# Algorithms and Data Structures (ICS0005)

## 6 ECTS, exam

**The prerequisite for this course is knowledge of fundamentals of programming in Java**

**Lecturer:** **Jaanus Pöial**, PhD

https://enos.itcollege.ee/~japoia/

# Schedule

**Lectures – every week**

**Labs – every second week (see SIS)**

**7 homeworks in Moodle using Java**

https://moodle.taltech.ee/course/view.php?id=18143

https://enos.itcollege.ee/~japoia/algorithms/

Textbook: Michael T. Goodrich, Roberto Tamassia. Data Structures and Algorithms in Java. John Wiley and Sons, Inc.

# Requirements

- Homework deadlines are strict (penalties: -25% for delay up to one week and -60% for more than a week) because labs are based on homework. At least 50% of automated tests must pass to meet the deadline, improvements are possible within one week.
- Feedback to homework is given in Moodle (within three working days) and it must be „ok“ before attending the lab and **all** automated tests must pass without exceptions.
- Labs are mandatory. The content of the labs is the further development of homework in pairs (on site).

- Current „official“ score is visible in SIS, grades are entered into the SIS after the lab. Grades in SIS can be transferred in case of retaking the course.
- Homework 6 is mandatory, the report must be defended in the lab.
- Plagiarism is not tolerated, if sources have been used, they must be referenced (for programs add comment to the beginning of code)
- Using AI is allowed for coding, but highly discouraged for reporting. You are solely responsible for any content generated with AI. Usage of AI must be referenced (which tool was used for which purpose)

# Grading

- The prerequisites for taking the exam are:
  - at least 25 points from homework and labs
  - homework 6 defended
  - registration for the exam through SIS
- The grade is calculated based on the total number of points, there is no separate threshold for the exam:
  - 50% from homework and labs (at least 25p required)
  - 50% from exam. The exam will take place in the TTU computer classroom in the Moodle environment without the use of any supporting materials (net closed). Time 150 min.
- The exam may be retaken once within the offered exam times (3 times are offered), in which case the grade for the last attempt will be taken into account.

# Materials

- SIS – links and passwords for study materials, grades (interim results), assessment criteria
- Course homepage in Web (static)
- Moodle course ICS0005, forum, homework. Source of current information about the course. Deadlines are measured and feedback to homework is given in Moodle.
- Code repositories in Bitbucket used during the labs
- Lecture recordings
- Textbook and other books
- Suggested software tools and reading, tests, examples

# Software

- Java JDK (any version >= 8).
- JUnit 4
- Git
- IntelliJ Idea

https://www.oracle.com/java/technologies/java-se-support-roadmap.html

For Moodle submissions: Project Language Level 8

# Topics

- Algorithms, properties of algorithms, time complexity and space complexity, asymptotic behaviour of functions, analysis of algorithms, main complexity classes
- Searching and sorting. Linear search, binary search, hashing. Insertion sort, binary insertion sort, quicksort, merge sort, counting sort, radix sort, bucket sort. Complexity of different methods
- Abstract data types. Stacks and queues. Reverse Polish Notation (RPN). Linked lists and other pointer structures

- Trees. Examples of trees - arithmetic expression tree. Traversal of trees - preorder, postorder, in-order. Forms of expression for trees - parenthetic expressions, pointer structures, textual representation. Examples of "top-down" and "bottom-up" traversal
- Graphs. Directed and undirected graphs, multigraphs. Cyclic and acyclic graphs. Connected components. Strongly connected and weakly connected graphs. Paths and cycles. Simple graphs. Matrices related to graphs: adjacency matrix, matrix of shortest pathlengths. Operations on graphs: sum, multiplication, transitive closure. Representation and implementation of graphs. Algorithms on graphs: Floyd-Warshall (lengths of shortest paths), topological sort of vertices, depth-first and breadth-first traversal, Dijkstra (shortest path), finding connected components, minimal spanning tree

- Recursion: Hanoi towers, elimination of recursion, tail recursion. Exhaustive search: 8 queens problem, knapsack problem
- Binary search trees, AVL trees, red-black trees, binomial trees. B-trees. Heaps, heapsort
- String algorithms: exact matching (linear, Knuth-Morris-Pratt, Boyer-Moore, Rabin-Karp)
- Coding and compressing, coding schemes: Huffman, Shannon-Fano. Greedy algoritms: Huffman encoding
- Dynamic programming: longest common subsequence problem

- Correctness of algorithms: preconditions, postconditions, loop invariants, weakest precondition, structural rules, total correctness and partial correctness, halting problem
- Exam