2023Year 2nd Semester Syllabus

Created Date	2023-08-04 10:36:24		Last-Modified	2023-08-07 15:01:21
Course Title	COMPILER DESIGN		Course Code-Section	CSI4104-01
Credit/Time/ Experiment,Lab,Practical Technique Time	3/Tue8,9,Thu7		Department	Computer Science
Time	Tue8,9,Thu7		Location	EngHD504
Exam Date & Time	Midterm exam		Final exam	
Class Language	English		Evaluation Type	Absolute evaluation

Instructor's Profile	Name	Burgstaller bernd	Contact Information	Telephone	02-2123-5728
	Department	DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING		Mail	BBURG@YONSEI.AC.KR
	Office	Engineering Hall 4, D910		Interview information	Zoom consultations upon appointment via email

TA's Name & Contact Information	Name		Contact Information	Telephone	

Course Description Brief Introduction of the Course	Please note: this is the undergraduate course on compiler design. It does not share any content with the graduate course CSI8105-01 on ``Advanced Compiler Construction”. Broadly speaking, the undergraduate course comprehensively discusses programming language design patterns (for lexing, parsing, semantic analysis, code generation and language run-times), while the graduate course is dedicated to compiler optimizations. The undergraduate course is recommended as a prerequisite for the graduate-level course. Overview A compiler is a computer program that translates text written in a given language (called the source language) into another language (the target language). With most compilers the source language is a high-level programming language (e.g., C, C++, Java), and the target language is a lower-level representation such as assembly language or bytecode. In this course we will focus on compiler techniques needed to implement programming languages on a virtual machine. In a series of five assignments, students will implement a compiler that translates a subset of C into Java bytecode. This bytecode can then be executed on the Java virtual machine (JVM).

Course Goals	1.	Korean	.	40%
		English	This course will cover both practical and theoretical aspects of a compiler. Our main emphasis will be on the compiler frontend (i.e., scanning, parsing, semantic analysis) and code-generation for the JVM.	
	2.	Korean	.	30%
		English	Crafting a compiler involves the use of algorithms and data structures. Software engineering principles need to be applied when conducting a reasonably large, object-oriented compiler project. We will employ Java packages, subtype polymorphism and dynamic dispatching.	
	3.	Korean	.	30%
		English	Two software-engineering topics will be covered for developing our compiler: Exception handling in Java, and the Visitor design pattern. This material will be interspersed with the regular lectures, in due	

			time for the assignments.	
	4.	Korean		0%
		English		
	5.	Korean		0%
		English		

Core Competencies	The total measurable competencies must be 100%. Each course objective should set the competency as 25%. The core and major competencies should equal at least 50%.		
Sub-Competencies/Learning Unit1			
Sub-Competencies/Learning Unit2			
Sub-Competencies/Learning Unit3			
Core Competencies(Liberal Arts)Major competency(	**Must reflect the interrelationship between core competencies (elective courses) and major competencies (major studies).**		
	This course addresses the foundations of compiler construction. Compilers receive programs written in a high-level programming language as input. A study on compilers is therefore strongly related to programming language syntax and semantics. Compilers employ fundamental models from language theory, such as regular expressions, finite state automata and context-free grammars. Regular expressions are used for pattern matching in scripting languages (e.g., with Python, Perl, and JavaScript) and with many operating system shells, command-line utilities (grep, sed, find) and text editors. Automata are useful for performing keyword searches in text documents, e.g., with web queries. Context-free grammars are used for the definition of programming languages and with XML document type definitions.		
Sustainable Development Goals			

Average Recommended Amount of Learning per	Average Reading Volume		Average amount of writing(Based on A4)	

Course Methods (%) Total Amount 100	Lecture	Practice Training	Presentation	Dabate	Team Project
	100%	0%	0%	0%	0%
Course Methods 2 Select Relevant Items	PBL Subject	Capstone Design	CBL, Social Innovation Course	Flipped Classroom	Work Experience,Internsh

Grading Policy(%) Total Amount 100 Free Input for Other Information	Midterm exam	Final exam	Quiz	Individual Assignment	Team Assignment	Attendance	Others
	26%	26%	0%	43%	0%	0%	5%

Assignment/ Report, Project Guide	Title of Assignment/Project Name, and Method of Filling Out	Submission Deadline	Type of Submission and Method
Prerequisite	The following courses are pre-requisistes : Automata theory and formal languages , Data structures, Programming Languages, Object-oriented Programming, Computer Programming	Online Course Address	

Course Material	Course Material Name	Author	Publisher	Publish Year	ISBN

Main Learner Precautions	Target students: Fourth year (senior) students in Computer Science. Students from other majors are welcome subject to the availability of seats (preference will be given to majors in Computer Science). This course is open to exchange students. The course is limited to 95 students, which is the maximum seat capacity of our designated lecture room, D504. Because of the volatility of the course registration (bidding) process, all estimates on the likelihood for entering the course are unsound. Enquiries about this matter therefore will not be answered. Course requirements: The implementation language for this course is Java. Familiarity with Java or a related object-oriented programming language is required for the assignments, although this requirement is somewhat mitigated through the use of code skeletons that will be provided with the assignments. The assignments will be implemented, tested, and submitted on Linux; basic familiarity with the Linux command line and SSH is required (but a Linux quick-start guide will be provided if Linux is new to you). Grading policy: - 26.25% midterm exam - 26.25% final exam - 5% homeworks - 37.5% assignments: - 5% active classroom participation All homeworks and assignments are individual assignments. Homeworks and assignments should be turned in before or at the due date. When turned in late, 5% will be deducted from the grade per day until the deliverable has been received, with a maximum extension of five days. Assignments are marked based on an automated test script. Students will be provided with a representative set of testcases, but not all testcases used for grading will be made public. Students are encouraged to create their own testcases, to ensure that an implementation works correctly. Creating good test cases is part of the learning process of this course. Mastering the assignments is a requirement for the exams. Students are expected to apply the techniques acquired with the assignments to exam problems. The course language is English, including lectures, assignments, homeworks and exams. The course uses absolute grading according to the following grading table (in %): A: 100-80 B: 79-70 C: 69-60 D: 59-50 F: 0-49
Attatchment	

Weekly Plan

week	Period	Weekly Topic & Contents	Remarks
1	2023-09-01 2023-09-07	Introduction, Lexical Analysis	(9.1.) Fall semester classes begin (9.5. - 9.7.) Course add and drop period
2	2023-09-08 2023-09-14	Regular Expressions	
3	2023-09-15 2023-09-21	Automata	
4	2023-09-22 2023-09-28	Context-free Grammars	09.28 추석

5	2023-09-29 2023-10-05	Recursive Descent Parsing, Exception Handling for Parsing Errors	(9.28. - 9.30.) 추석연휴 (10.3.) National Foundation Day 09.29 추석, 09.30 추석, 10.03 개천절
6	2023-10-06 2023-10-12	Abstract Syntax Trees	(10.8.) First third of the semester ends (10.9.) Hangul Proclamation Day 10.09 한글날
7	2023-10-13 2023-10-19	Attribute Grammars	
8	2023-10-20 2023-10-26	Midterm Exam	(10.20. - 10.26.) Midterm Examinations
9	2023-10-27 2023-11-02	Visitor Design Pattern	(10.27. - 10.31.) Course withdrawal period (11.1. - 11.3.) Application Period for S/U evaluation
10	2023-11-03 2023-11-09	Type Checking	
11	2023-11-10 2023-11-16	Run-time Environments	(11.14.) Second third of the semester ends
12	2023-11-17 2023-11-23	JVM and Java Bytecode	
13	2023-11-24 2023-11-30	Java Bytecode Generation, Java Class File Format	
14	2023-12-01 2023-12-07	AST Interpreters, Bytecode Interpreters, Threaded Code Execution	
15	2023-12-08 2023-12-14	Self-study week	(12.8. - 12.14.) Self-study
16	2023-12-15 2023-12-21	Final exam week	(12.15. - 12.21.) Final Examinations

• Students with disabilities(SWDs) can request accommodations related to lectures, assignments, or tests by contacting the course professor at the beginning of semester.
(However, accommodations may vary depending on the essentiality of lecture and discretion of professors.)

[Lecture]
- Visual Impairment: alternative, braille, enlarged reading materials, note-taker
- Physical Impairment: alternative reading materials, access to classroom, note-taker, assigned seat
- Hearing Impairment: note-taker/stenographer, recording lecture
- Intellectual Disability/Autism: note-taker

[Assignments and Test]
- Visual/Physical/Hearing Impairment: (reasonable) extra days for submission, alternative type of assignment, extended test time, alternative type of test, arranging separate test room, and proctors, test ghostwriter
- Intellectual Disability/Autism: (reasonable) extra days for submission, alternative type of assignment