

# CS 3360: Design and Implementation of Programming Languages

## Spring 2011

CRN: 23998

Lecture: TR 1:30-2:50 pm in CS 322

Website: <http://www.cs.utep.edu/cheon/cs3360>

Instructor: Yoonsik Cheon (x-8028, ycheon@utep.edu); office hours: TR 3:00-3:50 pm in CS 202B

TA: Melisa Vela (x-8949, smvelaloya@miners.utep.edu); office hours: W 3:00-5:00 pm in CS 128

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*, 9th edition, Addison Wesley, 2009. 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 two kinds of assignments: homework assignments and programming projects. 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 on the homework assignments, and an average of 3 hours per week on programming projects. Note, however, that each programming project is estimated to require 8-10 hours, so your work load in weeks that projects are due may be higher (with other weeks being correspondingly lower).

Homework assignments may ask 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 homework assignments, **no late submission will be accepted** unless arrangements have been made in advance or unless unusual circumstances warrant an exception.

Programming projects are designed to allow you to gain 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 project submissions will be accepted, but **projects turned in late will be penalized 10% for each day or partial day of lateness** for up to five days. After five days, projects will not be accepted unless other arrangements have been made in advance or unless unusual circumstances warrant an exception.

All homework assignments and programming projects 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 you 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 50 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:	25%
Programming projects:	25%
Exams:	50%

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 not mandatory but is recommended. You should understand that your success in the course will improve greatly by attending class regularly. It is your responsibility to keep up to date with notes, assignments and exam.

### **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, 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 or suspect a disability and need accommodations, you should contact The Student Disabled Services Office (DSSO) at 747-5148. You can also email the office at [dss@utep.edu](mailto:dss@utep.edu) or go by the Union Building East, Room 106. For additional information, visit the DSSO website at [www.utep.edu/dsso/](http://www.utep.edu/dsso/).

## 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	Jan. 18	Introduction Describing syntax	Chap 1 Sec 3.1-3.3	
Week 2	Jan. 25	Describing syntax Attribute grammar	Sec 3.4	Homework 1
Week 3	Feb. 1	Object-oriented programming Aspect-oriented programming	Sec 12.1-12.6 Handouts	Homework 2
Week 4	Feb. 8	AspectJ		
Week 5	Feb. 15	AspectJ Names and binding	Sec 5.1-5.3	Project 1
Week 6	Feb. 22	Type and storage binding Type checking and scopes	Sec 5.4 Sec 5.6-5.10	Homework 3
Week 7	Mar. 1	Review for exam 1 <b>Exam 1</b>		
Week 8	Mar. 8	Data types	Sec 6.1-6.9	Homework 4
Week 9	Mar. 16	Spring break		
Week 10	Mar. 22	Functional programming Haskell and Hugs	Sec 15.1-15.3 Sec 15.8, handouts	
Week 11	Mar. 29	Haskell		Project 2
Week 12	Apr. 5	Describing semantics <b>Exam 2</b>	Sec 3.5	
Week 13	Apr. 12	Logic programming and Prolog	Chap 16	
Week 14	Apr. 19	Prolog Web scripting and PHP	Handouts	
Week 15	Apr. 26	PHP Expressions and control structures	Chap 7-8	Project 3
Week 16	May 3	Subprograms Review for final	Sec 9.1-9.6	
Week 17	May 12	<b>Final at 1:00–3:45 pm</b>		

## Important Dates

January 17:	Martin Luther King, Jr. day (university closed)
January 18:	Class begins
February 2:	Census day
March 3:	Exam 1
March 14-18:	Spring break (no classes)
April 1:	Course drop deadline
April 7:	Exam 2
May 6:	Dead day
May 12:	Final on Thursday at 1:00–3:45 pm