OOP2 – Object-Oriented Programming 2

Modulebook

2025-2026

Version 1.2 / 30-07-2025

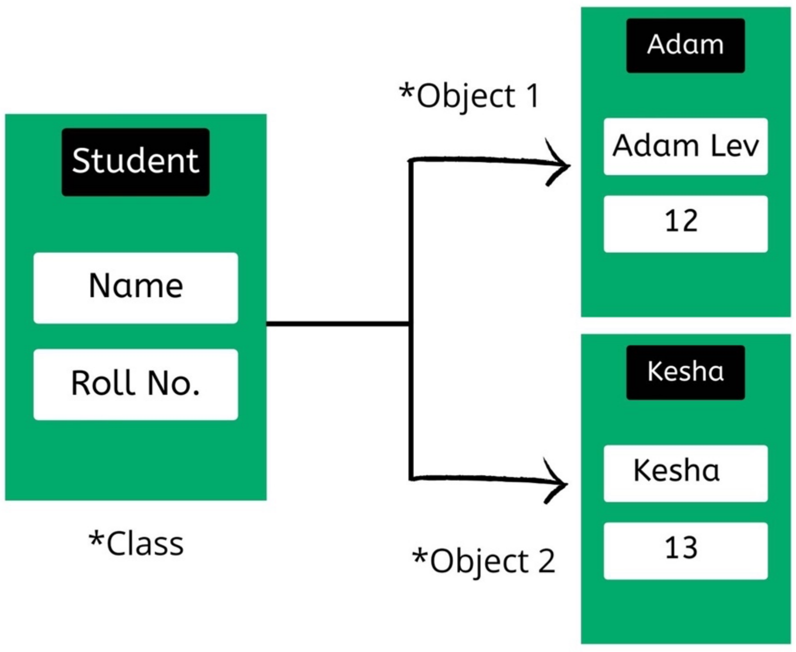
NHL STENDEN

hogeschool

Object-Oriented Programming 2

Version 1.2
30-07-2025

Academic year 2025-2026



Module coordinator	**Teachers**
Jan Doornbos	Martijn Pomp Jan Doornbos

Preface

After the introduction to Object-Oriented Programming in year 1, we now take it a step further.

In this module you will expand your knowledge of object-oriented programming. Among other things, you will work with polymorphism, abstract classes, interfaces, etc. For this, you will again go through both steps: starting with a design and then working it out in an object-oriented programming language. Whereas the design first consisted of only a class diagram, you will now work with a Use Case diagram, Activity Diagram and Sequence Diagram.

However, the primary goal of this module is not only to learn how to program in an object-oriented language, but also to learn how to break down difficult abstract problems into manageable pieces. Programming will take place in a professional development environment.

In the later work field, you will encounter object-oriented languages in a similar way as an ICT professional.

The study load for this module is 168 hours (6 EC).

The module is completed at the end with a practice exam. During this exam you will be individually tested on your skills in object-oriented programming.

Department Software,
Emmen, May 28th, 2024

Contents

1 Introduction

One of the most common occupations in the field is software engineer. Object-oriented programming is one of the basic elements for the computer scientist today. Much software is written this way. In this module, the student is introduced to object-oriented design and programming.

Typical professional situation

Paul has been working at **Pure Home Luxurity** for 1 year as a software engineer. Paul has a HBO Computer Science education behind him where he gained a lot of programming knowledge as well as working in projects.
In addition, he took several courses to increase his knowledge to create user-friendly computer interfaces.

In the past year Paul has participated fully in the **EasyCare** project which designed home automation systems for healthcare institutions. He participated in this project as a junior software engineer. His main contribution is developing the software according to the functional requirements.

Paul knows how to find the right balance between existing and new technology. He correctly assesses the financial and legal consequences of his work and product. He collaborates with colleagues (from different fields) and communicates with colleagues and clients.

Paul can work independently and in teams. He can communicate with different parties. He has broad technical knowledge and good empathy regarding what the customer wants. He can learn quickly in different areas and has mastered different design methods, programming languages and development tools to develop software.

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

1.1 Module theme

In this period, students will work on the basic concepts of object-oriented programming. We will extend the existing knowledge of object-oriented programming. You will do this in the language Java, because history has shown that this is a good language to learn these concepts. Also, this language is widely used in this region.

1.2 Learning Outcomes

By actively participating in the Object-Oriented Programming 2 module, the student works toward the generic learning outcome shown below. This learning outcome is specifically focused on level 1 and partially level 2 of the Software architecture layer.

Design

The student makes in a structured manner individually a design belonging to one or more given architecture layers. The student does this in accordance with the frameworks and/or requirements of the relevant stakeholders. The design complies with the given standards, so that at a later stage the student is able to make a well-considered realization for various (IT) problems.
During the module Object-Oriented Programming 2 the following requirements are addressed:

> Creating a Class Diagram;
> Creating a Use Case Diagram;
> Creating a Sequence Diagram;
> Creating an Activity Diagram.

> All diagrams are worked out using the Unified Modeling Language 2.5 (UML).

Realise

The student realizes in a structured manner, based on an individual design, a professional product belonging to one or more given architecture layers. The student does this in accordance with the frameworks and/or requirements of the relevant stakeholders. The realized professional product meets the given standards, so that the student delivers a realization that offers a solution for various (IT) problems.
During the module Object-Oriented Programming 2 the following requirements are addressed:

> There is a (working) product that meets the requirements of the assignment;
> Programming is done in a safe, neat and efficient manner using the given code conventions and given standards;
> The software system meets the following requirements:
  - Features data validation;
  - Uses variables;
  - Uses conditional statements;
  - Uses loops;
  - Is programmed in an object-oriented manner;
  - Contains coherent classes, fields and methods;

    - Uses different access modifiers;
    - Uses different types of collections (lists, maps *and* sets);
    - Contains inheritance;
    - Contains an interface;
    - Contains an abstract class;
    - Contains exceptions;
    - Contains enums;
    - Uses static methods and/or variables;
    - Is in line with the draft prepared.
- Software system testing meets the following requirements:
    - For each class, Unit Tests were created and conducted.

Manage & Control
The student individually manages in a structured manner the support of a professional product belonging to one or more given architecture layers. The student does this in accordance with the frameworks and/or requirements of the relevant stakeholders. The management complies with the given standards, so that the student is able to manage all products realized during the professional activities or to transfer the management to third parties.
During the module Object-Oriented Programming 2 the following requirements are addressed:

- A version control system is used;
    - An online repository is used.

1.3 Prior Knowledge

The prerequisite knowledge required to take the module is included in the most recent version of the OER.

1.4 Conventions

The conventions to be used regarding naming and format of the program code can be found in Appendix **Error! Reference source not found.**.

1.5 Version management

Version	Date	Author	Description
0.1	25-04-2023	Commission OOP	Initial version.
1.0	31-08-2023	J. Doornbos	Version for academic year 2023-2024.
1.1	28-05-2024	J. Doornbos	Version for academic year 2024-2025.
1.2	30-07-2025	J. Doornbos	Version for academic year 2025-2026.

1.6 Attendance of guest lecture, field trip or workshop

During the implementation of this module there is the possibility of scheduling a guest lecture, excursion or workshop. This is mandatory. If the student is unexpectedly not present, a substitute assignment will be provided.

2 Examination

To assess whether you have achieved the objectives of this module, an individual final test (practice test) will be used.

Table 2.1 gives an overview of the final test with relevant information about the norms, marks and credits.

The assessment will take place by means of an assessment form (Scoring Rubrics), which can be found in Appendix 7.2.

This module shall be assessed with a pass if the standard has been met, which is determined by means of the scoring rubrics in Appendix 7.2.

Table 2.1 Overview examination

Method	Max. points	Standard %	Standard in points	Credits	First Change	Resit
Diagram portfolio			O/V		Week 5	Week 7
Practical test	100	55%	55	6 EC	Week 9	Week 10 P2
Total				6 EC		

2.1 Assessment of practical test

The final assessment is a skills test where the student must create a Java program at the computer. Beforehand, the student has already submitted a portfolio of diagrams. This must be sufficient to receive the full final assessment.
This skills test will have the same level of difficulty as the material covered and must be completed within three hours.

The assessment of the practical test will be done using the assessment form (Scoring Rubrics) which can be found in Appendix 7.2.
This module is assessed with a passing grade if the standard is met for the practical test and the diagrams portfolio.

See section 2.3 for more information on assignment assessment.

2.2 Formative assessment

During period the student is given the opportunity to test his knowledge and skills. The student is given an assignment and works on it. The student then conducts a code review with a fellow student to assess their work. Also available is an elaboration that meets test level so that the student can calibrate whether his own work is sufficient.

2.3 Active participation

Active participation is expected from students. This means that the student attends tutorials and code reviews. The student makes an active contribution in these sessions. In addition, he has completed his homework from a previous week.

During the tutorials and code reviews, students are required to be present. If the student misses one of these moments, the individual final grade for this module will be reduced by 0.2, if the student misses two moments, the individual final grade will be reduced by 0.4, if the student misses three moments, the individual final grade will be reduced by 0.8, if the student misses four moments, the individual final grade will be reduced by 1.6, etc.

The same calculation applies to failure to hand in weekly homework.

If one is later than five minutes after the start of lectures or code reviews, he may be denied access to the relevant lecture.

2.4 Module resit

Students who fail the diagram portfolio can take a resit. This takes place in week 8 of the same period. Should the student fail the practical test, he can also take a resit for this. This takes place in the resit week of the following period.

For students who must resit the entire module, there is no attendance requirement. However, homework must be handed in each week. The mentioned conditions regarding homework in section 2.3 remain in force.

3 Programme

This module includes a number of (weekly) lectures, interactive lectures and code reviews. It will be concluded with a practical test. The different lecture formats are described in detail below.

3.1 College formats

The different college types are described below.

3.1.1 Lectures

All knowledge and concepts that the student needs to carry out their assignments are covered in the lectures. The lectures are not mandatory.

3.1.2 Interactive tutorials

In these lectures, students can work on their assignments. In addition, these lectures provide the opportunity for additional depth or explanation. Sometimes the choice can be made to deal with a certain problem in class. The content of these lectures is therefore not completely fixed.

3.1.3 Code Reviews

At the end of the week, so-called Code Reviews are held. In these, students have completed the previous assignment. A student's assignment is then chosen randomly and addressed in class. This will cover code conventions, effective programming, etc.
The code review is mandatory and students must have turned in their assignments before participating in the code review.

3.2 Programme overview

The following is an overview of the weekly activities.

Week	Task. No.	Study Activity
1	3.3.1	Kick-off
	3.3.2	Lecture 'Use Case Diagram, Inheritance & abstract class'
	3.3.3	Tutorial
	3.3.3	Tutorial
2	3.3.4	Code Review
	3.3.5	Lecture 'Activity Diagram & Interface'
	3.3.3	Tutorial
	3.3.3	Tutorial
3	3.3.4	Code Review
	3.3.6	Lecture 'Sequence Diagram, Collections, enums & static'
	3.3.3	Tutorial
	3.3.3	Tutorial
4	3.3.4	Code Review
	3.3.3	Tutorial
	3.3.3	Tutorial
5	3.3.4	Code Review
	3.3.7	Tutorial (Practice exam)
	3.3.7	Tutorial (Practice exam)
6	3.3.4	Code Review
	3.3.3	Tutorial
	3.3.3	Tutorial
7	3.3.4	Code Review
	3.3.3	Tutorial
	3.3.3	Tutorial
8	3.3.4	Code Review
	3.3.7	Tutorial (Practice exam)
	3.3.7	Tutorial (Practice exam)

3.3 Weekly programmes

3.3.1 Kick-off

Week	1
Work form	Lecture
Duration	45 minutes
Objectives	> The student obtains an overview of the contents of the Object-Oriented Programming 2 (OOP2) module.
Contents	During the kick-off, the student is instructed on such things as how to proceed, assessment, materials needed for the module.
Preparation	Read this module book.

3.3.2 Lecture 'Use Case Diagram, Inheritance & abstract class'

Week	1
Work form	Lecture
Duration	45 minutes
Objectives	> The student gains an understanding of what a Use Case Diagram is. > The student learns to create a Use Case Diagram. > The student gains an understanding of what inheritance is. > The student will gain an understanding of what an abstract class and abstract method is.
Contents	See objectives.
Preparation	The student is already reading up on the above topics. The student has installed Astah on his machine.

3.3.3 Tutorial

Week	1 to 8
Work form	Tutorial
Duration	Two moments of 90 minutes each
Objectives	
Contents	The material covered in the previous lecture will be discussed in more detail. Questions may be asked, and students may work on assignments.
Preparation	

3.3.4 Code Review

Week	2 t/m 8
Work form	Code Review
Duration	90 minutes
Objectives	> The student learns to review code from others and provide feedback on it.
Contents	A random student's written software is displayed. The other students present give critical feedback on the written code. In this way, students learn from each other, and the quality of the code is increased. The teacher may also choose to conduct the code review in pairs.
Preparation	The student completed the assignment scheduled for the previous week and handed in (digitally) 24 hours in advance.

3.3.5 Lecture 'Activity Diagram & interface'

Week	2
Work form	Lecture
Duration	45 minutes
Objectives	> The student will gain an understanding of what an Activity Diagram is. > The student will learn what different elements an Activity Diagram consists of. > The student learns to create an Activity Diagram. > The student gains an understanding of what an interface is. > The student will learn how to use an interface.
Contents	See objectives.
Preparation	The student pre-reads himself on the above topics.

3.3.7 Tutorial 'Sequence Diagram, collections, enum & static'

Week	3
Work form	Lecture
Duration	45 minutes
Objectives	> The student will gain an understanding of what a Sequence Diagram is. > The student will learn what components a Sequence Diagram consists of. > The student will learn to create a Sequence Diagram. > The student is introduced to other types of collections. For example, HashMap, HashSet. > The student will learn what an enum is and how it is used. > The student will learn what static means and how this keyword can be used.
Contents	See objectives.
Preparation	The student pre-reads himself on the above topics.

3.3.7 Tutorial (Practice Exam)

Week	5 & 8
Work form	Tutorial
Duration	Two times during the week of 90 minutes each.
Objectives	
Contents	> The student is given a practice test during the first tutorial. This will be discussed during the second tutorial.
Preparation	The student has completed assignments from previous weeks.

4 Structure & Organization

The schedule below gives an overview of all contact hours in this module.

In addition, students are expected to plan their own (project) meetings where they can work on the assignments. This also applies to the time the student needs to prepare and make (individual) assignments. This schedule also gives a good overview of the expected study load per student.

Table 4.1 Study Contact Hours (SCU) and Study Load Hours (SBU) per week:

Work form	Week 1		Week 2		Week 3		Week 4		Week 5		Week 6		Week 7		Week 8		Week 9		Totaal	
	SCU	SBU	SCU	SBU	SCU	SBU	SCU	SBU	SCU	SBU	SCU	SBU	SCU	SBU	SCU	SBU	SCU	SBU	SCU	SBU
Kick-off	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1
Lecture	1	2	1	2	1	2	1	2	1	2	1	2	1	2	1	2	0	0	8	16
Tutorial	4	5	4	5	4	5	4	5	4	5	4	5	4	5	4	5	0	0	32	40
Code Review	0	0	2	3	2	3	2	3	2	3	2	3	2	3	2	3	0	0	14	21
Exam	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	4	10	4	10
Self Study	0	7	0	7	0	7	0	7	0	7	0	7	0	7	0	7	0	0	0	56
Total Study Activity	6	15	7	17	7	17	7	17	7	17	7	17	7	17	7	17	4	10	59	144
Extra Activities	0	3	0	3	0	3	0	3	0	3	0	3	0	3	0	3	0	0	0	24

SCU	= Student Contact Hours (45 minutes)
SBU	= Study Load Hours (60 minutes)

5 Literature & software

Listed below is the literature and software required to successfully complete and pass the course.

5.1 Compulsory Curriculum

> This module book.

5.2 Reference work

- Java 19 JDK API Specification
  https://docs.oracle.com/en/java/javase/19/docs/api/index.html
- UML 2.5 Specification
  https://www.uml-diagrams.org
- Agile Modeling
  https://agilemodeling.com/essays/umldiagrams.htm

5.3 Software

- Astah Professional
  https://astah.net/products/astah-professional/
  Er is een licentie is beschikbaar.

- Java SDK
  https://www.oracle.com/nl/java/technologies/downloads/

- IntelliJ
  https://www.jetbrains.com/idea/

6 Module evaluation

The module will be evaluated by means of a questionnaire at the end of the module. This questionnaire will cover all parts of the module including organizational aspects, content, quality of teaching staff, etc.

We would like to ask you to participate in this evaluation. The results of this evaluation will be used to improve the next version of this module.

7 Appendixes

7.1 Coding Conventions

As Information Technology Emmen, we use certain code conventions within Java.

These can be found in the repository below:
https://github.com/NHL-Stenden-Emmen/coding-conventions

7.2 Scoring Rubrics

Criteria	Weighing	Sub Weighing	0-24%	25-54%	55-74%	75-100%
Diagrams portfolio consisting of: - *Klassendiagram* - *Use Case Diagram* - *Activity Diagram* - *Sequence Diagram*		O/V	The diagrams do not meet the UML 2.5 specifications.	The diagrams do not cover the entire case. There are more than ten errors in them. These include incorrect relationships, missing cardinality, missing use cases, incorrect use of symbols, etc.	The diagrams do not cover the entire case. There are up to ten errors in them. These include incorrect relationships, missing cardinality, missing use cases, incorrect use of symbols, etc.	The diagrams cover the entire case, comply fully with the UML 2.5 specifications and contain a maximum of five errors.
		O/V				
Coding Conventies	20%		More than ten errors were found involving code conventions.	Between five and ten errors were found in use of the code conventions.	Up to five errors were found in the use of code conventions.	The code fully conforms to the supplied code conventions.
Code Quality	50%	100%	The program crashes or does not compile (0%) or there is duplicate code or it was programmed in an inefficient way.	Duplicate code is present. Programming is done in an inefficient way. Hardcoded strings are used when this could be solved differently.	The code is reusable. No duplicate code is present. Programming is done in an efficient manner. The SDK is put to good use whenever possible.	The code is reusable. Methods have one responsibility. There is no duplicate code present. Programming is done in an efficient manner. The SDK is used usefully whenever possible. The correct loops are used.
Testting	30%		No Unit tests have been written or the Unit test(s) written do not cover all scenarios.	The written Unit test(s) cover only the happy paths and/or cover less than 60% of the code (line coverage).	The written Unit test(s) cover between 60% and 80% of the code (line coverage), excluding accessors and mutators.	The written Unit test(s) cover more than 80% of the code (line coverage), excluding accessors and mutators.
	100%	**100%**				
Total		**100%**				

NHL STENDEN hogeschool