



# Programmeren 2

# Lesmateriaal

Moodle-course:

- "2021 Inf1.2S Programmeren"

Boek:

- *geen fysiek boek (op Moodle staan verwijzingen)*

Opdrachten:

- wekelijks (6x), individueel, verplicht
- inleveren op Moodle (CodeGrade), deadline volgende week
- alle AutoTests en handmatige controle moeten 10/10 zijn

# Programma periode 1.2

01 (wk-46)	Enumeraties / structures / classes
02 (wk-47)	2-dim arrays / Flow Control
03 (wk-48)	Lists / Dictionaries
04 (wk-49)	File I/O / error handling
05 (wk-50)	opbouw / structuur
06 (wk-51)	opbouw / structuur
07 (wk-52)	kerstvakantie
08 (wk-53)	kerstvakantie
<hr/>	
09 (wk-01)	herhaling/vragen/oefententamen
10 (wk-02)	<i>tentamens</i>
11 (wk-03)	<i>herkansingen</i>
12 (wk-04)	<i>herkansingen</i>

# Regels omtrent methoden

- Deze periode (bij Programmeren 2) gaan we veel werken met methoden, omdat de programma's groter worden en we het overzichtelijk willen houden...
- Regels omtrent methoden:
  - 1) geef elke methode een betekenisvolle naam
  - 2) één methode, één taak! (*dus bv niet zowel berekenen als tonen*)
  - 3) houdt een maximum van **30 code-regels** aan

**static methoden...**

# Allereerst... weg met 'static' methoden!

```
static void Main(string[] args)
{
    Console.Write("Enter a year: ");
    int year = int.Parse(Console.ReadLine());

    if (IsLeapYear(year))
        Console.WriteLine($"{year} is a leap year.");
    else
        Console.WriteLine($"{year} is not a leap year.");

    Console.ReadKey();
}

static bool IsLeapYear(int year)
{
    if (year < 0) return false;

    bool deelDoor400 = ((year % 400) == 0);
    bool deelDoor100 = ((year % 100) == 0);
    bool deelDoor4 = ((year % 4) == 0);

    return (deelDoor400 || (deelDoor4 && !deelDoor100));
}
```

We willen af van de static!

# Voortaan ziet de main er zo uit...

```
class Program
{
    static void Main(string[] args)
    {
        Program myProgram = new Program();
        myProgram.Start();
    }

    ☆ void Start()
    {
        int year = 1900;
        if (IsLeapYear(year))
            Console.WriteLine($"{year} is a leap year.");
        else
            Console.WriteLine($"{year} is not a leap year.");
    }

    ☆ bool IsLeapYear(int year)
    {
        if (year < 0) return false;

        bool deelDoor400 = ((year % 400) == 0);
        bool deelDoor100 = ((year % 100) == 0);
        bool deelDoor4 = ((year % 4) == 0);

        return (deelDoor400 || (deelDoor4 && !deelDoor100));
    }
}
```

De Main-methode maakt een (Program) object aan (via new) en roept als enige de Start-methode aan.

☆ De andere methoden (hier 'Start' en 'IsLeapYear') hoeven nu niet meer static te zijn!

# Enumerations



# Leesbaar en duidelijk?

Bekijk de volgende methode:

```
public bool IsWeekend(int dayOfWeek)
{
    return ((dayOfWeek == 0) || (dayOfWeek == 6));
}
```

Aanroep:

```
// onmogelijk, maar de compiler gaat niet klagen!!
bool result = IsWeekend(100);

// geldige invoer, maar over welke dag hebben wij het hier?
result = IsWeekend(1);
```

# Enumeraties

- Een verzameling van (bij elkaar horende) constanten, bijvoorbeeld:
  - Geslacht: Man, Vrouw
  - StudentType: VoltijdStudent, DeeltijdStudent
- Enumeraties zijn "strongly typed constants"  
(het ene type kan niet zomaar omgezet worden naar een ander type; dit voorkomt programmeerfouten!)

# Enumeraties (in aparte files)

```
public enum WeekDays {
    Sunday, Monday, Tuesday, Wednesday, Thursday, Friday, Saturday
}

class Program
{
    static void Main(string[] args)
    {
        bool result1 = IsWeekend(WeekDays.Monday); // geldige invoer
        bool result2 = IsWeekend(1);             // verkeerde invoer
    }

    public static bool IsWeekend(WeekDays day)
    {
        return ((day == WeekDays.Sunday) || (day == WeekDays.Saturday));
    }
}
```

*(beter leesbaar EN compiler waarschuwt bij verkeerd gebruik)*

# Enumeraties

De “velden” van een enumeration hebben een integer waarde, beginnend bij 0.

Dit kan je veranderen door bijv.

```
public enum WeekDays {  
    Sunday, Monday, Tuesday, Wednesday=8, Thursday, Friday, Saturday  
}
```

Nu heeft Sunday nog steeds de waarde 0, maar Wednesday=8, Thursday=9, etc.

## Voorbeeld code

```
public enum Days {
    Zondag, Maandag, Dinsdag, Woensdag, Donderdag, Vrijdag, Zaterdag
}

class Program
{
    static void Main(string[] args)
    {
        // doorloop alle dagen
        for (Days d = Days.Zondag; d <= Days.Zaterdag; d++) {
            Console.WriteLine(d);
        }

        // lees dagnummer (bv "1")
        string s = Console.ReadLine();
        Days n = (Days)int.Parse(s);

        switch (n)
        {
            case Days.Vrijdag:
                Console.WriteLine("Pffff, bijna weekend");
                break;
            case Days.Maandag:
                Console.WriteLine("Zucht, weer naar het werk .....");
                break;
        }
    }
}
```

*Uitvoer:*

Zondag

Maandag

Dinsdag

Woensdag

Donderdag

Vrijdag

Zaterdag

1 (invoer van gebruiker)

Zucht, weer naar het werk...

# Enumeraties – andere voorbeelden

LampStatus { Aan, Uit }

StoplichtStatus { Rood, Oranje, Groen, Storing }

AutomaatStatus { InBedrijf, Uitverkocht, Defect, ... }

PolitiekePartij { D66, PvdA, CDA, VVD, GroenLinks, ... }

Orientation { Portrait, Landscape }

...

# Enumeraties - tips

- Maak een methode voor het inlezen van een enumeratie (inclusief controle op foutieve invoer)

Die kan je dan bv zo gebruiken:

```
Weekdag dag = LeesWeekdag("kies een dag: ");
```

- Maak ook een methode voor het tonen van een enumeratie (beeldscherm)

Die kan je dan bv zo gebruiken:

```
ToonWeekdag(dag);
```

# Structures



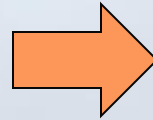
# Structures

- Groeperen van data-elementen die bij elkaar horen...

3 afzonderlijke array's

studentnummer	naam	geb.datum
studentnummer	naam	geb.datum
studentnummer	naam	geb.datum
studentnummer	naam	geb.datum
studentnummer	naam	geb.datum
studentnummer	naam	geb.datum

3 array's



6 structures

studentnummer	naam	geb.datum
studentnummer	naam	geb.datum
studentnummer	naam	geb.datum
studentnummer	naam	geb.datum
studentnummer	naam	geb.datum
studentnummer	naam	geb.datum
studentnummer	naam	geb.datum

1 array met struct's

# Tot nu toe... (met array's)

```
Console.Write("Geef het aantal studenten: ");
int aantalStudenten = Int32.Parse(Console.ReadLine());

// maak een array aan voor alle student namen en cijfers
string[] studentNamen = new string[aantalStudenten];
float[] studentCijfers = new float[aantalStudenten];

// vraag naar de namen
for (int i = 0; i < aantalStudenten; i++)
{
    Console.Write("Geef de naam van de " + (i + 1) + "e student: ");
    studentNamen[i] = Console.ReadLine();
}

// vraag naar de cijfers
for (int i = 0; i < aantalStudenten; i++)
{
    Console.Write("Geef het cijfer voor " + studentNamen[i] + ": ");
    studentCijfers[i] = float.Parse(Console.ReadLine());
}

// geef totaal overzicht
PrintStudenten(studentNamen, studentCijfers);
```

De namen en cijfers van de studenten staan in aparte array's. We kunnen dit niet kwijt in 1 array.

Wat als een student ook een nummer, geboortedatum en woonplaats moet hebben? array's, array's, array's...

Als we de studenten willen printen via een methode, dan moeten we alle array's meegeven.

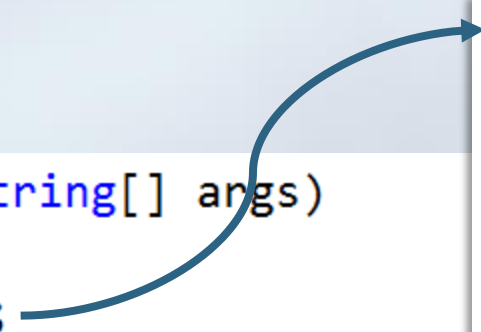
# Struct (elk in een aparte file)

- Met een 'struct' kunnen we alle informatie van een student (naam, cijfer, ...) bij elkaar laten horen

```
static void Main(string[] args)
{
    Student student1;

    student1.Naam = "Peter van Vliet";
    student1.Nummer = "13902";
    student1.GeboorteDatum = new DateTime(1971, 6, 24);
    student1.Cijfers = new float[10];
    student1.Woonplaats = "Heiloo";

    Console.WriteLine("Naam: " + student1.Naam);
}
```



```
struct Student
{
    public string Naam;
    public string Nummer;
    public DateTime GeboorteDatum;
    public float[] Cijfers;
    public string Woonplaats;
}
```

# Structs - tips

- Maak een methode voor het inlezen een struct (inclusief controle op foutieve invoer)

Die kan je dan bv zo gebruiken:

```
Student user = LeesStudent("Voer je gegevens in: ");
```

- Maak ook een methode voor het tonen van een struct (beeldscherm)

Die kan je dan bv zo gebruiken:

```
ToonStudent(user);
```

# Classes

# struct vs class

- Alles wat met een struct kan (*samenvoegen van meerdere samenhangende variabelen*) kan ook met een class ...

```
struct Student
{
    public string Naam;
    public int Leeftijd;
}
```

```
class Student
{
    public string Naam;
    public int Leeftijd;
}
```

```
class Program
{
    static void Main(string[] args)
    {
        Program myProgram = new Program();
        myProgram.Start();
    }

    void Start()
    {
        Student student;
        student = new Student();
        student.Naam = "Kees";
        student.Leeftijd = 19;

        Console.WriteLine("naam: {0}", student.Naam);
        Console.WriteLine("leeftijd: {0}", student.Leeftijd);
    }
}
```

Als Student een class is, dan is new vereist!

# struct vs class

- ... maar een class wordt normaliter gebruikt (zeker bij object-georiënteerde applicaties)
- Belangrijk verschil is dat een struct een 'value type' is en een class is een 'reference type'
  - een value type wordt op de stack opgeslagen, en een reference type op de heap
  - als een value type gekopieerd wordt, dan worden deze in z'n geheel gekopieerd; als een reference type gekopieerd wordt dan wordt alleen de reference (pointer) gekopieerd!

# struct vs class

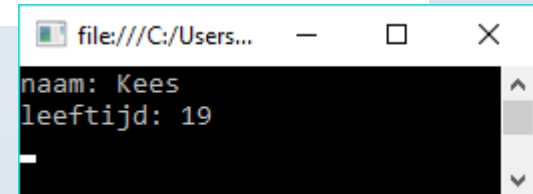
```
class Program
{
    static void Main(string[] args)
    {
        Program myProgram = new Program();
        myProgram.Start();
    }

    void Start()
    {
        Student student1;
        student1 = new Student();
        student1.Naam = "Kees";
        student1.Leeftijd = 19;

        Student student2;
        student2 = student1;    // kopieer student 1
        student2.Leeftijd = 20;

        Console.WriteLine("naam: {0}", student1.Naam);
        Console.WriteLine("leeftijd: {0}", student1.Leeftijd);
    }
}
```

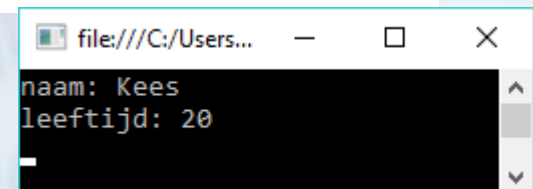
```
struct Student
{
    public string Naam;
    public int Leeftijd;
}
```



file:///C:/Users...  
naam: Kees  
leeftijd: 19

student1 is ongewijzigd!

```
class Student
{
    public string Naam;
    public int Leeftijd;
}
```



file:///C:/Users...  
naam: Kees  
leeftijd: 20

student1 is gewijzigd!



# Huiswerk

- Bestudeer de aangegeven paragrafen uit het 'Yellow Book' (zie Moodle)
- Week 1 opdrachten (zie Moodle)