# OITHREAD - Threading in C#

## Module book

2023-2024

**Version 3.2 / 09-01-2024**

NHL STENDEN hogeschool

Threading in C#

Version 3.2
09-01-2024

Academic year 2023-2024


| Module coordinator | Teachers |
| --- | --- |
| Rob Loves<br>rob.loves@nhlstenden.com | Rob Loves<br>rob.loves@nhlstenden.com |

# Preface

In this module you will learn more about working with (multi-)threading. Multi-threading is performing multiple tasks on multiple threads simultaneously.

According to Wikipedia:
*"In computer science, a thread is a process that is executed within a process. Using threads, a computer program can perform multiple tasks simultaneously, just as an operating system with multi tasking can run multiple processes at the same time. The use of multiple threads is called multi-threading.*

*One advantage of threads is the possibility to divide a process over multiple processors. Threads can also be beneficial on a machine with one processor: for example, one thread can perform a calculation while others wait for input or output. Without users being aware, the threads on a machine with one processor use the processor time alternately so that the tasks appear to be performed simultaneously.*

*Using threads, a web browser can, for example, perform multiple tasks simultaneously, such as retrieving a web page from a server and completing a user's actions in menus or dialogue windows. The threads that communicate with the server often need to wait until the messages are delivered and answered."*

The Threading in C# module provides a good foundation to later start with multi-threading in C# or in a different language. You learn the information about multi-threading from a book. After working through the book, you must devise an assignment related to multi-threaded programming.

The main objective of this module is to acquire a good theoretical and practical foundation in multi-threading. After the Threading in C# module, you will have a good foundation to work in this professional field later on.

When working as a software developer in a professional capacity, you will certainly come across a multi-threaded environment in which, for example, you will divide different processes over different cores.

The study load for this module is 84 hours (3 EC).

Experience with various methods used at Stenden University in the past, has proved that the method of the Threading in C# module is the best method for teaching Treading in C# to a novice programmer. After student evaluations, we have modified
the curriculum so that there is more time for the final assignment and for testing the acquired cognitive knowledge.

This module uses a final assignment as final examination. For this assignment you need to create a multi-threaded program in a group, using the elements from this module. The mark students will receive for this assignment, is the mark given for the entire module.

Rob Loves en Rob Smit.

Emmen, February 3rd 2023

# Contents


**1 Introduction 1**

1.1 Competences 2

1.2 Theme of the module 3

1.3 Objectives of the module 3

1.4 Prerequisite knowledge 3

1.5 Conventions 3

1.6 Versie management 4

1.7 Attendance of guest lectures, excursions or workshops 5

**2 Examination 6**

2.1 General assessment of final group assignment 6

2.2 Assessment of the presentation 6

2.3 Module resit 6

**3 Programme 7**

3.1 Teaching methods 7

3.2 Curriculum overview 8

3.3 Weekly schedule 9

**4 Structure & Organisation 11**

**5 Literature / software 12**

5.1 Mandatory study material 12

5.2 References 12

5.3 Software 12

**6 Module evaluation 13**

**7 Appendices 14**

7.1 Final asssignment 14

7.2 Work method 15

7.3 Scoring rubrics 16

7.4 Individual grade 17

# 1 Introduction

Multi-threaded applications, combined with different programming languages, are one of the basic components for computer scientists today. In this module, the multi-threaded component will be provided in combination with one of the most commonly used languages: C# & .NET.

**Typical professional situation**

Johan has been working for a year as a software engineer at All Systems Go. Johan completed a Bachelor of Information Technology, during which he acquired plenty of programming skills as well as experience working in projects. He also followed various courses to increase his knowledge to make user-friendly computer interfaces.

In the past year, Johan fully participated in the GettyWeather project, which resulted in a system that retrieves information on the weather from various databases. He worked on this project as a junior software engineer. His main contribution was developing the software based on the functional requirements in the language of C#.

Johan is able to strike the right balance between the existing and new technology. He correctly estimates the financial and legal consequences of his work and product. He works together with colleagues (from various fields) and communicates with colleagues and clients.

Johan can work independently and in a team. He can communicate with various parties. He has broad technical expertise and is sensitive to the needs of the customer. He is quick at learning new skills in various areas, and is adept at various design methods, programming languages and software development tools.

In the medium term, Johan can develop into a senior software engineer and in the long term, into a software development manager.

## 1.1 Competences

In this module you will work as a starting professional on four different competences that relate to managing, designing and realising of applications.

| | Managing | Analysing | Consulting | Designing | Realising |
|---|---|---|---|---|---|
| User Interaction | > Applying version management and setting up and configuring a collaboration environment for the realisation of ICT and/or digital media products, taking into account maintainability and available resources. **(level 2)** | | | | |
| Business Processes | | | | | |
| Infrastructure | | | | | |
| Software | > Setting up and using a management system to support software development as a team **(level 1)** | | | > Designing a software system using modelling techniques according to a standard method. **(level 1)** | > Building a software system consisting of several sub-systems, using existing components, and making it available. **(level 2)** |
| Hardware interfacing | | | | | |

## 1.2 Theme of the module

In this module period, students must solve a technical problem using a multi-threaded system. The multi-threaded system consists of various components.

## 1.3 Objectives of the module

After successfully completing this module, students have the ability to:

> Explain and/or apply the following concepts:
  - (Multi)Threading;
  - Locking;
  - Mutex;
  - Semaphore;
  - Thread pool;
  - Task Parallel Library / Async & Await;
  - Linq & Plinq;
  - Asynchronous I/O.
> Using Microsoft Visual Studio 2015 or newer to:
  - Make a class diagram;
  - Generate working code from the class diagram.
> Design and build a multi-threaded system that communicates using C# over various processor cores.

## 1.4 Prerequisite knowledge

Students must meet the requirements that apply to the "C# 2", "Data Structures" and "Operating Systems" modules, to take part in the module "Threading in C#".

Other prerequisite knowledge for taking the module is described in the most recent version of the OER.

## 1.5 Conventions

The conventions that apply to the designation and structure of the programme code can be found on Blackboard within ICT-algemeen.

## 1.6 Versie management

| Version | Date | Author | Description |
|---|---|---|---|
| 1.0 | 24-10-2011 | B. Meijerink / J. den Ouden | Initial version |
| 1.1 | 04-04-2012 | B. Meijerink / J. den Ouden | Changes to the assessment system. |
| 1.2 | 06-02-2013 | B. Meijerink / J. den Ouden | Changes for the new academic year. |
| 1.3 | 03-02-2014 | B. Meijerink / J. den Ouden | Changes for the new academic year. |
| 1.6 | 03-12-2015 | J. Pijpker / R. Smit | Revision of assignment and planning. |
| 2.0 | 30-08-2016 | R. Smit | Complete revision of module book and planning. |
| 2.1 | 21-12-2016 | R. Smit | Minor textual changes. |
| 2.2 | 31-01-2018 | R. Loves | Changes for the new academic year. Rewritten some texts and added a new scoring rubrics. |
| 2.3 | 24-12-2018 | R. Loves | Changes for the new academic year. |
| 2.4 | 04-02-2020 | R. Loves | Changes for the new academic year. |
| 2.5 | 14-01-2021 | R. Loves | New layout and changes for the new academic year. |
| 2.6 | 19-01-2022 | R. Loves<br>R. Smit | Changes for new academic year.<br>Scoring rubrics adjusted.<br>End assignment adjusted. |
| 2.7 | 31-12-2022 | R. Loves | Changes for new academic year. |
| 3.1 | 03-02-2023 | R. Smit | Align versioning with Dutch module book.<br>Small adjustments to the objectives of the module (grouping).<br>Minor edit to assignment. |
| 3.2 | 09-01-2024 | R. Loves | .Net Maui instead of UWP.<br>Changes for new academic year. |

## 1.7 Attendance of guest lectures, excursions or workshops

A guest lecture, excursion or workshop can be scheduled during the course of this module. Attendance is compulsory. If a student is unable to attend, the student will be given an alternative assignment.

# 2 Examination

There will be a final assignment, completed in groups of up to four students, to assess whether you have achieved the objectives of this module.

Table 2.1 outlines the final examination along with relevant information on the standards, marks and credits.

Grading is done with an assessment form (Scoring Rubrics) as included in Appendix 2.

This module will be graded with a passing mark when the standard has been reached according to the scoring rubrics from Appendix 2.

*Table 2.1 Examination overview*

| Method | Max. points | Standard-% | Standard in points | Credits | Deadline | Resit |
|---|---|---|---|---|---|---|
| Final assignment | 100 | 55% | 55 | 3EC | | |
| Total | 100 | 55% | 55 | 3EC | | |

## 2.1 General assessment of final group assignment

The tasks will be graded according to the assessment form as included as Appendix 2. The group will receive a grade for the delivered **end product** to assess whether you have achieved the objectives of this module.

**The deadline for handing in the assignment is the Friday of module week 8, 5.00 PM.**

Students pass this module if the standard of the group assignment is met.

## 2.2 Assessment of the presentation

The final presentation will be graded according to the assessment form as included in Appendix 2. The presentation must be completed using a PowerPoint or equivalent substitute. The group presentation can be between 6 and 8 minutes long.

## 2.3 Module resit

Students who do not receive a passing grade for the final group assignment may do another assignment.

# 3 Programme

In this module, we will be working with multi-threaded systems in C#, with the "Multithreading in C# 5.0 Cookbook" from the literature list.

The curriculum overview in Chapter 3.2 lists the tasks that must be completed.

This module contains a number of weekly lectures, seminars and a presentation. The remainder of the module is mainly self-study and group work. At the end of the module, the results achieved will be presented to the other students and to lecturers.

This module does not include any tutorials. Students should work independently through the chapters and complete the final assignment. During the seminars problems can be discussed.

The whole will be completed with a final assignment that must be carried out in groups of up to seven students. **Students themselves must propose and develop the assignment as described in Appendix 1.**

We refer to the information guide for the meaning of the teaching modules below.

## 3.1 Teaching methods

The diffent types of teaching methods are described below.

### 3.1.1 Lectures

During the weekly lectures a number of threading concepts and structures will be explained. These lectures, that are interactive, give a good visual representation of the potential problems students might encounter during the rest of this module. Questions can be asked about these subjects during the lecture.

There are four lectures in total. After these four lectures, seminars will follow in which questions can be asked and the lecturer can provide examples or clarifications.

### 3.1.2 Seminars

During the seminars, students have the opportunity to go through the assignments together with fellow students and a lecturer. This way students have the opportunity to discuss problems and to detect and correct any errors in their reasoning. In addition, the lecturer can use questions to determine whether learning objectives of the assignments have actually been achieved. During seminars students are expected to be prepared and to have an active, critical attitude.

## 3.2 Curriculum overview

An overview of the weekly activities is displayed below.

| Week | Task. Nr. | Study activity |
|---|---|---|
| 1 | 3.3.1<br>3.3.2 | Introductory lecture<br>Lecture chapters 1, 2 and 3 |
| 2 | 3.3.3 | Lecture chapters 4 and 5 |
| 3 | 3.3.4 | Lecture chapters 6 and 7 |
| 4 | 3.3.5 | Lecture chapter 9 |
| 5 | 3.3.6 | Deadline for proposal of assignment |
| 5-8 | 3.3.7 | Seminar |
| 8 | 3.3.8 | Deadline for handing in the assignments |
| 9 | 3.3.9 | Presentations |

## 3.3 Weekly schedule

### 3.3.1 Introductory lecture

| | |
|---|---|
| Week | 1 |
| Work method | Lecture |
| Duration | 1 |
| Objectives | > Students acquire an overview of the content of the Threading in C# module.<br>> Students acquire an overview of the content of the Threading in C# module. |
| Contents | During the introductory lecture, students receive information on the work methods/procedures, assessment, materials, and contents of the Threading in C# module. |
| Preparation | Work through the module book. |

### 3.3.2 Chapters 1, 2 and 3

| | |
|---|---|
| Week | 1 |
| Work method | Lecture |
| Duration | 1 |
| Objectives | > Students acquire insight into the basic concepts of threading.<br>> Students acquire insight into managing threads.<br>> Students acquire insight into the thread pool. |
| Contents | During this lecture the basics of threading will be repeated. In addition, some new concepts that have to do with threading will be introduced. |
| Preparation | Work through Chapters 1, 2 and 3 of the book. |

### 3.3.3 Chapters 4 and 5

| | |
|---|---|
| Week | 2 |
| Work method | Lecture |
| Duration | 1 |
| Objectives | > Students acquire insight into tasks and the task parallel library.<br>> Students acquire insight into asynchronous programming.<br>> Students acquire insight into C# specific operators. |
| Contents | During this lecture tasks and asynchronous programming will be introduced. In addition, the C# specific task actions will be discussed. |
| Preparation | Work through Chapters 4 and 5 of the book. |

### 3.3.4 Chapters 6 and 7

| | |
|---|---|
| Week | 3 |
| Work method | Lecture |
| Duration | 1 |
| Objectives | > Students acquire insight into concurrent collections and the difference with normal collections.<br>> Students acquire insight into using PLINQ. |
| Contents | During this lecture the concurrent collection will be explained, together with its advantages when compared to normal collections. In addition, the sub-languages LINQ and PLINQ are discussed. |
| Preparation | Work through Chapters 6 and 7 of the book |

### 3.3.5 Chapter 9

| | |
|---|---|
| Week | 4 |
| Work method | Lecture |
| Duration | 1 |
| Objectives | Students acquire insight into asynchronous I/O. |
| Contents | During this lecture asynchronous I/O and its advantages and pitfalls will be discussed |
| Preparation | Work through Chapter 9 of the book |

### 3.3.7 Seminar

| | |
|---|---|
| Week | 5, 6 ,7 and 8 |
| Work method | Seminar |
| Duration | 1 |
| Objectives | Answering questions |
| Contents | |
| Preparation | Prepare questions, if any |

### 3.3.8 Presentations

| | |
|---|---|
| Week | 9 |
| Work method | Test |
| Duration | 1 |
| Objectives | |
| Contents | The student groups present their work . |
| Preparation | Presentations on work done |

# 4 Structure & Organisation

The schedule below is an overview of all contact hours for this module.

In addition, students are expected to plan their own (project) meetings to work on their assignments. This also applies to the time students need to prepare and complete (individual) assignments. These hours are also incorporated in the schedule below, to clearly show the expected study load per student.

*Table 4.1 Student contact hours (SCH) and study load hours (SLH) per week:*

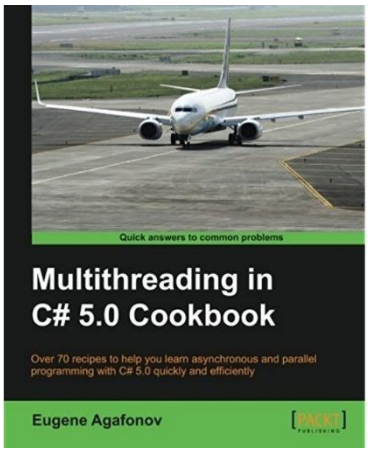
| Work method | Week 1 | | Week 2 | | Week 3 | | Week 4 | | Week 5 | | Week 6 | | Week 7 | | Week 8 | | Week 9 | | Totaal | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | SCH | SLH | SCH | SLH | SCH | SLH | SCH | SLH | SCH | SLH | SCH | SLH | SCH | SLH | SCH | SLH | SCH | SLH | SCH | SLH |
| Lecture | 2 | 6 | 1 | 3 | 1 | 3 | 1 | 3 | | | | | | | | | | | 5 | 15 |
| Seminar | | | | | | | | | 1 | 3 | 1 | 3 | 1 | 3 | 1 | 3 | | | 4 | 12 |
| Final exam | | | | | | | | | | | | | | | | | | | 0 | 0 |
| Self study | 1 | 3 | 2 | 6 | 2 | 6 | 2 | 6 | 2 | 6 | 2 | 6 | 2 | 6 | 2 | 6 | | | 15 | 45 |
| **Total study activities** | 3 | 9 | 3 | 9 | 3 | 9 | 3 | 9 | 3 | 9 | 3 | 9 | 3 | 9 | 3 | 9 | 0 | 0 | 24 | 72 |
| Extra curricular activities | | | | | | | | | | | | | | | | | | | 0 | 12 |
| **End total** | | | | | | | | | | | | | | | | | | | 24 | 84 |

SCH = Student contact hours (45 minutes)
SLH = Study load hours (60 minutes)

# 5 Literature / software

Below is mentioned which literature and software is needed to successfully pass and complete the course.

## 5.1 Mandatory study material




Book: Multithreading in C# 5.0 cookbook
Author: Eugene Aganov
Publisher: Packt Publishing
ISBN: 978-1-84969-764-4.

## 5.2 References

> http://www.albahari.com/
> http://channel9.msdn.com/Blogs/bruceky/How-to-Parallelize-Your-Application-Part-2-Theads-v-Tasks
> http://channel9.msdn.com/Shows/The+Knowledge+Chamber/Multi-Core-and-Parallel-Programming-Practices
> http://www.albahari.com/threading/threading.pdf

## 5.3 Software

> Visual studio 2022 or newer in combination with .NET 8.0 or higher.

# 6 Module evaluation

At the end of the module period, the module will be evaluated by means of a questionnaire. This questionnaire covers all components of the module including organisational aspects, content, quality of teaching staff, etc.

We would appreciate your participation in this evaluation. The results will be used to improve the next version of this module.

# 7 Appendices

## 7.1 Final asssignment

The final assessment consists of a final assignment where the students have to work in groups of up to four students to solve a problem. This problem must be proposed by students in week 5 at the latest and **must** include at least the following components:

- C# .net 8.0 or higher;
- Multi-threading (See the rubric for clarification);
- .Net Maui;
- A clean GUI for operating the program;
- Version control must be used.

In the absence of one or more of these components in the submitted project, the entire project does not meet the minimum requirements and the grade "Not Assessable" (NBB) will be awarded.

The following components **may** be used to solve the problem:
- PLINQ (with a database of at least 100,000 records - e.g. photos, weather data etc.);
- Multiple asynchronised I/O calls to an API;
- Data visualisation using plots/graphs and the thread pool;
- Batch processing using tasks of the thread pool.

This problem must be presented to the lecturer for approval who may then provide feedback. **After the lecturers approval** the students can start working on the solution. With multiple assignments, tutors may reallocate the given assignments. Assignments are then given to students and they must develop them individually. Students can plan their own schedules.

The lecturer expects a design and the complete development of the problem. **Groups may only begin developing after the lecturer has approved the design**.

The final assignments will be graded according to the assessment form (Scoring Rubics) as included in Appendix 7.3. The final assignment must be handed in with the lecturer.

Final assignment must be completed with a presentation.

**The deadline for handing in the assignment is the Friday of module week 8, 5.00 PM.**

## 7.2 Work method

The final assignment will be assessed on various aspects including multi-threading. In addition, the presentation is also assessed and included in the whole.

To assess the presentation, the graduation assessment form will be used. These Dublin Descriptors are being found on Blackboard in IT Placement and Graduation > Graduation Bachelor.
The scoring rubrics can be found in Appendix 3.

The final grade is determined with help of the students themselves. You can find the rules for the final grading in appendix 4.

## 7.3 Scoring rubrics

Opleiding: Informatica (Techniek)

Group:

| Soort | 0% | 25% | 50% | 75% | 100% | Scoringsperc. | Max. aant. punt | Behaalde punte |
|---|---|---|---|---|---|---|---|---|
| 1. Use of multithreading | Multithreading is being used in maximum 2 occassions, but it doesn't add much value (gui unresponsive during the threading action for instance). The same could be accomplished with sychronous methods/functions. | Multithreading is being used in more than 2 occassions, but doesn't add any value (gui unresponsive during the threading action for instance). The same could be accomplished with synchronous methods/functions. | Multithreading is being used and adds value. The gui is still responsive during threading actions for instance. Less than 4 keywords from chapter 1.3 are being used. Multiple cores are in use when executing the multithreaded actions. | Multithreading has been used, it adds value to the program. 4 or more keywords from chapter 1.3 are being used. Multiple cores are in use when executing the multithreaded actions. | Multithreading has been used, it adds value to the program. 4 or more keywords from chapter 1.3 are being used as well as techniques not discussed in lectures. Multiple cores are in use when executing the multithreaded actions. | 0% | 250 | 0,00 |
| 2. User interface | No user interface | There has been no styling done to the user interface and controls aren't properly alligned, or not all texts/controls are readable. | Styling has been done to the user interface but not all controls are properly alligned. All texts/controls are readable. The application doesn't resize. | Styling has been done to the user interface and all controls are properly lined up. All text/controls are readable. Texts/controls aren't readable on resizing (not responsive). | Styling has been done to the user interface and all controls are properly lined up. All texts/controls are readable. Everything stays visible/readable on resizing (responsive). | 0% | 100 | 0,00 |
| 3. Code | 3 or more cases of redundant code, almost no classes or functions have been used. No proper allignment has been used. No error handling. | 2 or more cases of redundant code. 5 or more empty functions or unused classes. More than 50% of the classes/functions and/or parameters have no logical names and/or wrong access modifiers. Allignment is off for more than 50% of the code. No error handling. | Multiple empty functions or one unused class. Less than 50% of the classes, functions and/or parameters have no logical names. Less than 50% of the access modifiers are wrong. Alignment is off for less than 50% off the code. Insufficient error handling. | No empty functions and classes. No unused classes. No redundant code. Less than 10% of the classes, functions and/or parameters have no logical names. Less than 10% of the access modifiers are wrong. Allignment is off for less than 10% of the code. Insufficient error handling. | No empty functions and classes. No unused classes. No redundant code. All classes, functions and/or parameters have logical names. All access modifiers are correct. Allignment is correct. Error handling correct. | 0% | 250 | 0,00 |
| 4. Code conventions and comments | No code conventions have been used. No comments used or it doesn't refer to the code. | No code conventions used, or only partly. No comments used or it doesn't refer to the code. | Code conventions only partly used. Comments are available, but not everywhere it's needed or it doesn't refer to the code. | Code conventions are used. Comments are available where needed and refer to the code. | Code conventions are used. Comments are available where needed and refer to the code. | 0% | 50 | 0,00 |
| 5. Version management | No version management tool has been used. | | Version management has been used, but not all group members have commited or most of the code is being added in a couple of commits or over a short period of time. | | Version management has been used and everyone has commited code. There are multiple commits spreaded over the entire period. | 0% | 50 | 0,00 |
| Presentation | | | | | See separate Dublin descriptors presentation. | 0% | 300 | 0,00 |
| | | | | | | | ### | 0,00 |

This scoring rubrics can also be found on Blackboard.

## 7.4 Individual grade

To calculate the individual mark of a student, students need to grade each other. This is how:

Let's say the teacher has determined that the average grade is a 7. If a group consists of 4 members, the total points to distribute is 4*7 = 28 points. These 28 points can be divided amongst the group members, including yourself. The minimum is a 1 and the maximum is a 10. The points may differ 2 point from the grade given by the teacher.
E.g.: The grades given by a student are 6, 6.5, 7.5 and 8.
The average of these grades are calculated. If the average differs more than 2 points from the teachers given average, the grade won't be used for the final individual grade. In the end, all the average grades of each student given by the group are used to calculate the individual grade per student.

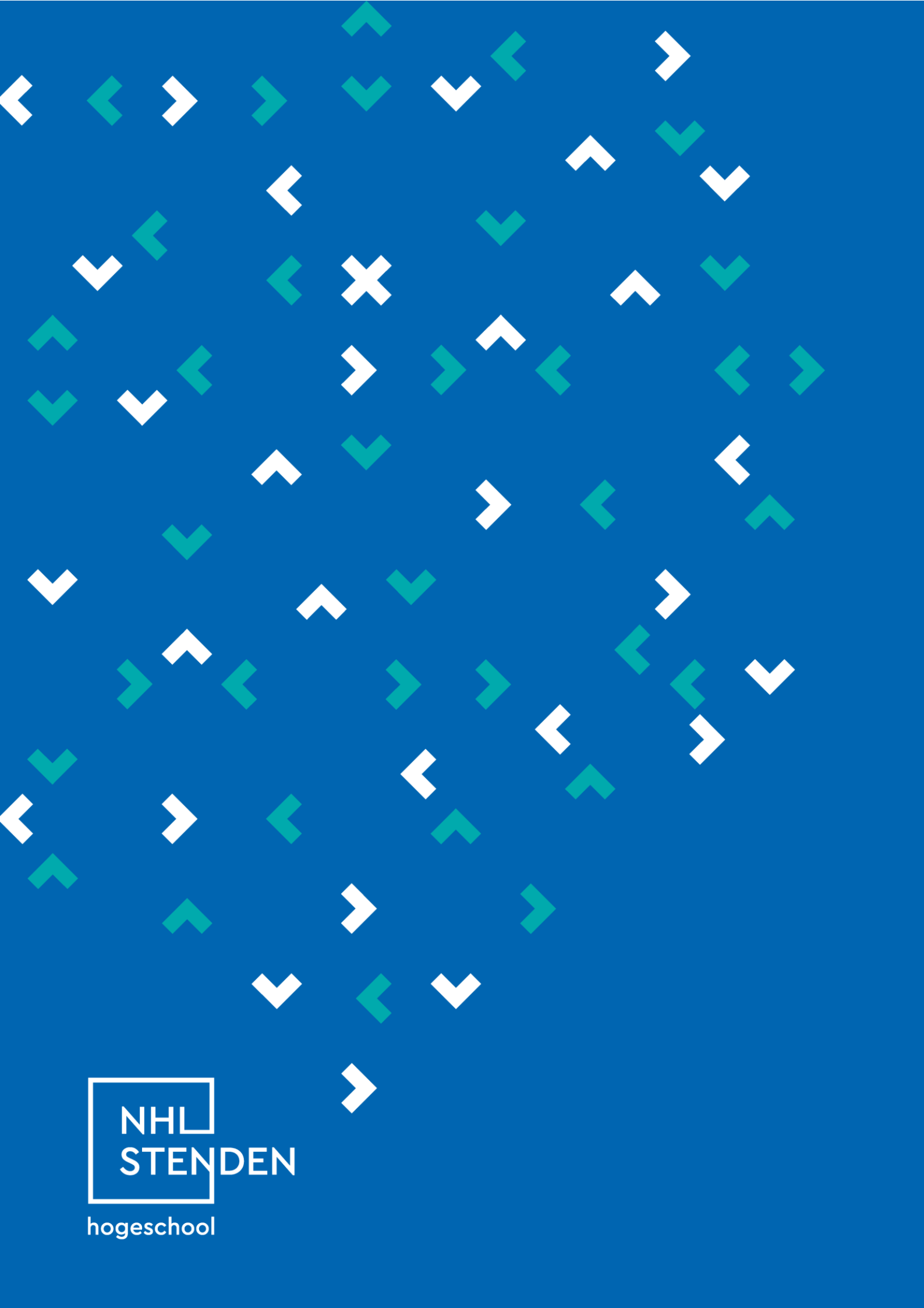
Note, in the end the teacher is responsible for the grades and he or she may deviate from the advice given by students.

**Example:**

Group grade: 7,3.
Members: 7.
Total amount of points to distribute: 7,3 * 7 = 51,1.

Distribute these point amongst your group members and yourself as fit.

| **Student name:** | **Grade:** |
|---|---|
| Pieters, Hans | |
| Hansema, Pieter | |
| Strijker, Albert | |
| Hoiting, Barbera | |
| Leffers, Eric | |
| Weerts, Bram | |
| Wighers, Reinier | |
| **Total:** | **51,1** |


NHL
STENDEN
hogeschool