CS 3360: Design and Implementation of Programming Languages
Fall 2015

CRN: 12500
Lecture: TR 9:00-10:20 am in CCSB 1.0202
Website: http://www.cs.utep.edu/cheon/cs3360/
Instructor: Yoonsik Cheon (x-8028, ycheon@utep.edu); office hours: TR 10:30-11:50 am in CCS 3.0606
TA: TBA
Prerequisite: CS 2302 with a grade of C or better

Course Goals

In this course we will study concepts and examples of programming languages with the goal of acquiring the tools necessary for critical evaluation and rapid mastery of programming languages and constructs.

The course attempts to balance theory and hands-on experience. We will survey the constructs and capabilities typically found in modern programming languages with attention to design trade-offs and implementation considerations. By gaining an understanding on the range of possibilities likely to be encountered in a language, students will be prepared to learn new languages quickly throughout their careers. By understanding the implications of design alternatives, students will be better able to anticipate the problems likely to arise in using a new language. Also, the presentation of design alternatives and trade-offs lays the groundwork for future advanced study of compilers and programming language semantics. To instantiate the discussion of general programming language characteristics, several languages will be presented in more detail: e.g., AspectJ (an aspect and object-oriented language), Haskell (a functional language), Prolog (a logic-programming language), and PHP (a Web scripting language). Students will gain practical experience with each programming paradigm by completing a programming project in each of the chosen languages.

Upon successful completion of this course, students will be able to:

- [Level 3: *Synthesis and evaluation*] Evaluate modern, representative programming languages critically
- [Level 3] Choose a suitable programming paradigm and language for a given problem or domain
- [Level 3] Design a small, domain-specific programming language, e.g., a test script language, by defining its formal syntax and semantics
- [Level 2: *Application and analysis*] Define syntax of a small context-free grammar in BNF and EBNF
- [Level 2] Use attribute grammars to describe the static semantics of small programming languages
- [Level 2] Define dynamic semantics of small subsets of programming languages, e.g., control structures
- [Level 2] Select and apply appropriate expressions and control structures for a given programming task
- [Level 2] Analyze and evaluate data and control abstractions of programming languages
- [Level 1: *Knowledge and comprehension*] Recognize major programming languages
- [Level 1] Explain operational semantics, axiomatic semantics, and denotational semantics as methods of expressing programming language semantics
- [Level 1] Explain design concepts, design alternatives and trade-offs, and implementation considerations for scope, binding, data types, expressions, control structures, subprograms, abstract data types, objects, concurrency structures, and exception handling in modern programming languages

Texts

The course textbook is Robert W. Sebesta, *Concepts of Programming Languages*, 11th edition, Addison Wesley, 2015. The textbook is available at the UTEP bookstore, and you are expected to acquire a copy for your use in this course, as reading assignments will be taken from the textbook:

Assignments

There will be three types of assignments: reading, non-programming and programming. All assignments will be handed out or announced in class. If you miss a class session, it is your responsibility to find out what you missed. You should expect to spend about 3-4 hours per week for reading and non-programming homework assignments, and an average of 3 hours per week for programming assignments. Note, however, that each programming assignment is estimated to require 8-10 hours, so your work load in weeks that programming assignments are due may be higher (with other weeks being correspondingly lower).

Reading and non-programing homework assignments ask you to read the textbook and prepare for the coming week's lectures. Exercises that use material not yet discussed in lecture will be graded generously. For the non-programming homework assignment, **no late submission will be accepted** unless an arrangement has been made in advance or unless an unusual circumstance warrants an exception.

Programming assignments are designed to allow you to gain a hands-on experience with a specific language and programming paradigm (e.g., AspectJ for object and aspect-oriented programming, Haskell for functional programming, PHP for Web and procedural programming, and Prolog for logic programming). Late submissions will be accepted, but **those turned in late will be penalized 10% for each day or partial day of lateness** for up to five days. After five days, submissions will not be accepted unless other arrangements have been made in advance or unless unusual circumstances warrant an exception.

All assignments are to be done individually. While you may discuss the assignment in general terms with others, your solutions should be composed, designed, written and tested by yourself alone. If you need help, consult the TA or the instructor.

Exams

There will be two mid-term exams and one final exam. The final exam will be comprehensive. The mid-term exams will take place during the regular class session and each will be 80 minutes in length, and the final exam will take place on the date specified by the university.

Grading

Your semester grade will be based on a combination of homework assignments, programming projects, quizzes, and exams. The approximate percentages are as follows:

Homework and quizzes:	35%
Programming assignments:	30%
Exams:	35%

In addition, a bonus of up to 5% is available for lecture attendance and participation. To earn this bonus, you must arrive at lecture on time and participate in class discussion in a constructive and prepared manner, e.g., by asking or answering questions that demonstrate that you have read and attempted to understand the material.

The nominal percentage-score-to-letter-grade conversion is as follows:

90% or higher is an A
80-89% is a B
70-79% is a C
60-69% is a D
below 60% is an F

I reserve the right to adjust these criteria downward, e.g., so that 88% or higher represents an A, based on overall class performance. The criteria will not be adjusted upward, however.

Attendance

Lecture attendance is required; you should understand that your success in the course will improve greatly by attending classes regularly. The following is excerpted from the 2014-2015 Undergraduate Catalog.

> The student is expected to attend all classes and laboratory sessions. It is the responsibility of the student to inform each instructor of extended absences. When, in the judgment of the instructor, a student has been absent to such a degree as to impair his or her status relative to credit for the course, the instructor can drop the student from the class with a grade of W before the course drop deadline and with a grade of F after the course drop deadline.

Standards of Conduct

You are expected to conduct yourself in a professional and courteous manner, as prescribed by the UTEP Standards of Conduct. Graded work (homework, projects and exams) is to be completed independently and should be unmistakably your own work, although you may discuss your work with others in a general way. You may not represent as your own work material that is transcribed or copied from another source, including persons, books, or Web pages. Instructors are required to—and will—report academic dishonesty and any other violation of the Standards of Conduct to the Dean of Students.

Disabilities

If you have a disability and need classroom accommodations, please contact The Center for Accommodations and Support Services (CASS) at 747-5148, or by email to cass@utep.edu, or visit their office located in UTEP Union East, Room 106. For additional information, please visit the CASS website at www.sa.utep.edu/cass.

Schedule

The following table shows a planned schedule for the course; refer to the course website for an up-to-date schedule.

Dates		Topics	Readings	Assignments
Week 1	Aug. 25, 27	Introduction Describing syntax	Chap 1 Sec 3.1-3.3	
Week 2	Sep. 1, 3	Describing syntax Attribute grammar	Sec 3.4	Homework 1
Week 3	Sep. 8, 10	Object-oriented programming Aspect-oriented programming	Sec 12.1-12.6 Handouts	Homework 2
Week 4	Sep. 15, 17	AspectJ		
Week 5	Sep. 22, 24	AspectJ Names and binding	Sec 5.1-5.3	Lab 1
Week 6	Sep. 29, Oct. 1	Type and storage binding Type checking and scopes	Sec 5.4 Sec 5.6-5.10	Homework 3
Week 7	Oct. 6, 8	Review for exam 1 **Exam 1**		
Week 8	Oct. 13, 15	Data types	Sec 6.1-6.9	Homework 4
Week 9	Oct. 20, 22	Functional programming Haskell and Hugs	Sec 15.1-15.3 Sec 15.8, handouts	
Week 10	Oct. 27, 29	Haskell		Lab 2
Week 11	Nov. 3, 5	Describing semantics **Exam 2**	Sec 3.5	
Week 12	Nov. 10, 12	Logic programming and Prolog	Chap 16	
Week 13	Nov. 17, 19	Prolog Web scripting and PHP	Handouts	
Week 14	Nov. 24, 26	PHP Thanksgiving		Lab 3
Week 15	Dec. 1, 3	Subprograms; review for final	Sec 9.1-9.6	
Week 16	Dec. 8	**Final** at 10:00 am – 12:45 pm		

Important Dates

August 24:	Class begins
September 7:	Labor Day – university closed
September 8:	Census day
October 8:	Exam 1
October 30:	Course drop deadline
November 5:	Exam 2
November 26-27:	Thanksgiving holiday - university closed
December 4:	Dead day
December 8:	Final on Tuesday at 10:00 am – 12:45 pm