SCI2011: Introduction to Programming, Period 1, 2017

Course offered by Department of Data Science and Knowledge Engineering

All information in this manual is preliminary and may be subject to updates during the execution of the course.

1. Teachers

Lecturer (Coordinator)	Dr. Gerasimos (Jerry) Spanakis
Email:	jerry.spanakis@maastrichtuniversity.nl
Physical address:	Department of Knowledge Engineering Sint Servaasklooster 39, Room 2.001
Teaching Assistants (to be confirmed)	Simon Craenen Chen Xi

2. Course material and resources

Literature: The following two books are available for free download. However, please note that course slides (uploaded at portal) cover all that you need to learn. Use slides as a guide and refer to the textbooks (or any other source you find on the Web) when needed.

- A. Downey & C. Mayfield, Think Java: How to Think like a Computer Scientist, Green Tea Press, 2012.
  http://greenteapress.com/wp/think-java/ **(ABD)**

- David J. Eck, Introduction to Programming Using Java, Sixth Edition, 2011.
  http://math.hws.edu/javanotes/ **(DJE)**

Additional textbooks:
- Robert Sedgewick, Kevin Wayn: Introduction to Programming in Java: An Interdisciplinary Approach
- Stuart Reges, Marty Stepp: Building Java Programs: A back to Basics Approach
- Bruce Eckel: Thinking in Java

Websites:

- Courses Portal: You will find all slides, labs and assignments
- Socrative Environment: https://b.socrative.com/student/ (Room is AIRPLANE)
- Java resources (see intructions on portal on how to download and install Java and IntelliJ IDEA to your laptop or desktop):
  http://www.oracle.com/technetwork/java/javase/downloads/index.html
  https://www.jetbrains.com/idea/download/

- Video tutorials:
  http://freevideolectures.com/Course/2513/Java-Programming/

3. Learning goals – Aim of the course

The course assumes absolutely no background in computer science, programming, or mathematics. Students who complete the course will:

1) Identify, interpret and apply fundamentals of programming such as variables, conditionals, iteration, etc.
2) Identify, interpret and apply fundamentals of object-oriented programming, including defining classes, invoking methods, using class libraries, etc.
3) Give examples of important topics and principles of software development.
4) Point out obvious mistakes in programs and analyze how they run.
5) Design, compose and evaluate programs that solve specific problems.
6) Use a software development environment to create, debug, and run programs.

4. Course outline (rough)

Note that the following table is a rough plan only! Please check courses portal for the actual weekly schedule. Schedule changes will be announced in class as well.

Wk	**Date**	**Topic**	**Book sections +** ***Assignment info***
1	Sep. 6th	Basics of programming in Java	**ABD:** 1, 2.1-2.5, 3.1-3.2 **DJE:** 1, 2.1-2.2, 2.6
2	Sep. 13th	Part I: Data: Variables, Types, Representation Part II: Conditional statements	**ABD:** 2, 3.1-3.5, 3.8, 3.10, 4.1-4.2, 4.8-4.9, 5.1-5.5, 9.4-9.6 **DJE:** 2.5-2.6, 3.1.1, 3.1.3-3.1.4, 3.2, 3.5-3.6, 4.5 *-Announcement of assign. #1*
3	Sep. 20th	Part I: Loops Part II: Methods	**ABD:** 4.3-4.7, 5.6, 6.1-6.3, 6.5, 7, 9.8 **DJE:** 3.1.2, 3.3-3.4, 4.2-4.4, 4.7
4	Sep. 27th	Part I: Arrays Part II: Methods revisited	**ABD**: 8.1-8.8, 9.3, 9.9 **DJE:** 3.8, 7.1,7.2, 7.5, 5.1 *-Delivery of assign. #1* *-Announcement of assign. #2*
5	Oct. 4th	Objects and Classes	**ABD:** 10.1-10.9, 11.1-11.8 **DJE:** 5.2-5.6, 5.8 *-Announcement of assign. #3*
6	Oct. 11th	Object-Oriented Programming Principles Wrap-up	**ABD:** 6.4, 12.1-12.3, 12.6-12.7, 12.9, 14.1-14.7 **DJE:** 9.1, 7.4, 8.1-8.2 *- Delivery of assign. #2*
7	Oct. 18th	:(Exam 1600-1830 (Tapijn)	*- Delivery of assign. #3*

5. Course format and composition

The course will consist of a weekly (interactive) lecture, a tutorial lab and a work lab.

Preparation: It is recommended (but it actually depends on your understanding) to study the book chapters before the lecture. It will increase understanding and help you grasp the contents of the lecture and the aftermath tutorial. If you decide to skip the lecture, please do study the book chapters and/or slides before coming to the tutorial and lab.

Lecture: During lectures you will be instructed in the basics of programming via slides and hands-on programming examples. If you possess a laptop, please take it with you as playing around with the examples and the programming environment increases understanding. During the lecture you might be asked to provide a short answer through the socrative application (either via webpage or via the app you can download on your mobile), the link is https://b.socrative.com/ and the room is AIRPLANE.
* If you possess a laptop, please take it with you to the lectures, tutorial and the lab. Playing around with the examples (not facebook!) and the programming environment increases understanding.

Tutorial: After the lecture, you will practice basic programming principles through examples and algorithms (using your laptop will help you!). The goal of the tutorial is to build the foundation needed to start writing your own programs, given the content of each week's lecture. Be sure that you are able to understand, explain and practice all the objectives stated in the course composition (see below) and answer the questions, problems that are distributed by your tutor. Tutorial acts as a bridge between the lecture and the lab session. During tutorials you will design problems on paper, explore how programs run and implement simple examples using the computer. You may be asked to work in pairs and help each other.
* There is no point in coming to a tutorial unprepared. Find your way through the course either by studying the material beforehand (lecture slides or book sections) and/or by actively attending the lecture.

Lab: Labs are practical programming sessions. You will be asked to solve one (or more) short problem(s) using things learned in the lecture or/and the tutorial. If your program solves the problem, you will earn full marks for the lab. If you attend the lab (for its entire duration) and attempt the problem, you will earn at least half marks for the lab. The tutor will provide assistance, but the goal of the lab is to discover how to do everything -you already learned during the lecture and the tutorial- by yourself. At the end of the lab (or other tutor-defined deadline) you have to upload your program in order to claim your grade.
* If you are absent from a lab (or deliver nothing by the end of it) you obviously get 0 points for that lab. If you are absent from a lab you can work from home and submit it according to the deadline. In that case you get half points (and gain priceless experience in programming!).
* For the tutorials and the labs there is a minimum attendance requirement of 85%, which amounts to 10 out of 12 sessions. However, it is strongly recommended not to skip any of the lectures, tutorials and labs since the amount of effort that you will have to spend studying by yourself to catch up is over 100% and with unpredictable outcome.

Assignments: Assignments will be announced during the period (see Section 4) and must be completed individually. Assignments are due at 23:59 on the date listed on the assignment (usually two weeks after the announcement).

Week 1
Reading: See course outline

Lecture 1: The first lecture will provide a general introduction to Computer Science and programming., e.g., What are we going to learn during this course? What is Java? How do we install it? How to write basic code? How do we run a program?

Tutorial 1: a) Explain and differentiate the compilation and execution of Java programs
b) Identify and give examples syntax, logic, semantic, runtime errors
c) Summarize the basic syntax, semantics and rules of Java language
d) Distinguish and apply concepts like: keywords, reading/printing numbers, declaring/using variables
e) Identify the features of Java as a strongly typed language

Lab 1: You will be required to write your first program and compile it. Debugging your errors will start at this point! Afterwards you will be asked to write a simple program that makes a simple computation.

Week 2
Reading: See course outline

Lecture 2: The second lecture will provide the basic building blocks for constructing a more complex program. You will learn more about variables, types and conditionals.

Tutorial 2: a) Interpret different primitive types in Java (int/double/char/boolean)
b) Variables: declaration and initialization
c) Explain type compatibility and type casting: automatic and user triggered (infer when type casting is necessary)
d) Differentiate between String and char “types”
e) Illustrate operation precedence in examples
f) Search with API java (already implemented methods Math, Scanner, String)
g) Solve practical problems regarding logical expressions and conditionals (decisions)
h) Debug programs (how to be sure that my program runs OK or what I do when it doesn’t :))

Lab 2: You will solve two small problems (one scientific, one real-life).

Week 3
Reading: See course outline

Lecture 3: The third lecture handles loops and methods. Loops allow us to make iterations easier and methods are used to make your program modular and reusable. Debugging of your programs (a crucial task in programming) will also be discussed.

Tutorial 3: a) Interpret loops in Java (while/do-while/for) and relate one with each other.
b) Infer the three parts of the loop (initialization, termination condition, iteration).
c) Identify and analyze the properties of a variable such as its associated value, its scope and its lifetime
d) Illustrate parameter passing for simple variables (*by-value*).
e) Organize code in methods: how to define them (type/name/arguments) and how to call them.
f) Give examples of types of methods

g) Illustrate arguments of methods: type and name
h) Differentiate between argument/parameter
i) Summarize what happens in program flow when a method is called
j) Solve practical problems involving iteration
k) Organize your program in methods, analyze their dependencies, construct fully-defined methods, call them.

Lab 3: You will solve two small problems (one scientific, one real-life)

Week 4
Reading: See course outline

Lecture 4: The fourth lecture describes arrays and the basic idea behind creating our own data structures. Arrays are lists of variables and are often used when a program requires multiple instances of similar data. Loops can be used to iterate over arrays.

Tutorial 4: a) Give examples when arrays are needed in programming
b) Distinguish between array index and array content
c) Define arrays (type, indexing, naming, size)
d) State operations that can involve arrays (i.e. loop through every element of the array)
e) Interpret representation of arrays in Java and how this affects arrays when they are arguments in methods (*pass-by-reference*)
f) Explain the concept "Classes before objects".
g) Illustrate how to define an object and differentiate from using it
h) Illustrate how to call: constructor, modifiers, methods
i) Solve practical problems with arrays

Lab 4: In this lab you will practice programming in solving a simple simulation program with arrays and methods.

Week 5
Reading: See course outline

Lecture 5: Lecture 5 introduces the main concepts of Object Oriented Programming (OOP). Classes represent the concept of a template, e.g., a human. An Object is an instantiation of such a template, e.g., you or any other student. Objects use everything that we have already learned. It does however introduce a different way of thinking about your program, a different way of structuring, and organizing everything we have learned up until now. You will be able to start declaring your own classes & objects.

Tutorial 5: a) Differentiate between class and object
b) Design and implement a class: essential parts: member variables(fields), constructor(s) and behavior (methods)
c) Construct an object using a class and activate methods on it
d) Differentiate between object declaration and initialization
e) Illustrate that constructors are different than the original methods (in what terms)
f) Illustrate how many constructors can we have in a class
h) State the name used for the constructor methods
i) Give examples of ***private*** variables and why do we use them
k) Explain and differentiate the use of ***static*** keyword for classes and for methods
p) Differentiate between variables of primitive types and variables of class types
s) Design solutions to practical problems by defining classes and constructing objects

Lab 5: You will solve a practical problem by creating your own classes and objects.

Week 6

Reading: See course outline

Lecture 6: The final lecture will build up on OOP and allow you to create your own classes and how you can utilize them by creating objects. Concepts of encapsulation, inheritance, polymorphism and overloading will be introduced. Moreover, we are scaling things into describing how you can build more complex structures, like arrays of objects. Searching through these structures is also discussed It will provide a roundup of everything we have learned during this course.

Tutorial 6: a) Describe **Encapsulation** and its necessity
b) Explain **Inheritance, Overriding, Polymorphism,** state their advantages and recommend examples
c) Illustrate ***this*** and ***super*** keywords
d) Identify super- and sub- classes in a class hierarchy
e) Recognize and analyze overridden and inherited methods in a class hierarchy
f) Design solutions with arrays of objects to solve real-life problems.

Lab 6: Practice some aspects of object-oriented programming principles and search in arrays of objects.

Week 7

Written exam. Open books and open notes. You can bring whatever you want with you (notes, books, slides, assignments, etc.) but you will not be allowed to exchange anything during the exam.

...but why an exam? Usually in the real-life of a programmer a computer and a compiler are available. This technical (practical) part is evaluated throughout the period and accounts for 50% of your grade (6 labs/3 assignments). But programming is not just about making something work and run (and that's why the course is called Introduction to Programming and not Introduction to Java).

The final exam aims at checking whether you developed that specific way of thinking, required for *effective* and *efficient* problem solving. There will be questions of asking the basics, questions about finding obvious errors in small programs, questions about explaining why a program runs (or not) and some questions asking you to develop some *algorithmic* code (for the latter no emphasis will be given to absolute correct syntax but emphasis is given to whether your algorithm *is correctly designed* and covers all *implementation details*). Emphasis is given on the *concepts* and the "behind the scenes" *understanding*.

6. Evaluation (grading scheme and policies)

Students are evaluated on their performance on:

Practical part: Practical part consists of (six) weekly labs and (three) assignments. The six labs count for 20% of the final grade and the three assignments count for 30% of the final grade (10% each).

Final exam (individual & open books/open notes): The maximal amount of points for the exam is 100. Exam grade counts for 50% of the final grade.

The final grade will be constructed in the following way:

$$\boxed{\textbf{final grade} = 0.2 \times \textbf{labs} + 0.3 \times \textbf{assignments} + 0.5 \times \textbf{final exam}}$$

Resits

In order to be eligible for the resit, students must have met the attendance requirement as described in Section 5 (Course format and composition). In this case there is a possibility of a written exam (open books/open notes), same format as the final exam. Final grade is computed with the above formula. Note however that there is no possibility to redo the assignments or the labs.

7. Honor Policy

● Unless otherwise specified, the only allowed (but also strongly suggested and promoted) collaboration for the assignments and labs is the discussion of implementation ideas
● No code or solutions are to be distributed to other students either electronically (i.e. e-mail) or on paper or posted online where they can easily be discovered (i.e. Facebook, webpages, forums, discussion groups, etc.)
● Unless otherwise noted, assignments are pledged: you promise that you have neither given, nor received unauthorized help.
● By submitting your assignment, you agree that you are the sole author of your code. If you have used a piece of code that helps you in solving a lab problem or an assignment (e.g. a very, very smart algorithm that you found online) you should clearly state it in your code comments, as well as the source of it.
● When there is doubt regarding the honorability of an action, you need to ask before doing it.
● Please do not copy your assignments/labs and/or do not try to cheat! It is very easy for your grader to understand when a piece of code is not yours! Any honor violation or cheating will not be tolerated!

□