CSCI 235 SYLLABUS

COURSE NAME, NUMBER	Programming I: Problem Solving, CSCI 235
SEMESTER, YEAR	Fall 2008
INSTRUCTOR	T. VanDrunen
OFFICE / TELEPHONE / EMAIL	Armerding 112 752-5692 Thomas.VanDrunen@wheaton.edu
OFFICE HOURS	MTuWThF 2:00-3:00 pm; Th 9-11 am
COURSE WEBSITE	`http://cslab.wheaton.edu/~tvandrun/cs235`

RESOURCES

Savitch, Walter. *Absolute Java*, second or third edition, Addison Wesley, 2005 or 2008 (respectively).
This is the same text as is used in CSCI 245.

COURSE DESCRIPTION

Algorithms, compilers, and programs in a modern, object-oriented programming language. Types, control structures, modularity, and recursion. Object-oriented fundamentals, encapsulation, interface implementation, and subtype polymorphism. Exceptions, libraries, and file I/O.

GOALS AND OBJECTIVES

1. Students will be able to design basic algorithms.
   - To organize data by types and variables.
   - To arrange expressions and statements in their logical order.
   - To use the control structures of branching, iteration, and recursion.
   - To encapsulate functionality into methods.
2. Students will be able to express algorithms in the Java programming language.
   - To make syntactically correct use of the features of Java in implementing the concepts above.
   - To use Java terminology in describing the parts and process of a computer program.
3. Students will be able to practice the basics of object-oriented design
   - To design new types.
   - To encapsulate data and functionality together.
   - To design relationships among types.

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 students' ability to discern types of expression, sketch algorithms, and write small pieces of code.
2. Labs will reinforce the material in class by providing practice and experience. Labs in particular will teach the use of correct syntax.
3. Projects will teach students to put ideas from class together for solving larger problems and will mark their progress in that ability. Specific projects will require students to design an algorithm to solve a problem, design the types necessary to model the information being processed, and implement the types and algorithms with Java.
4. Tests and the final exam will evaluate students' mastery of the comprehensive material.

Grading:

	weight
test 1	12.5
test 2	12.5
quizzes	5
labs	10
projects	40
final exam	20

SPECIAL EXPECTATIONS

Lab protocol

Thursday, 11:15-1:05 is our lab block. Do not bring food to lab. Our laboratory activity will follow a specific protocol called *pair programming*. Two students will work together at one computer, producing a single product, sharing two roles: The *driver* controls the mouse and keyboard and does the actual programming; the *navigator* watches the driver, catches simple mistakes, thinks of ways to test what is currently being programmed, and thinks ahead to the next task in the lab. Students in a pair switch roles between each sub-task, approximately every ten minutes. The program is produced through discussion and collaboration; neither member of the pair should dominate. While you work, your computer will be logged in through a class account; do not log in as yourself during lab unless you are specifically instructed to do so.

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.

What is not allowed is sharing code. You may not program together, and you may not watch each other programming for projects, either to give or receive help. Although it says above that erroneous code may be looked at if the student is stuck, *working* code should be not be shown. As with problems sets in a math or science course, while it is ok to help each other find the right place to look for the answer or discern why an answer is not working out, you should not give or receive the answer. Moreover, downloading relevant code from the Internet is manifestly dishonest.

While getting *code* from the Internet is cheating, you may find electronic and print resources helpful in getting *ideas* for a project. In this case, think in analogy to avoiding plagiarism in a research paper: if you use resources like this, it is crucial 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. When missing a class is unavoidable, it is 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.

CSCI 235

I. Prolegomena — **2 weeks**
- A. Algorithms
- B. Compilers, the VM model
- C. Java history, intro

II. Programming fundamentals — **4 weeks**
- A. Types, variables, expressions, statements
- B. I/O, Strings
- C. Flow of control
- D. Arrays
- E. Modularity and methods
- F. Recursion

III. Object-oriented fundamentals — **5 weeks**
- A. Encapsulation
- B. Class and object
- C. Interfaces
- D. Subtype polymorphism

IV. Other topics — **4 weeks**
- B. Exceptions
- C. Collections
- D. Streams and file I/O
- E. GUI

CSCI 245

V. Algorithms and analysis (CSCI 345) — **2 weeks**
- A. Analysis
- B. Sorting
- C. Searching
- D. Instrumentation

VI. Software development (CSCI 335) — **1 week**
- A. Methodology
- B. Version control
- C. Javadoc

VII. Object-oriented programming (CSCI 235) — **5 weeks**
- A. Review
- B. Subclasses, abstract classes
- C. Overriding
- D. Class hierarchies
- E. Generics and Java dark corners
- F. UML and object-oriented concepts

VIII. Data structures (CSCI 345) — **2.5 weeks**
- A. General; linked vs. array-based
- B. Stacks and queues
- C. Other data structures

IX. Design Patterns (CSCI 335) — **1.5 weeks**
- B. Factory Method
- C. Strategy and State
- D. Adaptor and Decorator

X. Systems (CSCI 351) — **3 weeks**
- A. C programming
- B. Representation
- C. Circuits
- D. Computer organization