JAVATNS - Java, The Next Step
Modulebook

2022-2023
Version 1.17 / 29-06-2022

NHL STENDEN hogeschool

Java, The Next Step

Version 1.17
29-06-2022

Academic year 2022-2023

Module coordinator	Teachers
Martijn Pomp martijn.pomp@nhlstenden.com	Martijn Pomp martijn.pomp@nhlstenden.com Niels Doorn niels.doorn@nhlstenden.com

Preface

This module follows on seamlessly from the module Introduction to programming in Java. In this module you will be extending your knowledge of Java programming. This addition to your knowledge will include new collections, classes, inheritance, etc., but you will also acquire skills for testing your code. As in the module Introduction to programming in Java, you will be doing this by studying a book but also and in particular by carrying out assignments. This is because programming is a skill that you can master only by practising it intensively.

The primary purpose of this module is not just to learn Java, but also to learn how to divide tricky, abstract problems into manageable parts. The Java programming will take place in the development platform IntelliJ or any development platform of your choice. After completing this module, the student will have laid a firm basis for getting off to a good start with the final Java programming module and the project in the second block of year 2.

In your future professional field, you as an IT professional will come across an object-oriented language such as Java in a similar way.

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

From the various methods used at NHL Stenden University of Applied Sciences, it has emerged in the past that the method used in the module "Java, the next step" is the most suitable method for a programmer who already has knowledge of PHP to learn Java. Student evaluations have led to the programme being adapted in such a way that more time is available for programming in Java.

Martijn Pomp & Niels Doorn
Emmen, June 29, 2022

Content

1 Introduction

Object-oriented programming is one of the basic elements of today's computer scientist. In this case object-oriented programming will be given in one of the most used languages: Java

Typical professional situation

Paul has been working for 1 year at ***Pure Home Luxury*** as a software engineer. Paul has graduated in Computer Science at Higher Professional Education level, during which he gained a lot of knowledge of programming and working in projects.
He has also followed various courses to learn more about how to make user-friendly computer interfaces.

During the past year Paul has been a full member of the ***EasyCare*** project, which has designed domotic systems for healthcare institutes. He worked within this project as a junior software engineer. His most important contribution was to develop the software on the basis of functional requirements.

Paul is able to find the right balance between existing and new technology. He makes an accurate assessment of the financial and legal implications of his work and product. He works together with colleagues (from various disciplines) and communicates with colleagues and clients.

Paul is able to work both independently and in a team. He is able to communicate with the various parties concerned. He has broad technical knowledge and is easily able to relate to what the customer wants. He is able to quickly familiarize herself in various areas and has mastery of various design methods, programming languages and development tools to develop software.
Paul will be able to develop in the mid-term into a senior software engineer, and in the long-term into a software development manager.

1.1 Competences

In this module the entry level professional works on three competencies involving the analysis, design and realisation at level 1.

	Manage & Control	Analysis	Advise	Design	Realisation
User interaction					
Business processes					
Infrastructure					
Software		> Collect and validate functional requirements for a software system with one stakeholder according to a standard method. > Define acceptance criteria for functional requirements stated above. **(level 1)**		> Create a design for a software system, including a data base with model techniques according to a standard method. **(level 1)**	> Build, test and make available a simple software system. The set-up, filling and querying of a data base is part of the software system. **(level 1)**
Hardware interfacing					

1.2 Module theme

There is no central theme in this module, and the module "Java, the next step" is a standalone element.

1.3 Module goals

After completing this module, the student will be able to:

> Apply subjects covered in chapters 6 to 10 of the BlueJ book in Java programs.
> select and apply the most suitable (storage) collection;
> analyse an object-oriented case;
> apply the functionality and operation of objects and classes in the individual programming assignments;
> select and apply the most suitable test and apply it to a Java program (fragment);
> develop a simple object-oriented program based on set criteria;
> explain the meaning of cohesion, coupling, code duplication, encapsulation, inheritance and polymorphism;
> set up a process to develop a Java program;
> program in Java.

1.4 Prior knowledge

The student must have completed the module "Introduction to programming in Java" (IPJAVA).
The student must meet the test requirements of the module "Introduction to programming in Java" in order to take part in the module "Java, the next step".

1.5 Conventions

Conventions regarding to naming and structuring code you can find on Blackboard within ICT-general.

1.6 Version management

Version	Date	Author	Description
1.0	05-10-2010	B. Meijerink	Initial document
1.1	25-08-2011	B. Meijerink	Assessment and supervision hours adjusted.
1.2	16-11-2011	B. Meijerink	Resit time adjusted.
1.3	05-06-2012	B. Meijerink	Adapted to the new academic year.
1.4	24-09-2012	B. Meijerink	Adapted to the new academic year.

Version	Date	Author	Description
1.5	08-11-2012	B. Meijerink	Resit for tutorial changed
1.6	08-07-2013	B. Meijerink	Adapted to the new academic year.
1.7	18-08-2014	B. Meijerink	Adapted to the new academic year.
1.8	18-08-2015	B. Meijerink	Adapted to the new academic year.
1.9	23-11-2015	B. Meijerink	Hbo-i competence matrix adapted to 2014 version
1.10	18-04-2016	B. Meijerink	Sign of assignments only during tutorials. Adapted to the new academic year.
1.11	07-03-2017	B. Meijerink	Adapted to the sixth edition of the book; Adapted to the new academic year.
1.12	22-05-2018	B. Meijerink	Adapted to the new academic year.
1.13	22-08-2018	B. Meijerink	Adapted to NHL Stenden.
1.14	04-09-2019	J. Doornbos	Adapted to the new academic year.
1.15	31-08-2020	M. Pomp	New format; Adapted to the new academic year.
1.16	27-08-2021	J. Doornbos	Adapted to new academic year; Adapted to HBO-I 2018; Added Scoring Rubrics.
1.17	29-06-2022	J. Doornbos	Adapted to new academic year; Textual fixes in end assignments.

2 Tests

An individual final test is used to assess whether you have achieved this module's goals.

Tabel 2.1 provides an overview of the final test with relevant information about the pass mark, points and credits.

Tabel 2.1 Test overview

Method	Max. points	Pass mark %	Pass mark in points	Credits	Deadline	Resit
Final assignment	100	55%	55	3 EC	Week 8	
Total	100	55%	55	3 EC	Week 8	

2.1 Assessment of the individual final assignment

All of the assignments from the module *Java, the next step* must be discussed **and signed off.** before the student is allowed to start the individual final test.

If it becomes clear during the discussion that the student has not completed the assignment himself, the assignment will be given a mark of 1.0 (0 points for all scoring rubrics).

A replacement assignment to which the same conditions apply will be offered once for the resit of the individual assignment.

The individual final assignment must contain all the steps laid down in appendix B3.2 of the module **Introduction to programming in Java** (the first Java module).

The individual final assignment will be assessed based on an assessment form (Scoring Rubrics) as included in appendix 7.6.

This module has been successfully completed once a pass mark has been awarded for the individual final assignment by week 8 of the module period at the latest.

2.2 Active participation

The student must complete all assignments preceding the individual final assignment in the relevant weeks. The student will have the opportunity to have these assignments signed off during the scheduled supervised tutorials.

2.3 Module resit

The resit opportunities for the tutorials are laid down in chapter 2.4.

Students who fail to complete the individual final assignment by week 8 at the latest can take a resit. The resit will be held in week 9 of the module period.

3 Programme

One of the most found professions in the IT area is software engineer. Software engineers (but other IT people as well) need a solid grounding in (object-oriented) programming languages.

As a Java programmer you will (usually) be working in a team of experienced Java programmers. You will be jointly responsible for the maintenance and continued development of existing packages and will help to build and test new applications for both internal and external use.

This module contains several weekly tutorials. During the tutorials the student will have the opportunity to discuss problems that arise with the assignments with the lecturers and to have the assignments signed off.

The process will be concluded with a practical test.

3.1 Lecture types

Below different lecture types are described.

3.1.1 Tutorials

The idea is for the student to prepare for the tutorials by means of individual study and to ask questions on that basis. The tutorials are also used to monitor the student's progress.
During the tutorials the student can have parts of the accompanying week signed off. There can of course be situations where a student has not yet finished a week's work and can therefore have it signed off during the next week. That does not however mean that the student may have to complete a large amount of work at the end of the period.
Attendance of the tutorials is compulsory. The lecturer will be available for parts of the tutorials.

3.2 Programme overview

An overview of the weekly activities is given below.

Week	Task. Nr.	Study activity
1	3.3.1	Introductory lecture.
	3.3.2	Tutorial 1 (Book chapter 6).
2	3.3.2	Tutorial 2 (Book chapter 6).
3	3.3.3	Tutorial 3 (Book chapter 7).
4	3.3.4	Tutorial 4 (Book chapter 8).
5	3.3.4	Tutorial 5 (Book chapter 8).
6	3.3.5	Tutorial 6 (Book chapter 9).
	3.3.7	Tutorial 6 (Java, The Next Step assignment). Start document assessment.
7	3.3.6	Tutorial 7 (Book chapter 10).
	3.3.7	Tutorial 7 (Java, The Next Step assignment).
8	3.3.7	Tutorial 8 (Java, The Next Step assignment). Application code assessment.
9		Resit.

3.3 Weekly programmes

3.3.1 Introducty lecture

Week	1
Work form	Lecture
Duration	1 hour.
Teaching aims	> The student is given an overview of the content of the module Java, the next step; > The student understands the method and assessment operated in the module Java, the next step.
Content	During the introductory lecture you will be given instruction on subjects including the method/work forms, assessment, materials and content of the module Java, the next step.
Preparation	
Individual assignments	

3.3.2 Tutorial 1 + 2 (Book chapter 6)

Week	1 + 2
Work form	Tutorial
Duration	2 hours each.
Teaching aims	> The student gains an understanding of library classes. > The student learns to read and write documentation.
Content	During this tutorial the student will be able to work on chapter 5 of the book.
Preparation	
Individual assignments	Study and complete assignments 6.1 to 6.38, 6.42 to 6.45, 6.48 to 6.84, 6.88 and 6.89.

3.3.3 Tutorial 3 (Book chapter 7)

Week	3
Work form	Tutorial
Duration	2 hours.
Teaching aims	> The student learns how to work with fixed-size collections.
Content	During this tutorial the student will be able to work on chapter 7 of the book. There will also be an opportunity to complete chapter 6.
Preparation	
Individual assignments	Study and complete assignments 7.1 to 7.18, 7.22 to 7.35, 7.37 to 7.45.

3.3.4 Tutorial 4 + 5 (Book chapter 8)

Week	4 + 5
Work form	Tutorial
Duration	2 hours each.
Teaching aims	> The student learns to design classes.
Content	During this tutorial the student will be able to work on chapter 8 of the book. There will also be an opportunity to complete chapter 7.
Preparation	

Individual assignments	> Study appendix 1;
	> Study the information up to chapter 8.7 and
	> complete assignments 8.1 to 8.7;
	> Study appendix 2;
	> Complete assignment 8.8;
	> Study the information up to chapter 8.15 and
	> complete assignments 8.9 to 8.14;
	> complete assignments 8.16 to 8.21;
	> study appendix 3;
	> complete assignments 8.22 to 8.25;
	> study appendix 4;
	> complete assignments 8.28 to 8.33;

3.3.5 Tutorial 6 (Book chapter 9)

Week	6
Work form	Tutorial
Duration	2 hours.
Teaching aims	> The student learns how to test Java programs; > The student learns how to debug Java programs;
Content	During this tutorial the student will be able to work on chapter 9 of the book. There will also be an opportunity to complete chapter 8.
Preparation	
Individual assignments	> Study the information up to chapter 9.5 and > complete assignments 9.1 to 9.20; > Study the information from chapter 9.9 up to chapter 9.11 and > complete assignments 9.22 to 9.35; > Study the information from chapter 9.12 to chapter 9.14.

3.3.6 Tutorial 7 (Book chapter 10)

Week	7
Work form	Tutorial
Duration	2 hours.
Teaching aims	> The student learns what inheritance is; > The student learns what polymorphism is;
Content	During this tutorial the student will be able to work on chapter 10 of the book. There will also be an opportunity to complete chapter 9.
Preparation	
Individual assignments	Study and complete assignments 10.1 to 10.19.

3.3.7 Tutorial 6 – 7 – 8 (Individual programming assignment)

Week	6 – 7 – 8
Work form	Tutorial
Duration	2 hours each.
Teaching aims	> The student is able to independently implement a larger programming assignment. > The student is able to independently analyse a given assignment; > The student is able to independently formulate a list of requirements; > The student can independently formulate a UML class diagram consisting of at least 3 classes > The student can independently implement this class diagram in a Java program.
Content	> During this tutorial the student will be able to ask questions about the individual programming assignment; > During this tutorial the student will be able to have the various steps signed off; > During this tutorial the student will be able to work actively on the program.
Preparation	
Individual assignments	Carry out the individual assignment as indicated in appendix 7.5.

4 Structure & Organisation

The chart below provides an overview of all contact hours in this module.

The students are also expected to plan their own (project) meetings at which they can work on the assignments. This applies also to the time the students need to prepare and complete (individual) assignments. The chart also provides a clear overview of the anticipated study workload per student.

Tabel 4.1 Student contact hours (SCH) and Study Workload Hours per week:

Method of working	Week 1		Week 2		Week 3		Week 4		Week 5		Week 6		Week 7		Week 8		Week 9		Total	
	SCH	SWH	SCH	SWH	SCH	SWH	SCH	SWH	SCH	SWH	SCH	SWH	SCH	SWH	SCH	SWH	SCH	SWH	SCH	SWH
Lecture	1	3	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	3
Tutorial	2	6	2	6	2	6	2	6	2	6	2	6	2	6	2	6	0	0	16	48
Final Test	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Self Study	0	0	0	3	0	3	0	3	0	3	0	3	0	3	0	3	0	0	0	21
Total study activities	3	9	2	9	2	9	2	9	2	9	2	9	2	9	2	9	0	0	17	72
Extra curriculair activities																			0	12
End total																			17	84

SCH	= Student Contact Hours (45 minuten)
SWH	= Study Workload Hours (60 minuten)

5 Literature / software

Literature and software that is necessary to fullfill this coarse.

5.1 Compulsory teaching materials

You can decide whether to use the Dutch or English version of the book ***Objects First With Java*** You therefore only need to order one of these two books.

> This module.

> English Book APA:
Barnes,D. J; Kölling, M (2017)
Objects First With Java (A Practical Introduction Using BlueJ), **Sixth Edition.**
Upper Saddle River: Pearson Education

ISBN ordering details

ISBN-10:	1-292-15904-9.
ISBN-13:	978-1-292-15904-1.

> Dutch Book APA:
Barnes,D. J; Kölling, M (2017)
Programmeren in Java met BlueJ (Een 'objecten eerst'- benadering), **Zesde Editie.**
Benelux: Pearson Education

ISBN ordering details

ISBN-13:	978-90-430-3499-9.

5.2 Reference materials

> Java website
https://www.oracle.com/nl/java/

> Java 13 documentation
https://docs.oracle.com/en/java/javase/13/docs/api/index.html

5.3 Software

> IntelliJ IDEA
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 contains all parts of the module, including organisational aspects, content, quality of the teaching staff, etc.

We kindly request that you take part in this evaluation. Its results will be used to improve the next edition of this module.

7 Appendices

7.1 Additional teaching materials for assignments 8.3 and 8.4

In exercises 8.3 and 8.4 you are asked to develop own game scenario. Because this is a rather time-consuming task we have already done it for you. From this point we will be developing a game that we have called Evacuate the house.
We define a house with rooms. Objects can be placed in these rooms (in a later exercise). The player can pick up these objects and put them somewhere else. The player can carry objects up to a certain weight. The idea is that the player empties the entire house and places the objects outside taking as few actions as possible.
We have already done some groundwork: On the electronic learning environment you will find a project with the name zuul_8_3-4_student under the Course Documents for "Java, the next step". This project is derived from zuul-bad and some changes have been made only in the Game class.
This project only contains part of a house because we cannot move upstairs or downstairs.

7.2 Additional teaching materials assignment 8.8

In this assignment we change the game in such a way that we can move upstairs and downstairs. We have already done some groundwork: In the electronic learning environment you will find a project with the name zuul_8_8_student under the Course Documents for "Java, the next step" in which only the game class is found.

You will need to copy the classes ***Parser, Command, Commandwords and Room*** into this.

The following is laid down in exercise 8.8: Implement the changes described in this section in your own zuul project”. **These changes have been made in the Game class, but not yet in the other classes.** You will have to do this yourself.

7.3 Additional teaching materials for assignment 8.22

In this assignment the room can contain a number of objects. We have already done some groundwork: In Microsoft Teams you will find a project with the name zuul_8_22_student with Java code under the Course Documents for "Java, the next step". Copy the content of this text file in the method *createRooms* from the class *Game*.

7.4 Additional teaching materials for assignments 8.28 t/m 8.33

For these exercises it is helpful to start by listing all of the possible commands that the user can give:

go
Go via the indicated exit to the next room.
back
Go back to the previous room.
pick up
Pick up the indicated object from the room the player is in.
leave
Leave the indicated object in the room the player is in.
stop
End the game
help
show the current assignments
room
give all information about the room the player is in (exits, objects, ...)
player
give all information about the player (including objects he is carrying)
object
give all information about the indicated object

7.5 Individual programming assignment

7.5.1 Procedure

You will already have completed an individual programming assignment in the section "Introduction to programming in Java" in the first year. In "Java, the next step" you carry out another individual assignment that covers the knowledge, skills, etc. that you have acquired in this module.

The method is the same as that used with the programming assignment of the section "Introduction to programming in Java". If necessary, reread appendices 3 and 4 from the module "Introduction to programming in Java".

For this final assignment it is compulsory for the student to use at least one useful test class.

The choice of classes is of great importance in the "Java, the next step" assignments. At least two different classes must be defined for each assignment. This might seem a bit over the top with various assignments. But remember that it's not about the complexity of the assignment but about practising with object-oriented programming. In the design aspect a lot of attention has to be paid to making a logical choice of classes and interfaces.

In this case the lecturer will designate an assignment from those listed below. The assignments are the same as those issued with "Introduction to programming in Java". But there are two requirements:

> You have to do a different assignment from that in "Introduction to programming in Java". The lecturer will allocate an assignment, but the student has to say which assignment he has previously carried out;

> You must carry out the assignment with a different collection from the assignment you have already done in "Introduction to programming in Java". (You will probably have chosen an Array list and now you will have to choose a Hashmap, Hashset, etc.)

7.5.2 Assignments

1. Volleyball
The Dutch volleyball union asked you to develop a Java application for registering scores of a volleyball tournament. National and international teams are competing against each other in the tournament. Every match consists of two teams who play against each other. The match is won by the team who scores 25 points first. There should be a point difference of two with the other team also. So it is possible to have a match result of 38 – 36; but also 25 – 20.
The application should return all the necessary information per match: the team who won, the team that lost, the score of each team and the difference in points. When a national team wins, it earns 3 tournament points. International teams earn 2 tournament points when they win.
It should be possible to add one point at a time or start with predefined points.
It should also be possible to get the winner of the tournament. This is the team with the most tournament points.
For publishing the results on their website, the volleyball union wants to have an overview of the tournament, with the results.

2. Spotify
The Spotify company asked you to develop a lite version of the Spotify application. Spotify is the most used music stream service now a days. The application should consists of the player itself, songs, artists and playlists. When a song is played 5 times it should be added to the 'Most played' playlist. This playlist has a maximum of 10 songs. The user should be able to add songs to a custom playlist. The custom playlist has a name that the user can specify. It should be possible to retrieve the total playtime (hh:mm:ss) of a playlist. When the user selects a song the client should show which song is playing, the duration of the song (mm:ss) and the artist. Premium users have the possibility to download some songs. Not all songs are downloadable. When a song is downloadable it gives a download link and also tracks the amount of downloads. The player should be able to return the most downloaded song.

3. Netflix
Netflix wants a new simple application which only allows the user to watch series. A serie is divided in multiple seasons and every season contains one or more episodes. The user of the application should be able to start an episode. When the episode is started it should show the duration of the episode. The user should also be able to rate an episode: a thumbs up or down. Some episodes are sponsored. A sponsored episode counts one thumbs up as two, so Netflix earns extra money. Netflix should show the most interesting episode per serie and also the most boring one. For binge watchers Netflix wants to show the complete time in hh:mm:ss for a complete season and a complete serie.

4. Digital test
To save the environment, NHL Stenden wants to digitalize their testing method. The user should be able to add questions to a test. There are two types of questions: multiple choice and open questions. A multiple choice questions is worth one point, an open question three. When all the questions are added to the test the test can be started. After completion of the test the score should

be displayed. When the score is more than 55% the student passed the test. The end result is shown in the following way: Your score is: 56%. You PASSED!! If the result is insufficient the following appears on the screen: Your score is 54%. You FAILED!! The percentage that is printed is variable and is depending on the actual results.

5. Prison

The prison currently has all the files about its prisoners on paper. They want to digitalize all their files. The prison has several cells. There are two types of cells: a small one and a large one. The small cell can hold one prisoner, a large cell two. The cell is identified by a cell number. The guards should be able to see which cells currently have a free spot and which prisoner is in which cell. The application should also contain information about the prisoner, e.g. the date of the offence (dd-mm-yyyy) and how many years he has to stay in prison.
The guards of the prison want to know which prisoner will be released first.

6. Bicycle rental

The NS Group wants new software to track the rental of their bicycles. The software should track how many bicycles are still available for rental. There are three types of bikes: a regular one, mountain bike and electrical bike. A customer can hire a bicycle for a certain amount of time. The end time will be registered when the customer returns the bike. In every bicycle is a GPS-tracker. The tracker tracks the distance the customer has travelled with the bicycle. When the customers starts renting he has to pay deposit from €20,-. The customer has to pay money per kilometre. This is:
> €0,20 for a regular bike;
> €0,25 for a mountain bike;
> €0,50 for a electrical bike.
Per hour the customer has to pay €2,-. When the customer returns the bike, the customer has to pay. For maintenance purposes the company wants to know the total distance the bike has been used.

7. Soccer club

FC Emmen, the local soccer club of Emmen, wants to register the amount of fouls of the players of a team in an application. Every team has at least 11 players. The soccer club has more teams. They want to track the amount of yellow, red and black cards for each player. The player has to pay a fine for each card he gets. The fine for a yellow card is € 18,32. The fine for a red card is € 41,60. The fine for black card is €349,76. You'll also get disqualified and are not allowed to play for the team again. FC Emmen wants to know how much money they have to pay to the National Soccer Association. They player with least amount of cards gets a Fairplay Award. This should be printed to the screen.

8. Athletics Club

De Sperwers, the local athletics club from Emmen, currently has a big paper administration of all the PR's and distances of their athletes. To get rid of all the paper work they want a simple application to register all the data. For every athlete they want to see how many rounds the athlete has run in which time. There are different tracks available: a grass track of 300m and a gravel track of 400m. To compensate the difficulty of the grass track the times run on this track will be reduced by 5 seconds. The club also wants to know the club record per track: the athlete who is the fastest on the certain track. For each athlete should be calculated:

> The total distance in km.
> The average time per lap.
> The max speed the (m/s).
> The total time the athlete has run.

The club wants to display the fastest athlete and his/her fastest laptime.

9. PostNL

PostNL is THE company to get packages and letters to customers. PostNL wants healthy postmen so they want an application to measure the max amount of items the postman can carry. Every postman may carry a maximum weight of items. This weight is different for every postman. The maximum weight a postman can carry is calculated by multiplying the weight of the postman by 5. A package has a variable weight. A letter has a default of 2 grams.
The Depot should know if they need to hire extra postmen so they are able to deliver all of the packages. The Depot tracks the amount of packages and letters delivered per postman per day and shows the name of the postman and the amount of packages and letters that he has delivered.

10. Pizza Delivery Service

One of the biggest companies that sells pizza needs an application to track their expenses. Every deliverer has to ride one or more routes a day with a predefined distance. Due to environmental issues, there are two options for delivering: by bike or by scooter. Per kilometre the costs per scooter are €0,12, per bike €0,02. The salary per day of the deliverer is calculated with the following formula: $age\ of\ the\ person \cdot 1{,}50 - 2{,}50$. In Euros. The application should give an overview from all the expenses per month. And an overview of the most earning pizza deliverer.

7.6 Scoring Rubrics

		Max points	0-25%	25-50%	50-75%	75%-100%
Start document	Input/output	4	Problem definition lacks detail.	More than half of the input, output, calculations and screen output are described, including format descriptions.	Most input, output, calculations and screen output are described, including format descriptions.	The precise form of the input, output, calculations and screen output are described, including format descriptions.
	Class diagram	8	The class diagram lacks detail and/or is not compliant with the UML specification.	The UML diagram contains all the classes. All classes contain most of the fields and methods. The types of some of the fields are correctly specified. The arguments of some of the methods are correctly specified. The visibility (public, protected, private, package private) of some fields and methods are defined. The UML class diagram is in accordance with the UML specifications.	The UML diagram contains all the classes. All classes contain most of the fields and methods. The types of the fields are correctly specified. The arguments of the methods are correctly specified. The visibility (public, protected, private, package private) of most fields and methods are defined. The UML class diagram is in accordance with the UML specifications.	The UML diagram contains all the classes. All classes contain all fields and methods. The types of the fields are correctly specified. The arguments of the methods are correctly specified. The visibility (public, protected, private, package private) of all fields and methods are defined. The UML class diagram is in accordance with the UML specifications.
	Testplan	8	Test plan is missing or lacks detail.	Test plan is mostly complete. Some boundary values are included.	Test plan is mostly complete. Some boundary values are included. Also negative testing is included.	Test plan is complete. All boundary values are included. Also negative testing is included.
	Subtotal	20				
Code	Comments	10	No comments.	Too much text in comments, unnecessary comments, bad styling.	Some comments, but with bad styling.	Neat and meaningful comments that makes sense. In JavaDoc.
	Coding Conventions	5	No code conventions at all. Code alignment is not consistent, variable names that don't make sense.	A lot of misalignments in code. Incorrect use of white space (lines). A lot of names for variables, methods and classes don't make sense.	Some misalignments in code. Names of variables, methods and classes are correct.	Code is aligned perfectly, understandable. Correct use of whitespace. Correct names are given to variables, methods and classes.
	Functionality/efficiency	20	There are unnecessary fields and methods. A lot of duplicate code. Methods are not atomic.	There are unnecessary fields and methods. There is also duplicate code in the application.	There are no unnecessary fields, but there are some unnecessary methods (or vice versa).	There are no unused fields and methods (with disregards of mutators and accessors). Also no duplicate code and unnecessary methods.
	Unit Tests	20	Almost no Unit Tests are created.	Some Unit Tests are created, but they are not meaningful.	Some classes contain Unit Tests.	All classes contain meaningful Unit Tests.
	Specifications	5	The application does not comply with the assignment at all.	Program does not comply with all the specifications.	Some of the specifications are not in the application.	The application fully complies with the assignment.
	Subtotal	60				

		Max points	0-25%	25-50%	50-75%	75%-100%
Coding on the fly		20	The student is not able to finish the assignment. Has no clue on how to approach the assignment.	The student knows how to handle the problem, he can describe it in pseudocode / words, but is not able to finish it in time	The student has written the assignment in code, but it is not fully done.	The student is able to fulfil the assignment alone without any help of the teacher / Internet in a efficient way.
	Subtotal	20				
	End total	**100**				

NHL STENDEN
hogeschool