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*

File I/O

File I/O (Input/Output)

- Namespace System.IO ➔ `using System.IO;`

- Reading from file:
  1) open file
  2) read file
  3) close file

Writing to file:
1) open file
2) write file
3) close file

Reading from (text)file

```
void Start()
{
    DisplayFile("test.txt");
    Console.ReadKey();
}

void DisplayFile(string filename)
{
    Console.WriteLine("reading file...");

    // open file
    StreamReader reader = new StreamReader(filename);

    // display all lines (from file) on screen
    while (!reader.EndOfStream)
    {
        string s = reader.ReadLine();
        Console.WriteLine(s);
    }

    // close file
    reader.Close();
}
```

StreamReader if we want to read.
The file is opened here.

Read 1 line from file.

Don't forget to close the file.

Writing to (text)file

```
void WriteFile(string filename)
{
    Console.WriteLine("writing file...");

    // open file
    StreamWriter writer = new StreamWriter(filename);

    // write all lines (from user) to file
    string s = Console.ReadLine();
    while (s != "stop")
    {
        writer.WriteLine(s);

        // read next line
        s = Console.ReadLine();
    }

    // close file
    writer.Close();
}
```

StreamWriter if we want to write. The file is created here.

Write 1 line to file.

Location of files

- Path 'starts from' location of executable → \bin\Debug\

```
void Start()
{
    DisplayFile("..\\..\\test.txt");
    Console.ReadKey();
}

void DisplayFile(string filename)
{
    Console.WriteLine("reading file...");

    // open file
    StreamReader reader = new StreamReader(filename);

    // ...

    // close file
    reader.Close();
}
```

2 levels back to reach the file.

PraktijkProgrammeren2 ▸ week3. FileIO ▸ bin ▸ Debug

Name

- week3.FileIO.exe
- week3.FileIO.exe.config
- week3.FileIO.pdb
- week3.FileIO.vshost.exe
- week3.FileIO.vshost.exe.config
- week3.FileIO.vshost.exe.manifest

File 'test.txt' is located in 'week3.FileIO'.

File.Exists

- Checking if a file (already) exists

```
void ReadFile(string filename)
{
    Console.WriteLine("reading file...");

    // file exists?
    if (File.Exists(filename))
    {
        // open file
        StreamReader reader = new StreamReader(filename);

        // ...

        // close file
        reader.Close();
    }
}
```

Error handling

Error handling

- What can cause an error?
  - user input
  - a bug
  - hardware (disk full, network down, …)
  - …
- What do we want to prevent?
  - a crash
  - incorrect behavior (of the program)
  - …

Error detection

- How can we (the program) detect something went wrong?
  - a crash → we will know that!
  - incorrect behavior → ???
  - …

Some examples

#1) Converting text to int

```
Console.Write("Enter your age: ");
string input = Console.ReadLine();
int age = int.Parse(input);
```

file:///C:/Users/Gerwin va...

```
Enter your age: eighteen
```

FormatException was unhandled

Input string was not in a correct format.

Troubleshooting tips:

When converting a string to DateTime, parse the string to take the date before putting each variable into the DateTime object.

Make sure your method arguments are in the right format.

Get general help for this exception.

Search for more Help Online...

Exception settings:

☐ Break when this exception type is thrown

Actions:

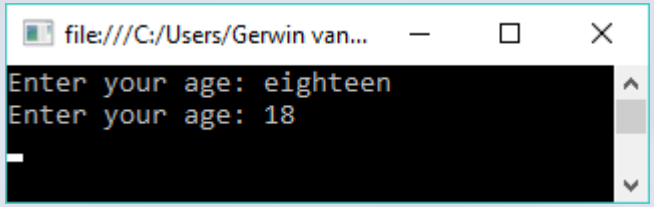
View Detail...

Copy exception detail to the clipboard

Open exception settings

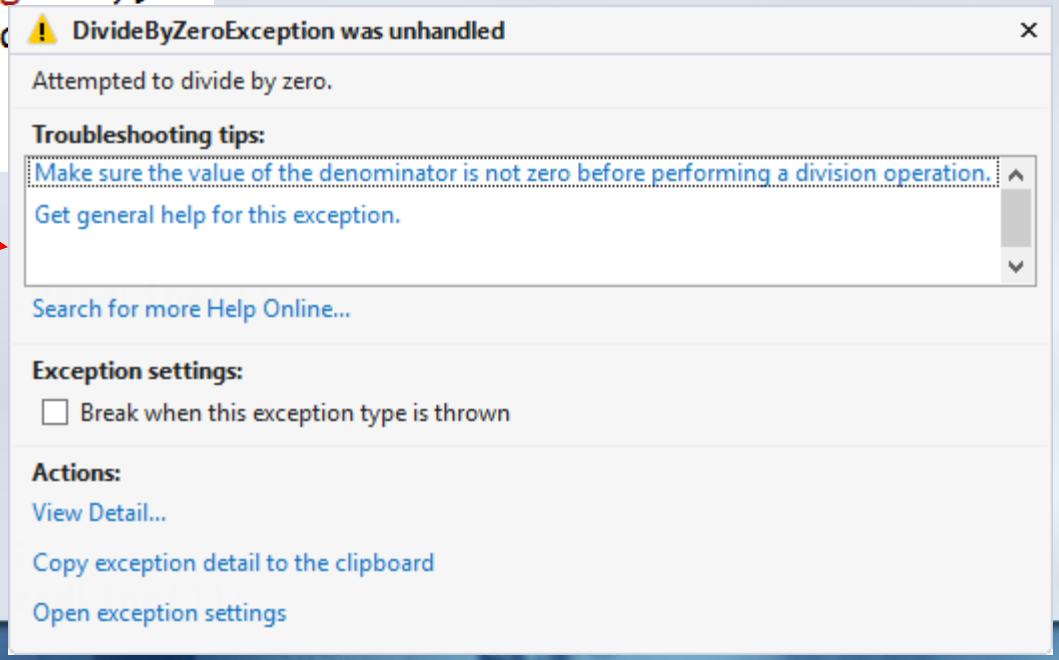
#1) Converting text to int (prevent)

```
int age;
bool correctInput = false;
while (!correctInput)
{
    Console.Write("Enter your age: ");
    string input = Console.ReadLine();
    correctInput = int.TryParse(input, out age);
}
```



#2) Dividing by zero

```
int sum = 0;
int count = 0;
Console.Write("Enter your age: ");
int age = int.Parse(Console.ReadLine());
while (age > 0)
{
    sum += age;
    count++;
    Console.Write("Enter your age: ");
    age = int.Parse(Console.Reac
}
double average = sum / count;
```

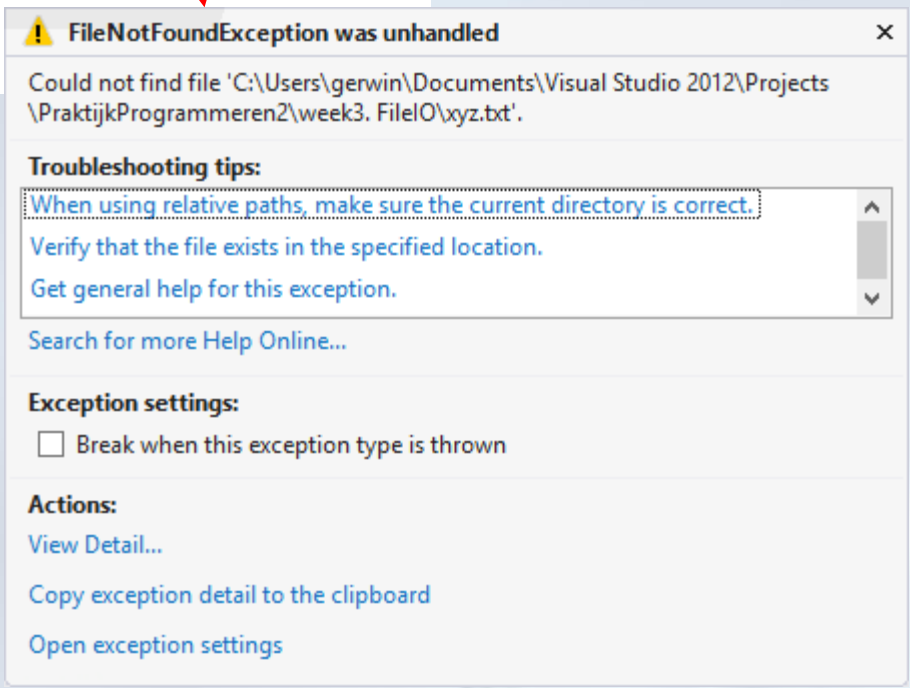


#2) Dividing by zero (prevent)

```
int sum = 0;
int count = 0;
Console.Write("Enter your age: ");
int age = int.Parse(Console.ReadLine());
while (age > 0)
{
    sum += age;
    count++;
    Console.Write("Enter your age: ");
    age = int.Parse(Console.ReadLine());
}
double average = 0;
if (count > 0)
    average = sum / count;
```

#3) Reading from a file

```
string filename = "..\\..\\xyz.txt";

StreamReader reader = new StreamReader(filename);

// ...

reader.Close();
```

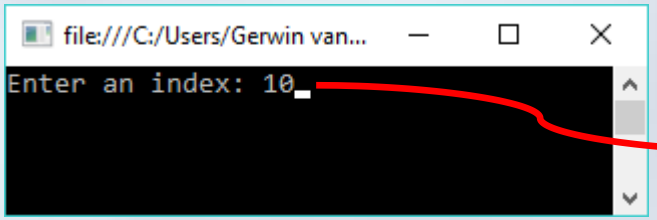


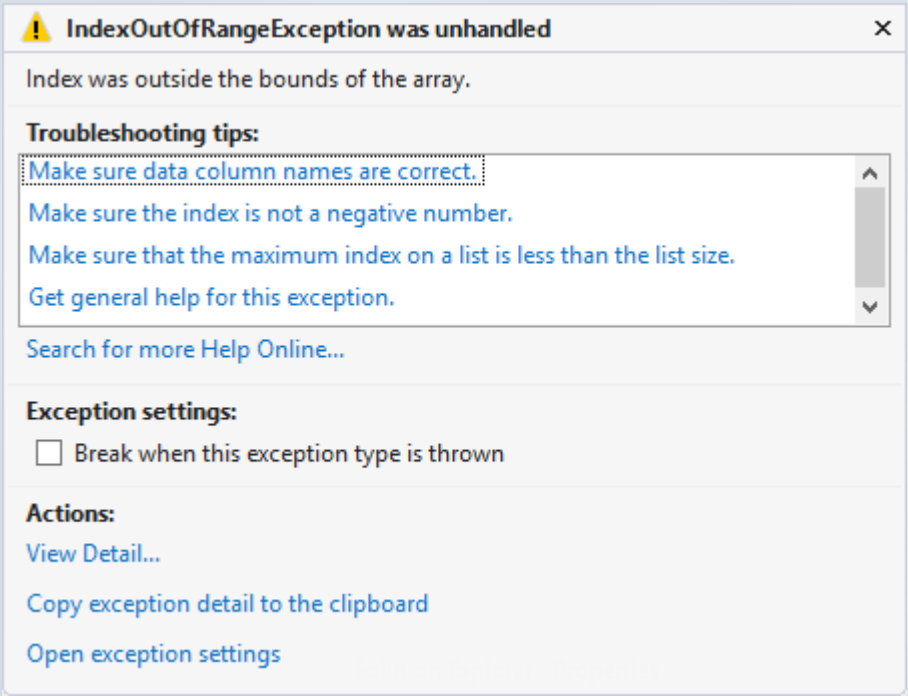
#3) Reading from a file (prevent)

```
string filename = "..\\..\\xyz.txt";

if (File.Exists(filename))
{
    StreamReader reader = new StreamReader(filename);

    // ...

    reader.Close();
}
```

#4) Array indexing

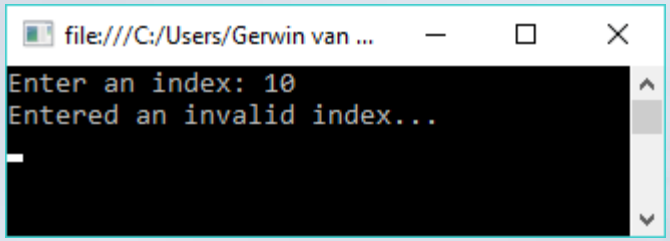
```
int[] numbers = { 1, 1, 2, 3, 5, 8, 13, 21, 34, 55 };
Console.Write("Enter an index: ");
int index = int.Parse(Console.ReadLine());
Console.WriteLine("number at position {0} is {1}", index, numbers[index]);
```





#4) Array indexing (prevent)

```
int[] numbers = { 1, 1, 2, 3, 5, 8, 13, 21, 34, 55 };
Console.Write("Enter an index: ");
int index = int.Parse(Console.ReadLine());
if ((index >= 0) && (index < numbers.Length))
    Console.WriteLine("number at position {0} is {1}", index, numbers[index]);
else
    Console.WriteLine("Entered an invalid index...");
```



Exceptions

- But you can't prevent everything…
- And some errors are 'showstoppers' (can't continue)

- Exceptions we've seen …
  - FormatException
  - DivideByZeroException
  - FileNotFoundException
  - IndexOutOfRangeException
- … can be 'catched'

Catching Exceptions

- Exceptions in a try-block are caught in a catch-block

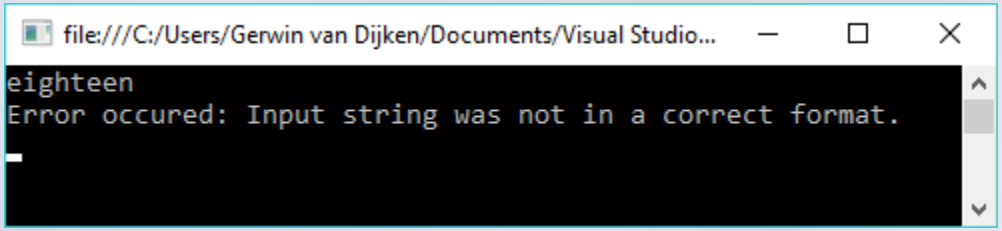
```
try
{
    string input = Console.ReadLine();
    int age = int.Parse(input);
    Console.WriteLine("You are {0} years old.", age);
}
catch (Exception e)
{
    Console.WriteLine("Invalid value for age!");
}
```

This line will not be executed when an exception occurs.

Catching Exceptions

- Exception object contains information

```
try
{
    string input = Console.ReadLine();
    int age = int.Parse(input);
    Console.WriteLine("You are {0} years old.", age);
}
catch (Exception e)
{
    Console.WriteLine("Error occured: " + e.Message);
}
```



The Exception object contains information about the error.

Finally

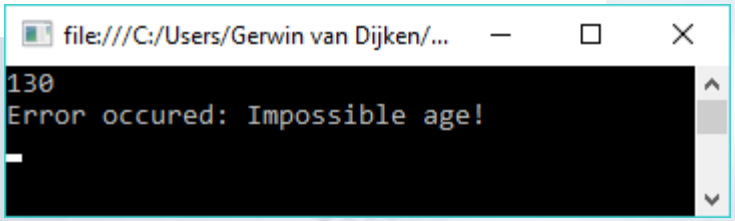
```
string filename = "..\\..\\xyz.txt";
StreamReader reader = null;
try
{
    reader = new StreamReader(filename);

    // read age from file
    string input = reader.ReadLine();
    int age = int.Parse(input);
}
catch (Exception e)
{
    Console.WriteLine("Error occured: " + e.Message);
}
finally
{
    reader.Close();
}
```

catch-block is only executed when an exception occurs in the try-block

finally-block is always executed

‘Throwing’ your own Exception

```
try
{
    string input = Console.ReadLine();
    int age = int.Parse(input);

    if (age > 120)
        throw new Exception("Impossible age!");

    Console.WriteLine("You are {0} years old.", age);
}
catch (Exception e)
{
    Console.WriteLine("Error occured: " + e.Message);
}
```



Catching all exceptions in the main

```
static void Main(string[] args)
{
    try
    {
        Program myProgram = new Program();
        myProgram.Start();
    }
    catch (Exception exception)
    {
        Console.WriteLine("Exception occured: {0}", exception.Message);
    }
    Console.ReadKey();
}
```

With this catch-block, the program will never crash, any exception will be catched and its message will be printed …

… but it makes more sense to react to exceptions in such a way that your program can continue!

Homework

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

- Assignments lesson 4
  (can be found on Moodle)