



# Programmeren 2

# Programma periode 1.2

01 (wk-46)	Enumeraties / structures / classes
02 (wk-47)	2-dim arrays / Flow Control
03 (wk-48)	<b>Lists / Dictionaries</b>
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>

# ArrayList / List

# Arrays

- Tot nu toe hebben we voor het opslaan van meerdere items een 'array' gebruikt

```
int[] numbers = new int[10];
```

*max. 10 int-waarden*

```
float[] grades = new float[5];
```

*max. 5 float-waarden*

```
bool[] primeNumbers = new bool[100];
```

*max. 100 bool-waarden*

# Nadeel van arrays

- Het nadeel is echter dat...?
- het aantal elementen/items van te voren bekend moet zijn... *(hetgeen niet altijd het geval is)*

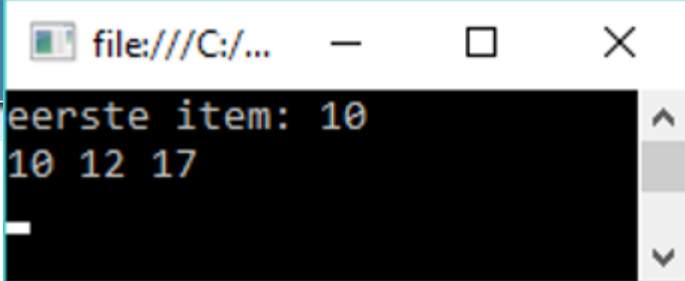
```
int[] numbers = new int[10];
```

- We zouden een 'veilige' grootte kunnen gebruiken (bv 10.000.000?)  
*(kost veel geheugen, vaak onnodig...)*

```
int[] numbers = new int[10000000];
```

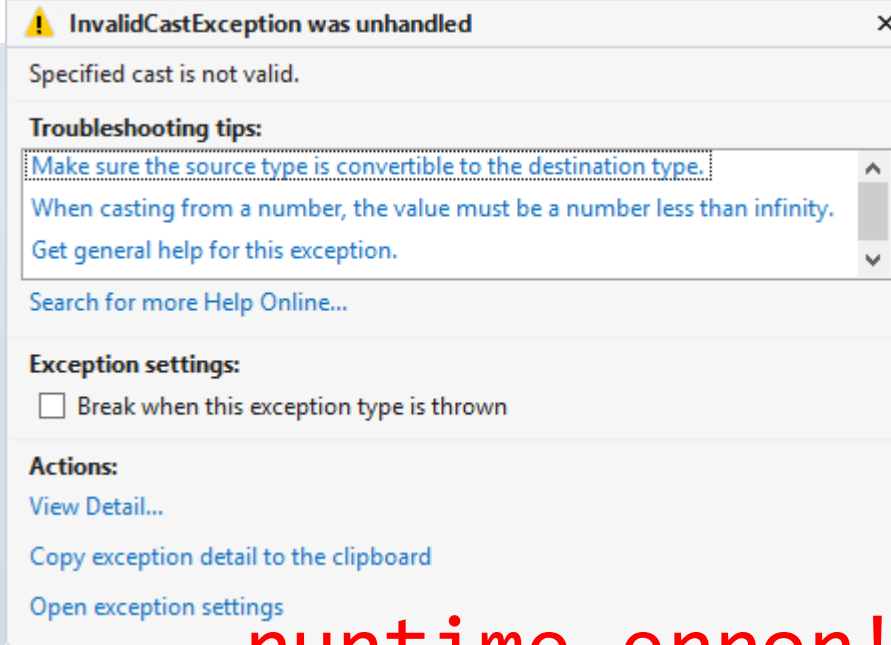
# ArrayList

```
ArrayList numbers = new ArrayList();  
numbers.Add(10);  
numbers.Add(12);  
numbers.Add(17);  
Console.WriteLine("eerste item: " + numbers[0]);  
foreach (int number in numbers)  
    Console.Write("{0} ", number);  
Console.WriteLine();
```



```
file:///C:/...  
eerste item: 10  
10 12 17
```

```
ArrayList numbers = new ArrayList();  
numbers.Add(10);  
numbers.Add(12);  
numbers.Add(17);  
numbers.Add("dertig");  
foreach (int number in numbers)  
    Console.Write("{0} ", number);  
Console.WriteLine();
```



**InvalidCastException was unhandled**

Specified cast is not valid.

**Troubleshooting tips:**

- [Make sure the source type is convertible to the destination type.](#)
- [When casting from a number, the value must be a number less than infinity.](#)
- [Get general help for this exception.](#)

[Search for more Help Online...](#)

**Exception settings:**

☐ Break when this exception type is thrown

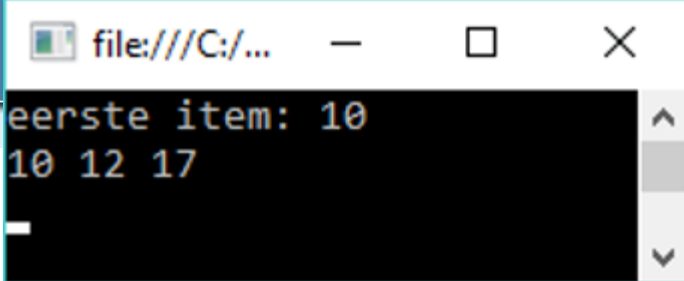
**Actions:**

- [View Detail...](#)
- [Copy exception detail to the clipboard](#)
- [Open exception settings](#)

runtime error!

# List – generics (typesafe)

```
List<int> numbers = new List<int>();  
numbers.Add(10);  
numbers.Add(12);  
numbers.Add(17);  
Console.WriteLine("eerste item: " + numbers[0]);  
foreach (int number in numbers)  
    Console.Write("{0} ", number);  
Console.WriteLine();
```



A screenshot of a Windows console window. The title bar shows the file path 'file:///C:/...' and standard window controls. The console output is as follows:  
eerste item: 10  
10 12 17  
The text is displayed in a monospaced font on a black background.

```
List<int> numbers = new List<int>();  
numbers.Add(10);  
numbers.Add(12);  
numbers.Add(17);  
numbers.Add("dertig");
```

class System.String  
Represents text as a series of Unicode characters.

Error:

The best overloaded method match for 'System.Collections.Generic.List<int>.Add(int)' has some invalid arguments

compile error!

# List – generics (typesafe)

```
List<int> getallen = new List<int>();
```

```
List<float> cijfers = new List<float>();
```

```
List<double> afmetingen = new List<double>();
```

```
List<bool> priemgetallen = new List<bool>();
```

```
List<Schaakstuk> schaakstukken = new List<Schaakstuk>();
```

```
List<Persoon> personen = new List<Persoon>();
```



# Oefening

- Schrijf een methode 'LeesLetter' die een letter inleest en deze retourneert. De methode geeft alleen een letter terug die nog niet eerder ingelezen was. Deze methode kan bv gebruikt worden in een spel 'galgje'.

# Oefening

```
char LeesLetter(List<char> blacklist)
do
    line = ReadLine()
    letter = line[0]
while blacklist.Contains(letter)
return letter
```

```
char LeesLetter(List<char> whitelist)
do
    line = ReadLine()
    letter = line[0]
while !whitelist.Contains(letter)
return letter
```

# Dictionary

# Lists vs Dictionaries

- Zoals we gezien hebben, gebruiken we een **List** als een items willen bijhouden, maar we van te voren niet weten om hoeveel items het gaat
- Als we naast het bijhouden van items, ook specifieke items (snel) willen opzoeken op basis van een sleutel (zoals bv een naam of isbn van een persoon), dan gebruiken we een **Dictionary**
- Een entry in een dictionary bevat een sleutel en een bijhorende waarde  
*(denk maar een woordenboek: de sleutel is het woord, de waarde is de betekenis van het woord)*

# Dictionary voorbeeld

- Een dictionary wordt aangemaakt met 2 types:
  - een datatype voor de sleutels
  - een datatype voor de waarden
- Een voorbeeld: we willen een lijst van studentnamen bijhouden, en op basis van de studentnummer de bijbehorende naam opzoeken

```
// student number => name of student  
Dictionary<int, string> students = new Dictionary<int, string>();
```

nummer	: int
naam	: string

# Dictionary – toevoegen

- Bij het toevoegen moet steeds een studentnummer en de bijbehorende naam opgegeven worden

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

void Start()
{
    // student number => name of student
    Dictionary<int, string> students = new Dictionary<int, string>();

    students.Add(565446, "David Beckham");
    students.Add(556324, "Lionel Messi");
    students.Add(544569, "Cristiano Ronaldo");
    students.Add(598341, "Diego Maradona");
}
```

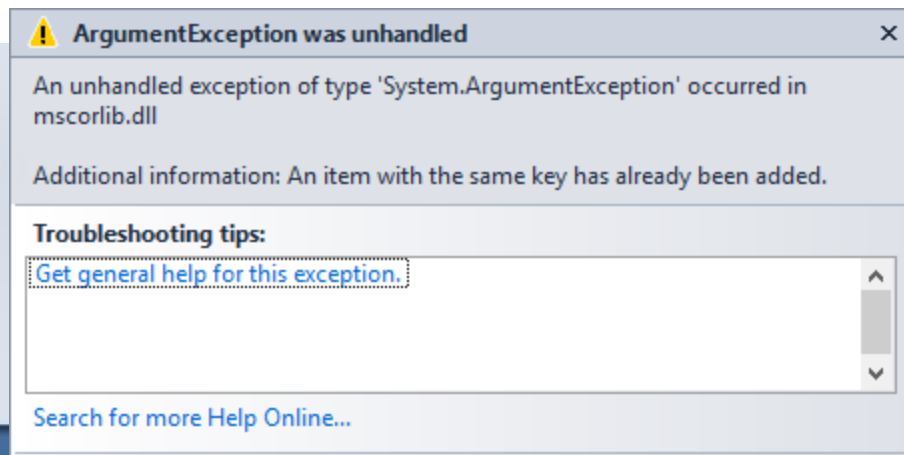
# Dictionary – geen dubbelen toegestaan

- Er kunnen geen entries bestaan met dezelfde sleutel

```
void Start()
{
    // student number => name of student
    Dictionary<int, string> students = new Dictionary<int, string>();

    students.Add(565446, "David Beckham");
    students.Add(556324, "Lionel Messi");
    students.Add(544569, "Cristiano Ronaldo");
    students.Add(598341, "Diego Maradona");
    students.Add(598341, "Neymar");    // key already exists...
}
```

*Dit studentnummer  
komt al voor in de  
dictionary!*



# Dictionary – test op aanwezigheid

```
void Start()
{
    // student number => name of student
    Dictionary<int, string> students = new Dictionary<int, string>();

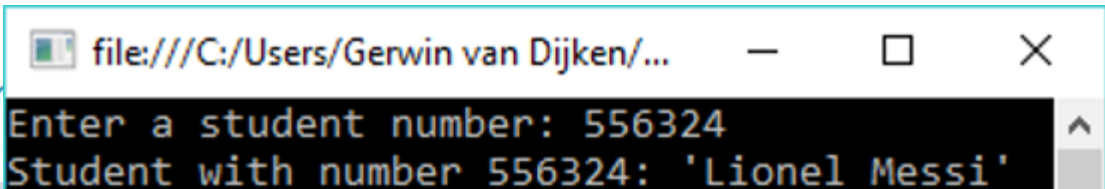
    // fill dictionary ...

    // read student number
    Console.WriteLine("Enter a student number: ");
    int number = int.Parse(Console.ReadLine());
    if (!students.ContainsKey(number))
    {
        Console.WriteLine($"There's no student with number {number}");
    }
    else
    {
        Console.WriteLine($"Student with number {number}: '{students[number]}'");
    }
}

Cor
```

Eerst testen op aanwezigheid...

... alvorens op te vragen!



file:///C:/Users/Gerwin van Dijken/...

Enter a student number: 556324  
Student with number 556324: 'Lionel Messi'



# Oefening

- Stel we hebben een lijst van teams (naam, spelers, ...), en we willen het team vinden op basis van een naam

```
List<Team> teams = new List<Team>()  
// vul teams hier...  
  
lees teamNaam  
team = null  
i = 0  
while i < teams.Length and team == null  
    if teams[i].naam == teamNaam  
        team = teams[i]  
    else  
        i++  
  
// ...
```

# Oefening

- Herschrijf de (pseudo)code van de vorig slide zodat geen List maar een Dictionary wordt gebruikt

```
// maak Dictionary aan  
Dictionary<string, Team> teams = new Dictionary<string, Team>()  
// vul teams hier...
```

```
lees teamNaam
```

```
team = null  
if (teams.ContainsKey(teamNaam))  
    team = teams[teamNaam]  
else  
    toon "team bestaat niet!"  
// ...
```

# Huiswerk

- Bestudeer de aangegeven paragrafen uit het 'Yellow Book' (zie Moodle)
- Week 3 opdrachten (zie Moodle)