInt32(d2)}";

MessageBox.Show(line);

Math.Abs(value) geeft een absolute waarde.

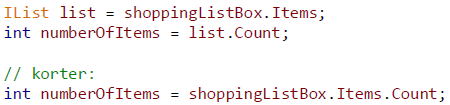
## Hoofdstuk 13: Gegevensstructuren: Lijsten en Listbox

Listbox:

* Vind je in de toolbox
* Toont een lijst
* Elementen kunnen toegevoegd worden

IList:

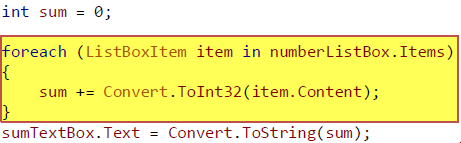
* De datastructuur achter de schermen van de listbox
* Interface
* Kan onafhankelijk van de listbox gebruikt worden.



Namespace: using System.Collections.ICollection

Methoden IList:

* Listbox.Items.Add(newItem) 🡨 toevoegen van listbox item
* Listbox.Items.Count 🡨 tellen van het aantal elementen in de list
* Listbox.SelectedIndex 🡨 geeft je de index van het aangeduide item in de lijst (vanaf 0)
* (ListBoxItem)shoppingListBox.Items[index] 🡨 met een cast de inhoud van een item index
* Listbox.Items.RemoveAt(index) 🡨 verwijdert item op aangegeven index
* Listbox.Items.Clear() 🡨 verwijdert alle gegevens in de list
* Listbox.Items.Insert(index, item) 🡨 voegt nieuw item toe op gegeven index



Een list item maken zonder listbox te gebruiken:

private IList<string> list = new List<string>();

Databinding: Bijvoorbeeld alle objecten van een list naar een listbox kopieeren:

shoppingListBox.ItemsSource = list;

## Hoofdstuk 14: Arrays

Een array is een object, dat anderen objecten groepeerd. Het is belangrijk dat je de dimensies altijd in constanten definieerd. Zo vermijd je bugs als je deze later wil wijzigen.



Zoeksleutel (bv index van array) zonder array gebruiken: System.Collections.Generic.Dictionary

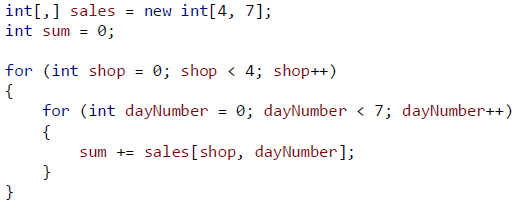
??????????????????????????????????????????????????????????????????????????????????

## Hoofdstuk 15: 2dimentionale Arrays

## 

Waarde toekennen aan deze plaatsen:

sales[2, 3] = Convert.ToInt32(salesTextBox.Text);

chessboard[3, 4] = fieldTextBox.Text;

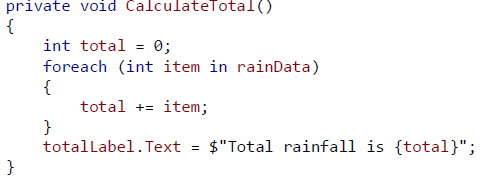
Grotes van 2dimentionale arrays berekenen:

* Array.GetLength(0) 🡪 geeft de lengte voor de eerste dimentie (rijen)
* Array.GetLength(1) 🡪 geeft de lengte voor de tweede dimentie (kolommen)
* Array.Length 🡪 Max aantal elementen in de 2dim array

Declaratie van een 2dim array

int[,] table = {{1, 0, 1},

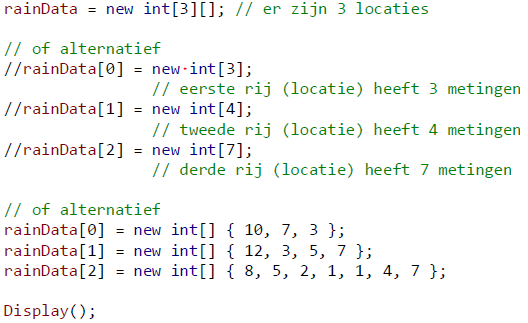
{0, 1, 0}};

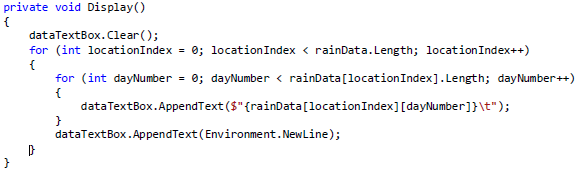
Foreach:

Jagged array?

Indien je niet verwacht om alle vakken op te vullen gebruik je best deze vorm







## Hoofdstuk 22: Interfaces

## Hoofdstuk 23: Polymorfie