CSCI 335 SYLLABUS

COURSE NAME, NUMBER	CSCI 335: Software Development
SEMESTER, YEAR	Spring 2010
INSTRUCTOR	T. VanDrunen
OFFICE / TELEPHONE / EMAIL	Armerding 112 752-5692 Thomas.VanDrunen@wheaton.edu
OFFICE HOURS	MWF 3:15–4:45 pm; Th 9:00-11:30 am
COURSE WEBSITE	`http://csnew.wheaton.edu/~tvandrun/cs335`

RESOURCES

Gamma et al, *Design Patterns*, Addison-Wesley, 1994.
Brooks, *The Mythical Man-Month*, Anniversary Edition, Addison-Wesley, 1995.

COURSE DESCRIPTION

Principles and practices of software development including design patterns, validation and testing, and coordination of team projects. Introduction to data bases and user interface design. Professional issues in computing.

INFORMAL DESCRIPTION

This course has two purposes: it covers the principles of software development (sometimes called *software engineering*), hence the name of the course; it also is a landing place for various topics that need to be covered in the CS core curriculum, but don't fit nicely in any other course. From the first purpose we have concepts and terms, UML, group dynamics, project and team management, methodology, software life cycle, etc); design patterns; professional and ethical issues; readings, including *The Mythical Man-Month* and "The Cathedral and the Bazaar"; tools such as Javadoc, JUnit, and makefiles; and some discussions about career path options. From the second purpose we have the basics of database use and design; user interface design, plus GUIs and event-driven programming; regular expressions, automata, and formal languages; and, time permitting, programming languages, distributed computing, web programming, scripting languages, and the history of the computing industry.

GOALS AND OBJECTIVES

1. Students will gain experience developing large software projects, including working on a software team.
2. Students will be able to understand and use the standard terminology and documentation tools for object-oriented software development.
3. Students will be able to articulate the fundamentals of database design and management, languages and automata, and language systems.
4. Students will be able to employ design patterns in their design of software systems.
5. Students will give thought to their career path and plan accordingly.

ASSESSMENT PROCEDURES

1. The group project will exercise students' abilities to work as a team and mark their progress in programming and design sophistication.
2. Short exercises and a presentation will measure students' participation.
3. The midterm and final will evaluate students' articulation of the concepts and terms.

Grading:

	weight
Project checkpoint 1	15%
Project checkpoint 2	20%
Project final submission	25%
Peer evaluation of project contribution	5%
Short exercises and participation	7.5%
Presentation	7.5%
Midterm	10%
Final	10%

SPECIAL EXPECTATIONS

Academic Integrity

Since all programming in this course will be group work, academic integrity concerns are not about code sharing among students. Instead, students must be careful to document and cite all ideas that come from outside sources. Failure to do so may result in point deduction or rejection of the project altogether.

Late assignments

If a programming project is running late, the group should negotiate with the instructor before the due date (ie, the negotiation should take place before the day on which it is due) to determine a revised due date and a point deduction. Short exercises will not be accepted late.

Attendance

Students are expected to attend all class periods. Because the your work in this class is a team effort, your absence harms not only your own experience in the class, but that of your classmates. When missing a class is unavoidable, it is a courtesy to inform the instructor, ahead of time if possible.

Special needs

Whenever possible, classroom activities and testing procedures will be adjusted to respond to requests for accommodation by students with disabilities who have documented their situation with the registrar and who have arranged to have the documentation forwarded to the course instructor. Computer Science students who need special adjustments made to computer hardware or software in order to facilitate their participation must also document their needs with the Registrar in advance before any accommodation will be attempted.

Examinations

This class has a midterm and a final. The midterm will be a take-home exam; the date and other specifics will be announced later. The final exam is Wednesday, May 5, at 1:30 AM. I do not allow students to take finals early (which is also the college's policy), so make travel appropriate travel arrangements.

Office hours

In the past I have tried to keep an "open-door" policy, that is, I don't mind if you stop by even when it is not my office hours. However, to help manage my load this semester, I've decided to reserve Tuesdays for uninterrupted work. Accordingly, *please to do not come to my office on Tuesdays without an appointment.* If you need help on a Tuesday, please make an "appointment" by sending me an email asking something like, "May I stop by and ask a question right now?" I will likely say yes. (Also, any time my door is closed, it means I'm doing something uninterruptable, such as making an important phone call. Rather than knocking, please come back in 5 minutes or send me an email.)

Project. Most of the work in this course will be on a group project in three phases. In the first pahse of the project, you will work in two teams of three. In the second phase, you will work in three teams of two, with each member of each pair coming from a different team from phase 1. In the third phase, you will all work together as a single team. Most of the grade for the project will be given in common to all members of the team; however, part of the evaluation of the final phase will involve teammates evaluating each other's contribution to the project. In total, the project will count for 65% of a student's grade in this course. Dates for completing the checkpoints are, approximately, Feb 19, Mar 26, and May 1.

Textbooks and readings. The two textbooks for this course are important texts which the students should plan to keep. Gamma et al is the standard source for the material covered in the design patterns theme of the course. Students should decide for themselves how they can use the text most effectively: some may choose to read the relevant sections prior to their presentation in class, some may choose to read them after, some may choose to skim first and read carefully after. But by all means students are responsible for the material in them, and mastery of the material will not be obtained without the combination of class presentation, textbook reading, and participation in related exercises.

The other required book, Brooks, will be the last topic covered in the course. Selections from this book must be read prior to class discussion on the material.

Library. In the bookcase of the computer science lab, there is a dedicated shelf of books providing additional resources for this course. You are referred to those books for fuller treatments of some of the topics touched on in this course.

Short exercises. On most class days you will be given a short exercise to reinforce the concept presented or discussed in that class period. Short exercises should be turned in by email (unless otherwise noted) before the start of the next class period. These exercises are primarily instructive, not evaluative. Your grades on these are mainly credit for doing the assignment, though a lack of effort or thoroughly incorrect answers will be penalized. Moderate collaboration on these exercises is permitted, though in exercises that require writing code, each student must turn in code written independently.

Presentation. Each student is required to make a presentation on one topic during the course of the semester. The student will present for about one half of the course period (or about 30 minutes). Each student must meet with the instructor twice prior to the presentation: A first meeting for the instructor to advise the student on resources, a second meeting for the student to show a detailed outline to the instructor and describe any slides, handouts, or related material he or she intends to use. Each student must choose a presentation topic no later than Friday, January 22. Topics available for presentation include life cycle models (Jan 25); project specifications (Feb 1); use cases (Feb 1); software architectures: layering (Feb 17), model/view/controller (Feb 19), client-server (Feb 19), pipe and filter (Feb 19); refactoring (Feb 22); eXtreme Programming (Feb 25); languages: Python (April 5), Scala (April 7), XML (April 12).

Schedule. See the course website for schedule. This schedule must be checked frequently in order to keep up with the readings and to find descriptions of short exercises.
The content stream of this course is the intertwining of several topics, each spread throughout the semester:

- Design patterns.

  A design pattern *is a reusable solution to a common problem or family of problems. Design patterns represent the collective wisdom of the object-oriented software development community. Gamma et al is a catalogue of the most widely recognized patterns. We will be examining many of these in class, and you are expected to find applications of them in the project. This is the largest topic of the course.*

- Practices and methodologies.

  Many course entitled "Software Development" or "Software Engineering" are primarily studies of methodologies and industrial practices. Due to time constraints, we will examine these in survey form. This will include workflows, agile methodologies, and industry standards.

- Tools.

  This course will survey many common tools in software development and management: automatic documentation production, revision control, IDEs, automatic testing frameworks, etc. The coverage of this material in class will not be sufficient for mastering any of these. You are responsible for researching and discovering tools and their features as needed in the project. These tools include

  - Javadoc
  - Subversion (SVN)
  - JUnit
  - Makefiles
  - Integrated Development Environments
  - Grep
  - Awk

- Programming languages.

  The Wheaton computer science major program currently does not have room for a required course on comparative programming languages (although we do have an elective on programming language concepts, CSCI 365). To mitigate that omission, CSCI 335 contains a module on the fundamentals of programming languages and a comparison of programming paradigms. This module includes

  - Formal languages and automata.
  - The structure of compilers and other language systems.
  - A comparison of the object-oriented and imperative paradigms (represented by Java and C, respectively) with the functional paradigm (ML) and the needs of languages tuned for the special purposes of scripting (Python) and web programming (PHP).

- User interfaces.

  We will spend about three days on the design of user interfaces, along with the related topic of event-driven programming.

- Databases.

  Similar to the module on programming languages, it has fallen to CSCI 335 to supply students in Wheaton's computer science major program with the core material in the field of data bases. We will spend about a week exploring the terminology and concepts of relational databases and a sample implementation of an interaction with databases in Java.

- Reflections on software development.

  We will spend a day talking about what a profession is and about software development as a profession; this will also give us an opportunity to talk about career paths in the field of computer science. We will conclude the semester by discussing an article (Eric Raymond, "The Cathedral and the Bazaar") and a book (Fred Brooks, The Mythical Man-Month*) about software design and development.*