CSCI 245 SYLLABUS

COURSE NAME, NUMBER	Programming II: Object-Oriented Design		
SEMESTER, YEAR	Spring 2007		
INSTRUCTOR	T. VanDrunen		
OFFICE / TELEPHONE / EMAIL	Armerding 112	752-5692	Thomas.VanDrunen@wheaton.edu
OFFICE HOURS	MWF 3:10-4:10 pm; Tu 10:15-11:45 am, 1:00-3:30 pm.		
COURSE WEBSITE	`http://cslab.wheaton.edu/~tvandrun/cs245`		

RESOURCES
Savitch, Walter. *Absolute Java*, second edition, Addison Wesley, 2005.
McDowell, Charlie. *On to C*, Addison Wesley, 2001.

COURSE DESCRIPTION

A continuation of CSCI 235. Searching and sorting algorithms, their analysis and instrumentation. Software development methodology including revision control and API production. Object-oriented programming, subclassing, inheritance, overriding, and class hierarchies. Abstract data structures including linked lists, stacks, queues, and trees. Software design patterns. Introductory system programming, data representation, and computer organization.

GOALS AND OBJECTIVES

1. Students will be able to analyze the complexity of simple algorithms.
2. Students will be able to manage non-trivial software development projects using version control systems and API generators.
3. Students will be able to design and implement non-trivial software systems.
   - Using object-oriented principles and syntax to design and implement type and class hierarchies, including the correct use of abstraction, polymorphism, subtyping, subclassing, inheritance, and overriding.
   - Using UML to manifest the system structure.
   - Using standard data structures.
   - Using widely-known software design patterns.
4. Students will be able to demonstrate their understanding of basic systems, computer organization, and data representation by writing simple C programs.

ASSESSMENT PROCEDURES

1. Quizzes will discipline students to stay current with terminology and concepts in class and help them identify concepts on which they need more work. Specific quizzes will test their ability to analyze algorithms, use and describe Java syntax for object-oriented programming, design class hierarchies, and connect problems with solution patterns.
2. Labs will reinforce the material in class by providing practice and experience. Specific labs will force students to practice software project management, the implementation of class hierarchies, the use and implementation of data structures, and the syntax of the C programming language.
3. Projects will teach students to put ideas from class together for solving larger problems and mark their progress in that ability. Specific projects will require the management of non-trivial software projects, the design and implementation of class hierarchies, the implementation of software design patterns, and system program. At least one project will be a team project.
4. Tests and the final exam will evaluate students' mastery of the comprehensive material.

Grading:

	weight
test 1	10
test 2	10
quizzes	5
labs	10
projects	35
group project	10
final exam	20

SPECIAL EXPECTATIONS

Academic Integrity

Since pair-programming is practiced in *labs*, students sometimes find it unclear what constitutes fair collaboration in *programming projects*. You are encouraged to discuss the problem in the abstract with your classmates; this may include working through examples, drawing diagrams, and even jotting down some pseudocode. If you are really stuck on a compiler error or bug (meaning that you have tried to figure it out for a long time are are stumped), you may ask someone to look at your code to help you find it. Sharing test cases is also a great way to help each other.

While getting code from the Internet is cheating, you may find electronic and print resources helpful in getting ideas for a project. Think in analogy to avoiding plagiarism in a research paper: if you use resources like this, it is critical that you cite them in your comments. Along these lines, if you do receive help beyond what is fair from outside resources (including the Internet) or a classmate, you should give credit. While I reserve the right to deduct points if I judge you to have profited unfairly, I will be very lenient if you are up front and honest about it.

Late assignments

You are allowed a total of two days during the course of the semester—either one assignment two days late or two assignments one day late each. Other late assignments will not be accepted.

Attendance

While I was an undergraduate, I missed a grand total of two classes, and one of them was to take the GRE. I expect the same from my students. Since being a student is your current vocation, since your learning now will affect your ability to support a family and church later in life, and since you, your family, and/or a scholarship fund are paying a large sum of money to educate you, being negligent in your schoolwork is a sin. I do not take attendance, but I do notice. If you must miss class, it is courteous to notify the instructor ahead of time.

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.

V. Algorithms and analysis (CS 235/345)		**2 weeks**
A. Analysis	.3 weeks	
B. Sorting	1 week	
C. Searching	.3 weeks	
D. Instrumentation	.3 weeks	
VI. Software development (CS 335)		**1 week**
A. Methodology	.3 weeks	
B. Version control	.3 weeks	
C. Javadoc	.3 weeks	
VII. Object-oriented programming (CS 235)		**5 weeks**
A. Review	.5 week	
B. Subclasses, abstract classes	1 week	
C. Overriding	.5 weeks	
D. Class hierarchies	.5 weeks	
E. Generics and Java dark corners	1.5 weeks	
F. UML and object-oriented concepts	1 week	
VIII. Data structures (CS 345)		**2.5 weeks**
A. General; linked vs. array-based	.5 weeks	
B. Stacks and queus	1 week	
C. Other data structures	1 week	
IX. Design Patterns* (CS 335)		**1.5 weeks**
B. Factory Method	.5 weeks	
C. Strategy and State	.5 weeks	
D. Adaptor and Decorator	.5 weeks	
X. Systems (CS 351)		**3 weeks**
A. C programming	1.5 weeks	
B. Representation	.5 weeks	
C. Circuits	.5 weeks	