Programming 2

Program term 1.2

01 (wk-46)	enumerations / 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)	program structure
06 (wk-51)	program structure
07 (wk-52)	Christmas holiday
08 (wk-53)	Christmas holiday

---

09 (wk-01)	practice exam
10 (wk-02)	*exams*
11 (wk-03)	*retake exams*
12 (wk-04)	*retake exams*

ArrayList / List

Arrays

- So far we have used arrays for storing multiple items

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

max. 10 int-values

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

max. 5 float-values

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

max. 100 bool-values

Disadvantage of arrays

- Disadvantage of an array is…?

- the number of elements/items is fixed…
  (but sometimes we don't know how many elements)

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

- We could use a 'safe' number of elements (like 10.000.000?)
  (will consume a lot of memory, and most of it will not be used…)

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

ArrayList

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

```
first item: 10
10 12 17
```

```
ArrayList numbers = new ArrayList();
numbers.Add(10);
numbers.Add(12);
numbers.Add(17);
numbers.Add("thirty");
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...

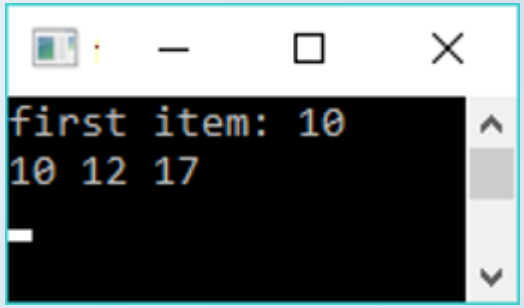
Copy exception detail to the clipboard

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("first item: " + numbers[0]);
foreach (int number in numbers)
    Console.Write("{0} ", number);
Console.WriteLine();
```




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

class System.String
Represents text as a series of Unicode characters.To browse the .NET Framework source code for this type, see the Reference Source.

Argument 1: cannot convert from 'string' to 'int'

compile error!

List – generics (typesafe)

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

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

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

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

```
List<ChessPiece> chessPieces = new List<ChessPiece>();
```

```
List<Person> persons = new List<Person>();
```

Exercise

- Write a method ‘ReadLetter’ that reads a letter and returns it. The method returns only a letter that was not read before.
  This method can be used in the game ‘hangman’.

Exercise

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


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

Dictionary

Lists vs Dictionaries

- A **List** is used when we want to collect items, but the number of items is unknown

- If we not only want to collect items but also want to search specific items <u>with a key</u> (like a name or a social security number of a person), then we will use a **Dictionary**

- An entry in a dictionary contains a key and a <u>corresponding</u> value
  (for example an English dictionary: the key is the word, the value is the meaning/description of the word)

Dictionary example

- A dictionary is created with 2 types:
  - a datatype for the keys
  - a datatype for the values

- An example: we want a list of student names, and find names according to a student number

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

number : int
name : string

Dictionary – add

- When adding an entry to a dictionary, we need to give a key (student number) and a value (corresponding name)

```
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");
}
```

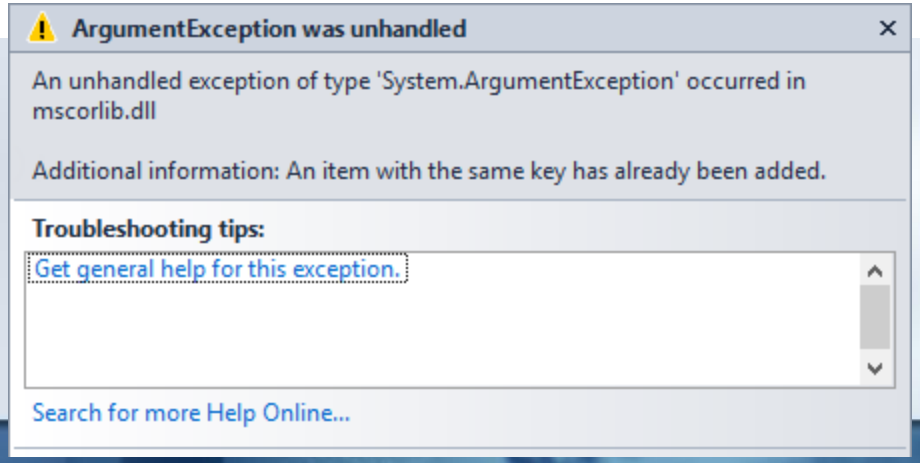
Dictionary – no duplicates allowed

- There can be no multiple entries with the same key

```
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...
}
```

this student number already exists in the dictionary!



Dictionary – test presence

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

    // fill dictionary ...

    // read student number
    Console.Write("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]}'");
    }

    Co
}
```

First test if key is present...

... before using the key!

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

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

Exercise

- Suppose we have a list of teams (name, players, ...), and we want to find the team with a certain name

```
List<Team> teams = new List<Team>()
// fil list of teams here...

read nameOfTeam
team = null
i = 0
while i < teams.Length and team == null
        if teams[i].name == nameOfTeam
                team = teams[i]
        else
                i++
// ...
```

Exercise

- Rewrite the (pseudo)code of the previous slide to replace the List with a Dictionary

```
// create Dictionary
Dictionary<string, Team> teams = new Dictionary<string, Team>()
// fil teams here...

read nameOfTeam

team = null
if (teams.ContainsKey(nameOfTeam))
       team = teams[nameOfTeam]
else
       display "team does not exist!"
// ...
```

Homework

- Read paragraphs ‘Yellow Book’
  (references can be found on Moodle)

- Assignments week 3
  (can be found on Moodle)