HOGESCHOOL ROTTERDAM / CMI

Development 3

INFDEV02-3
2015-2016

Number of study points:	4 ects
Course owners:	Youri Tjang & Giuseppe Maggiore

Module description

Module name:	Development 3
Module code:	INFDEV02-3
Study points and hours of effort:	This module gives 4 ects, in correspondence with 112 hours: • 2 X 3 x 6 hours of combined lecture and practical • the rest is self-study
Examination:	Written examination and practicums (with oral check)
Course structure:	Lectures, self-study, and practicums
Prerequisite knowledge:	INFDEV02-1 and INFDEV02-2.
Learning materials:	• Book: Think Java; author A. B. Downey (www.greenteapress.com/thinkjava) • Book: Head First Java (2nd ed.); authors K. Sierra, & B. Bates. (2005). • Slides: found on N@tschool and on the GitHub repository github.com/hogeschool/INFDEV02-3 • Exercises and Assignments, to be done at home and during practical part of the lectures (pdf): found on N@tschool and on the GitHub repository github.com/hogeschool/INFDEV02-3
Connected to competences:	realiseren en ontwerpen
Learning objectives:	At the end of the course, the student: • **is able to use** and **create** interfaces and abstract classes. `(ABS)` • **has developed skills** to adopt a new programming language with little support. `(LEARN)` • **is able to apply** the concepts of data encapsulation, inheritance, and polymorphism to software. `(ENC)` • **is able to apply** the concepts of data encapsulation, inheritance, and polymorphism to software. `(ENC)` • **understands** basic human factors. `(BHF)`
Course owners:	Youri Tjang & Giuseppe Maggiore
Date:	February 7, 2016

1 General description

Programming is one of the most ubiquitous activities within the field of ICT. Many business needs are centered around the gathering, elaboration, simulation, etc. of data through programs.

1.1 Relationship with other teaching units

Subsequent programming courses build upon the knowledge learned during this course.

The course analyis 3 covers UML, which is often used to demonstrate concepts in this course.
Knowledge acquired through the programming courses is also useful for the projects. A word of warning though: projects and development courses are largely independent, so some things that a student learns during the development courses are not used in the projects, some things that a student learns during the development courses are indeed used in the projects, but some things done in the projects are learned within the context of the project and not within the development courses.

2 Course program

The course is structured into six lectures. The six lectures take place during the six weeks of the course, but are not necessarily in a one-to-one correspondance with the course weeks. For example, lectures one and two are fairly short and can take place during a single week.

2.1 Chapter 1 - statically typed programming languages

Topics

- What are types?
- (**Advanced**) Typing and semantic rules: how do we read them?
- Introduction to Java and C# (**advanced**) with type rules and semantics
  - Classes
  - Fields/attributes
  - Constructor(s), methods, and static methods
  - Statements, expressions, and primitive types
  - Arrays
  - (**Advanced**) Lambda's

2.2 Chapter 2 - reuse through polymorphism

Topics

- What is code reuse?
- Interfaces, abstract classes and implementation
- Implicit vs explicit conversion
- (**Advanced**) Implicit and explicit conversion type rules
- Runtime type testing

2.3 Chapter 3 - reuse through generics

Topics

- Using generic parameters
- (**Advanced**) Using covariance and contravariance in the presence of generic parameters
- (**Advanced**) Designing interfaces and implementation in the presence of generic parameters

2.4 Chapter 4 - architectural considerations

Topics

- Encapsulation
- Input controllers
- State machines

2.5 Chapter 5 - yet more architectural considerations

Topics

- (**Advanced**) Composition versus inheritance
- (**Advanced**) Entity/component model

3 Assessment

The course is tested with two exams: A series of Assignments which have to be handed in, but won't be graded. There will be an Oral check, which is based on the Assignments and a written exam. The final grade is determined as follows:

`if Theoratical exam-grade` $>= 5.5$ `then return Oral check-grade else return 0`

Motivation for grade A professional software developer is required to be able to program code which is, at the very least, *correct*.
In order to produce correct code, we expect students to show: *i*) a foundation of knowledge about how a programming language actually works in connection with a simplified concrete model of a computer; *ii*) fluency when actually writing the code.
The quality of the programmer is ultimately determined by his actual code-writing skills, therefore the written exam will contain require you to write code, this ensures that each student is able to show that his work is his own and that he has adequate understanding of its mechanisms.

3.1 Theoretical examination INFDEV02-3

The general shape of a Theoratical exam for `INFDEV02-3` is made up of a series of highly structured open questions. In each exam the content of the questions will change, but the structure of the questions will remain the same. For the structure (and an example) of the theoretical exam, see the appendix.

3.2 Practical examination INFDEV02-3

There are 2 Assignments which are mandatory, and formatively assessed for Feedback.

- All assignments are to be uploaded to N@tschool or Classroom in the required space (Inlevermap or assignment);
- Each assignment is designed to assess the students knowledge related to one or more Learning objectives. If the teacher is unable te assess the students' ability related to the appropriate Learning objective based on his work, then no points will be awarded for that part.
- *The teachers still reserve the right to check the practicums handed in by each student, and to use it for further evaluation.*
- The university rules on fraude and plagiarism (Hogeschoolgids art. 11.10 – 11.12) also apply to code;

Glossary

ABS

The Learning objective stating that at the end of this course: the student **is able to use** and **create** interfaces and abstract classes.

Assignment

A large-sized task to be completed by a student in order to get a grade. Assignments are related to learning objectives. Students are expected to show their skill and understanding of those learning objectives within an assignment.

BHF

The Learning objective stating that at the end of this course: the student **understands** basic human factors.

ENC

The Learning objective stating that at the end of this course: the student **is able to apply** the concepts of data encapsulation, inheritance, and polymorphism to software.

Exercise

A small- to medium-sized task to be completed by a student in order to gain experience, understanding and feedback.

Feedback

"... feedback is conceptualized as information provided by an agent (e.g., teacher, peer, book, parent, self, experience) regarding aspects of one's per- formance or understanding. A teacher or parent can provide corrective information, a peer can provide an alternative strategy, a book can provide information to clarify ideas, a parent can provide encouragement, and a learner can look up the answer to evaluate the correctness of a response. Feedback thus is a "consequence" of performance." [1]

LEARN

The Learning objective stating that at the end of this course: the student **has developed skills** to adopt a new programming language with little support.

Learning objective

A brief description of what students are expected to learn during the course. In general, each course has between 1 and 10 learning objectives.

Oral check

The summative assessment at the end of the course, determining the final grade.

Theoratical exam

The summative assessment at the end of the course. Scoring >= 5.5 is a strict condition for being rewarded credit points. (Dutch: tentamen).

TYPE

The Learning objective stating that at the end of this course: the student **can apply** the concepts of data types.

[1] Hattie, J., & Timperley, H. (2007). The Power of Feedback. Review of Educational Research, 77(1), 81—112.

Appendix 1: Assessment matrix

	Dublin descriptors
ABS	1, 2, 4
LEARN	1, 4, 5
ENC	1, 2, 4
TYPE	1, 2, 4
BHF	1, 2, 4

Dublin-descriptors:

1. Knowledge and understanding
2. Applying knowledge and understanding
3. Making judgments
4. Communication
5. Learning skills