



# Programmeren 2

# 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)	Class Libraries / Gelaagde 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>

# Class Libraries

# Herbruikbare methoden

- In de eerste week hebben we een aantal veelvoorkomende Lees-methoden gemaakt (LeesInt, LeesString)
- Deze methoden kunnen gebruikt worden in meerdere projecten; tot nu toe moesten we ze kopiëren...
- Als een methode aangepast moet worden (bugfix, of om deze efficiënter te maken, ...) dan moet dit in meerdere projecten uitgevoerd worden
- Er is een betere manier → maak een bibliotheek (library) met herbruikbare methoden

# Class Library aanmaken (DLL)

## Create a new project

Search for templates (Alt+S)

[Clear all](#)

### Recent project templates

Console App (.NET Core) C#

Windows Forms App (.NET Core) C#

C#

Windows

Library

Class Library (.NET Framework)  
A project for creating a C# class library (.dll)

C# Library Windows

Class Library (.NET Core)  
A project for creating a class library that targets .NET Core.

C# Library Linux macOS Windows

WPF Custom Control Library (.NET Framework)  
Windows Presentation Foundation custom control library

C# Desktop Library Windows XAML

WPF User Control Library (.NET Framework)  
Windows Presentation Foundation user control library

C# Desktop Library Windows XAML

Windows Forms Control Library (.NET Framework)

Back

Next

*Is it a Console App?  
Is it a Windows Forms app?  
No, it's a Class Library!*

# Class Library aanmaken

Configure your new project

Class Library (.NET Core) C# Library Linux macOS Windows

Project name

MyTools

Location

C:\Users\gerwin.vandijken\source\repos\Programming2-demos

Back Create

Dit project (Class Library) wordt toegevoegd aan een bestaande solution

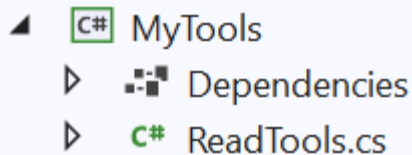
# Class Library aanmaken

Het is net als `Math.Abs(...)`:  
\* namespace: `system`  
\* class: `Math`  
\* (static) methode: `Abs`

Een Class Library bevat public classes ...

... met public methoden

Dit zie je in de Solution Explorer (geen `Program.cs`)



```
MyTools
├── Dependencies
└── ReadTools.cs
```

```
namespace MyTools
{
    public class ReadTools
    {
        public static int ReadInt(string question)
        {
            Console.Write(question);
            int value = int.Parse(Console.ReadLine());
            return value;
        }

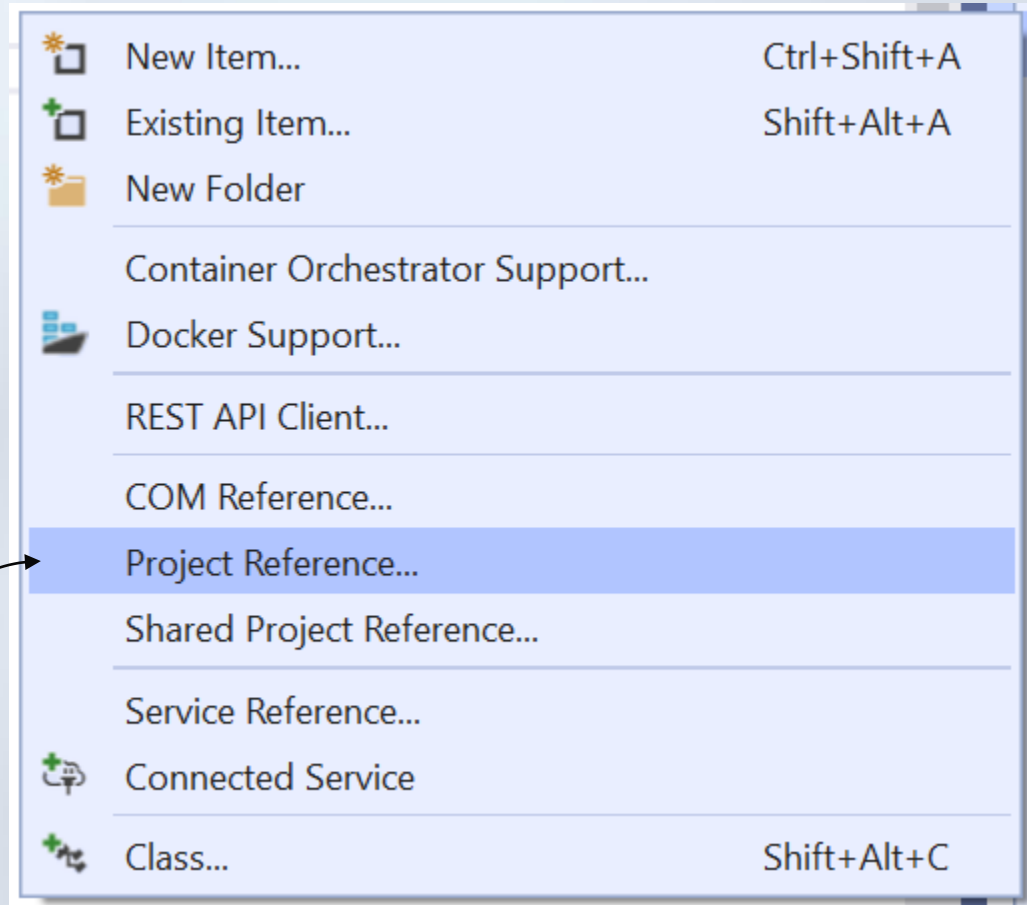
        public static string ReadString(string question)
        {
            Console.Write(question);
            string value = Console.ReadLine();
            return value;
        }

        // more methods here...
    }
}
```

# Class Library gebruiken

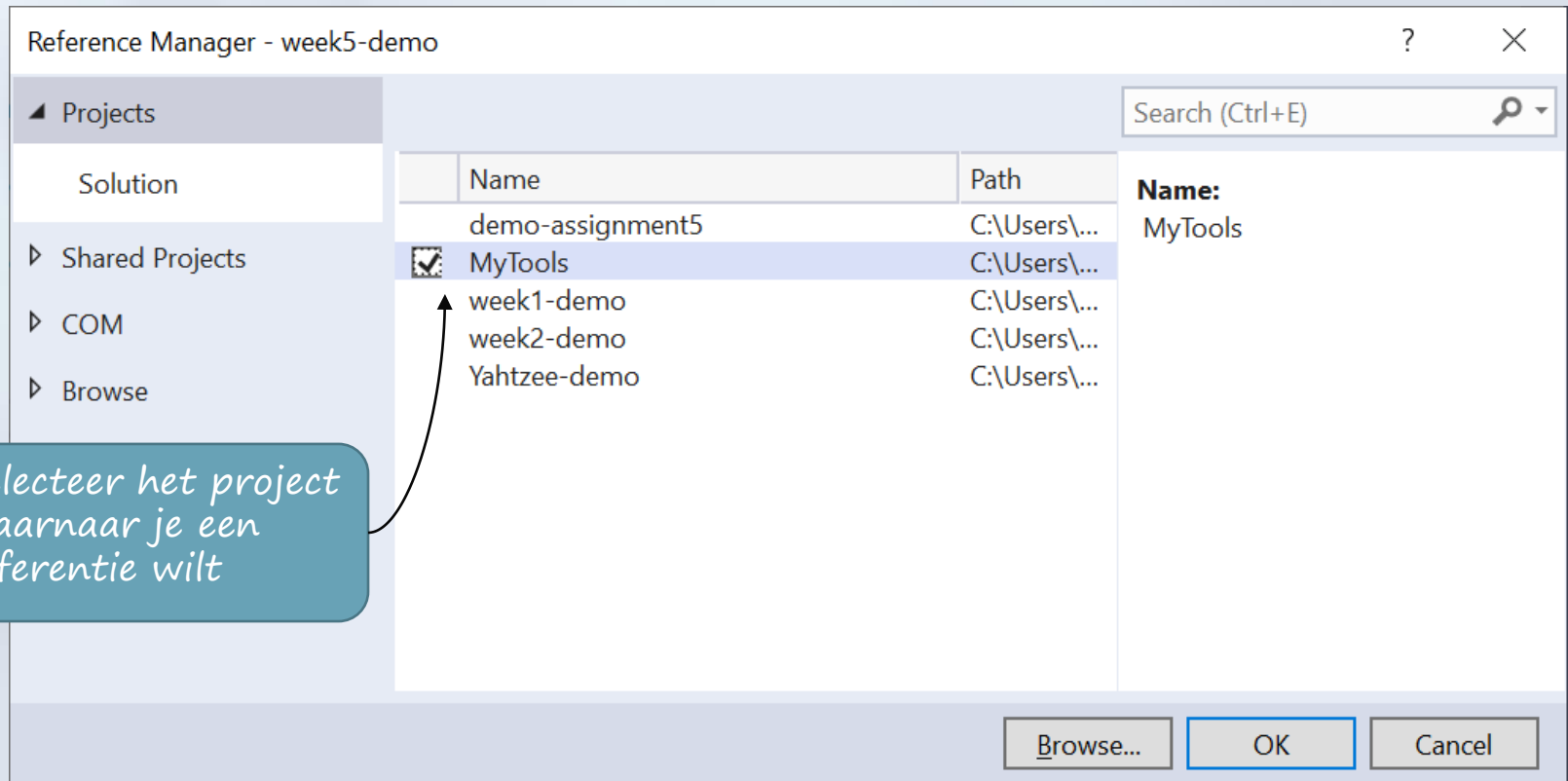
*Een project heeft een referentie nodig naar een Class Library*

*(rechtermuis klik op project, Add | Project Reference...)*





# Class Library gebruiken



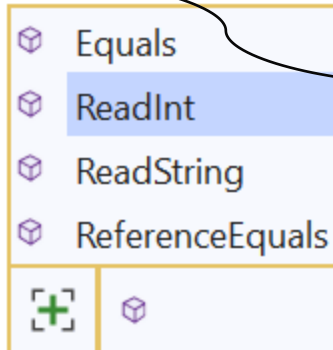
# Class Library gebruiken

```
using MyTools;

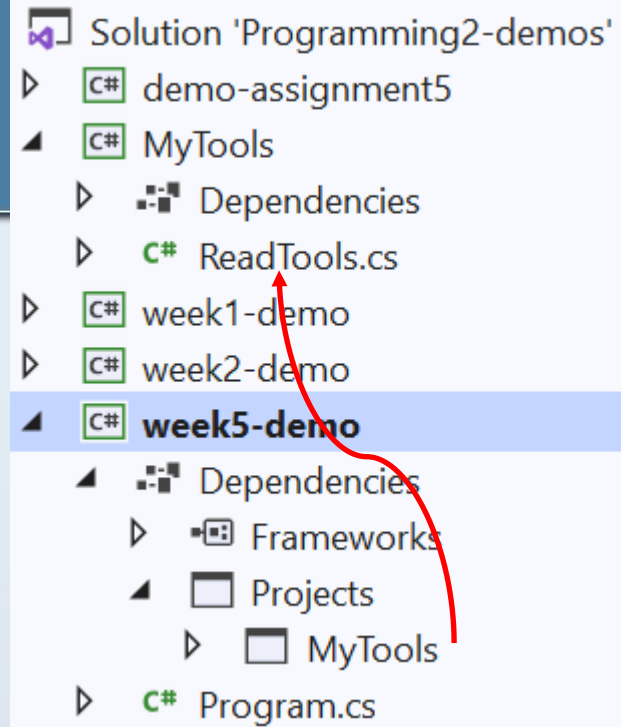
namespace week5_demo
{
    class Program
    {
        static void Main(string[] args)
        {
            Program myProgram = new Program();
            myProgram.Start();
        }

        void Start()
        {
            ReadTools.
        }
    }
}
```

Met een using-statement kun je makkelijker een class gebruiken (zoals class ReadTools)



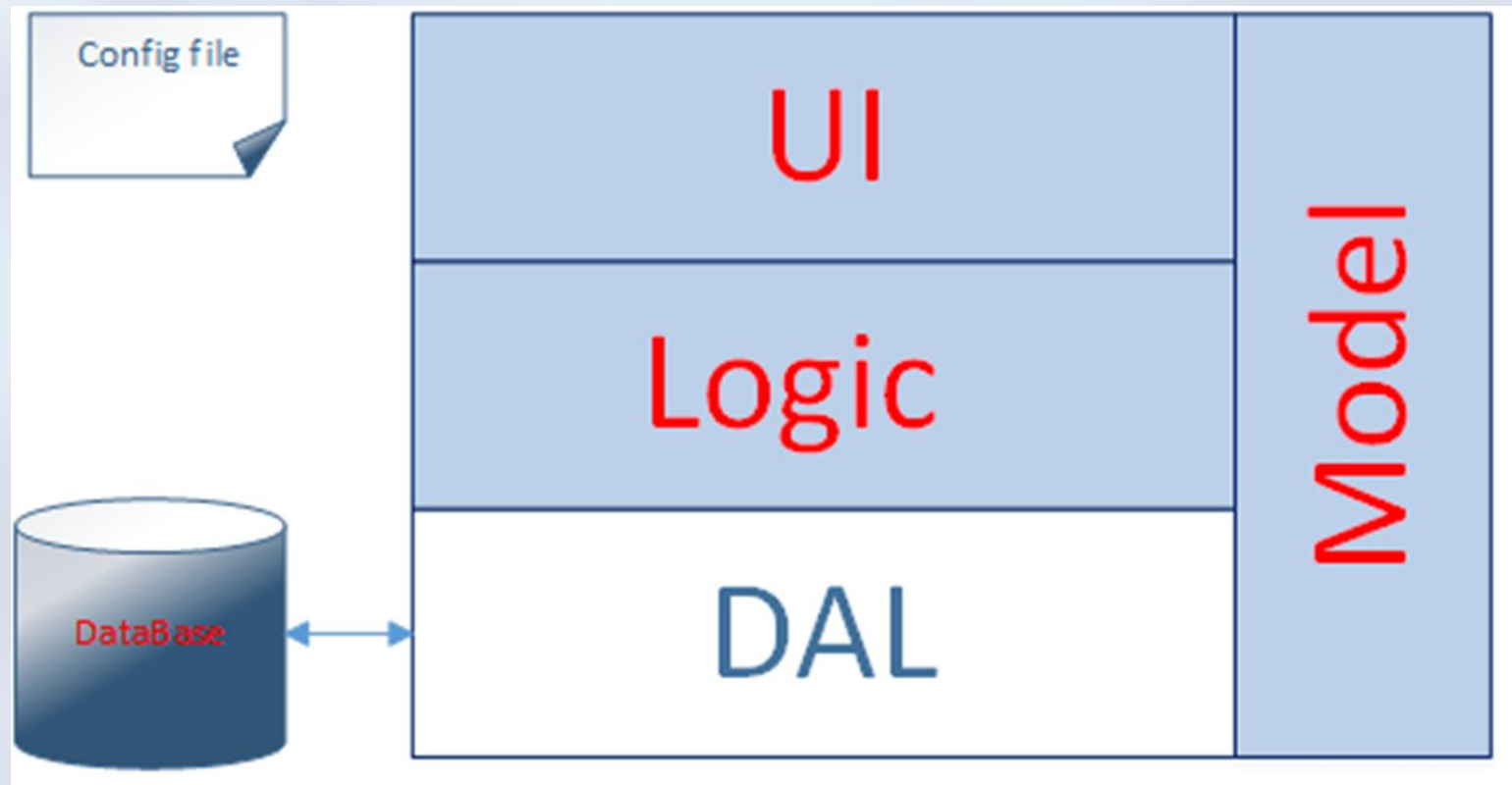
Type nu ReadTools. en de methode die je wilt gebruiken



Project week5-demo heeft een referentie naar Class Library MyTools

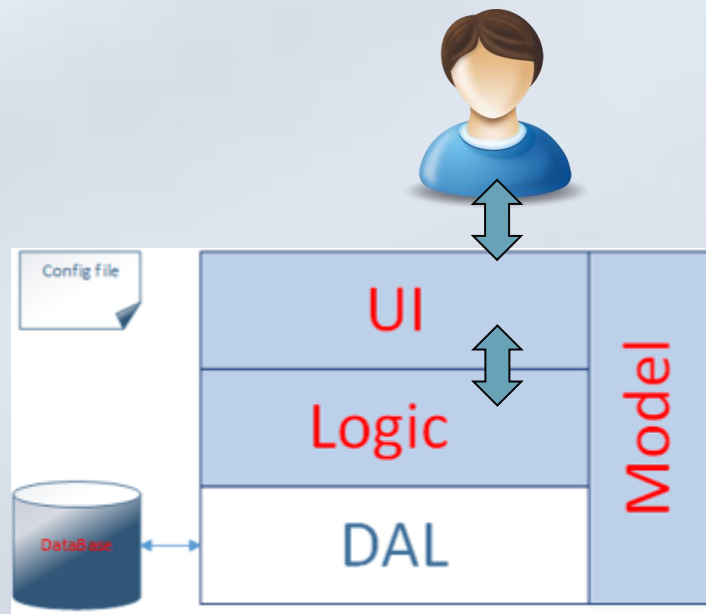
# Gelaagde Architectuur

# Gelaagde Architectuur



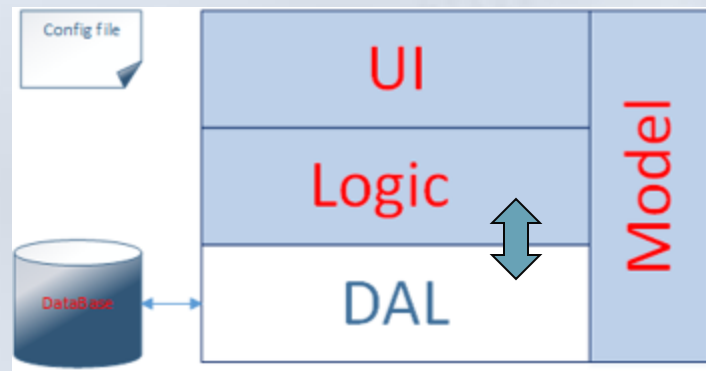
# User Interface laag (UI)

- This layer contains the actual application
- Responsible for contact with the user (input and output)
- Communicates with the Logic layer



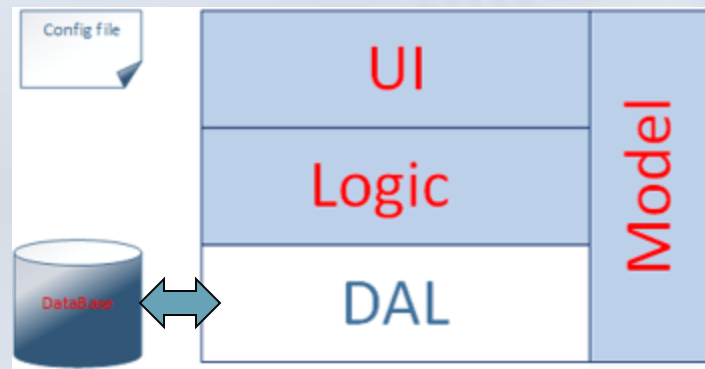
# Logica laag

- The logic layer contains the core of the system
- It contains the (business) logic
- The logic layer contains classes with methods to process the core functionalities of the application
- The logic layer delegates all persistence/database functionality to the DAL layer



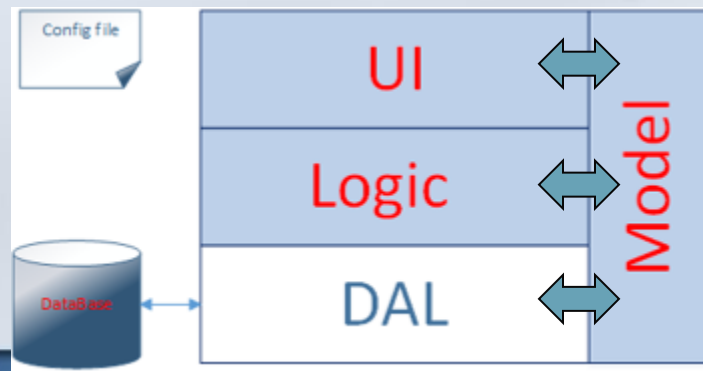
# Data Access Laag (DAL)

- A Library (DLL) to access the database
- The methods return 'model' objects
- DAL is responsible for converting (database) data to objects, and vice versa
- SQL is only used in the DAL layer  
*(create/insert, read/select, update, delete)*



# Model laag

- Contains Model objects
  - Model objects represent the 'things' in the systeem
  - Model objects are used in all layers
  - e.g. Person, Meeting, Customer, Book, Card, Account,
- When a Model object is returned from the DAL layer, all fields of this object are filled (*with coherent data*)  
This means we don't work with half-filled objects!





# Oefening

# Oefening: Lingo

- Ontwerp een Lingo spel waarin de gebruiker een 5-letter woord moet raden.
- De gebruiker krijgt 5 pogingen; bij elke poging geeft de gebruiker een (5-letter) woord op en krijgt hier feedback over: welke letters correct, welke letters niet correct, welke letters aanwezig maar op verkeerde plek.



# Oefening: Lingo

lingo woord = spook

speler woord = stoep

status[0] = **correct**

status[1] = incorrect

status[2] = **correct**

status[3] = incorrect

status[4] = **wrong-position**

speler woord = knoop

status[0] = **wrong-position**

status[1] = incorrect

status[2] = **correct**

status[3] = **correct**

status[4] = **wrong-position**

speler woord = kruik

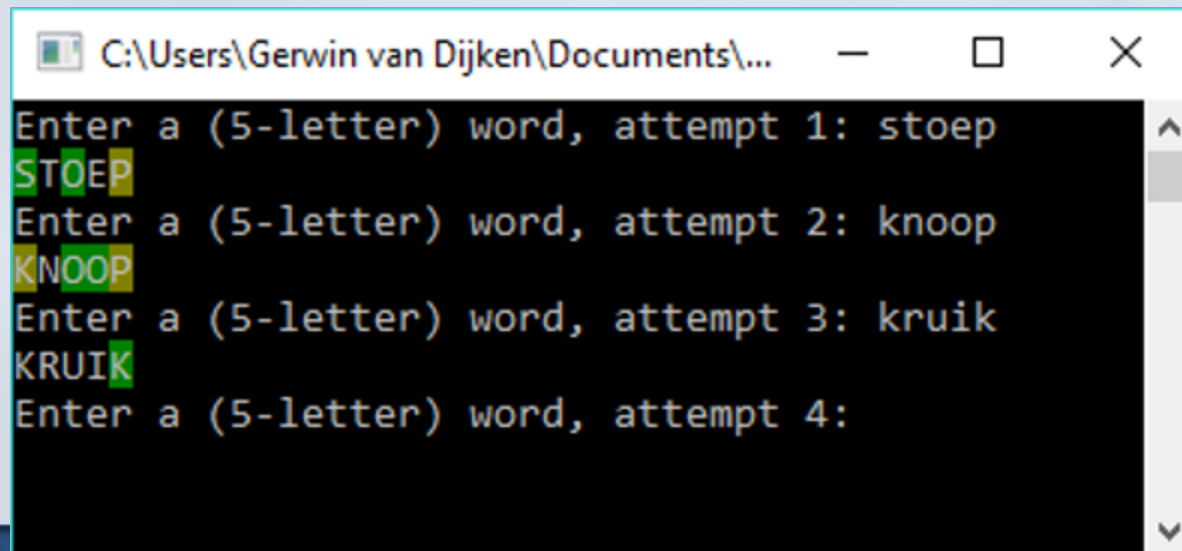
status[0] = incorrect (!)

status[1] = incorrect

status[2] = incorrect

status[3] = incorrect

status[4] = **correct**



```
C:\Users\Gerwin van Dijken\Documents\...  
Enter a (5-letter) word, attempt 1: stoep  
STOEP  
Enter a (5-letter) word, attempt 2: knoop  
KNOOP  
Enter a (5-letter) word, attempt 3: kruik  
KRUIK  
Enter a (5-letter) word, attempt 4:
```

# Oefening: Lingo

- top-down (*stepwise refinement*)

- 1) define the main task of the program
- 2) define the subtasks (needed by main)
- 3) define the subsubtasks

- bottom-up

- 1) define the subtasks of the program
- 2) define the main task of the program (call subtasks)

# Oefening: Lingo

## **Start(filename)**

```
words = ReadWords(filename, 5)
lingoWord = SelectWord(words)
```

```
lingoGame = new LingoGame()
lingoGame.Init(lingoWord)
PlayLingo(lingoGame)
```

## **ReadWords(filename, wordLength)**

```
// read words with length <wordLength> from file...
```

## **SelectWord(words)**

```
// return random word from list
```

# Oefening: Lingo

```
PlayLingo(lingoGame)
```

```
    attemptsLeft = 5
```

```
    wordLength = lingoGame.lingoWord.Length
```

```
    while attemptsLeft > 0 and !lingoGame.WordGuessed()
```

```
        playerWord = ReadPlayerWord(wordLength)
```

```
        letterResults = lingoGame.ProcessWord(playerWord)
```

```
        DisplayPlayerWord(playerWord, letterResults)
```

```
        attemptsLeft = attemptsLeft - 1
```

```
    return lingoGame.WordGuessed()
```

# Oefening: Lingo

**ReadPlayerWord(length)**

do

word = ReadString()

while (word.Length <> length)

return word

**DisplayPlayerWord(playerWord, letterResults)**

for i = 0 to playerWord.Length - 1

if (letterResults[i] = LetterState.Correct)

BackgroundColor = DarkGreen

else if (letterResults[i] = LetterState.WrongPos)

BackgroundColor = DarkYellow

display playerWord[i]

ResetColor()

# Oefening: Lingo

```
[class LingoGame]
```

```
public enum LetterState { Correct, Incorrect, WrongPosition }
```

```
public string lingoWord
```

```
public string playerWord
```

```
Init(lingoWord)
```

```
    this.lingoWord = lingoWord
```

```
    this.playerWord = ""
```

```
WordGuessed()
```

```
    return lingoWord == playerWord
```



# Oefening: Lingo

Lingo word : T R O O P

Player word : O R D E R

(reference letters: T O O P)

```
[class LingoGame]
```

```
ProcessWord(playerWord)
```

```
    this.playerWord = playerWord
```

```
    letterResults = new LetterState[lingoWord.Length]
```

```
    refLetters = new List<char>()
```

```
    for i = 0 to lingoWord.Length - 1
```

```
        if lingoWord[i] <> playerWord[i]
```

```
            refLetters.Add(lingoWord[i])
```

```
    ... (see next slide)
```



# Oefening: Lingo

Lingo word : T R O O P

Player word : O R D E R

*(reference letters: T O O P)*

**ProcessWord(playerWord)**

*... (see previous slide)*

```
for i = 0 to playerWord.Length - 1
    if lingoWord[i] = playerWord[i]
        letterResults[i] = LetterState.Correct
    else
        if refLetters.Contains(playerWord[i])
            letterResults[i] = LetterState.WrongPosition
            refLetters.Remove(playerWord[i])
        else
            letterResults[i] = LetterState.Incorrect

return letterResults
```

# Huiswerk

- Bestudeer de aangegeven paragrafen uit het 'Yellow Book' (zie Moodle)
- Week 5 opdrachten (zie Moodle)