Syllabus for CSCI 235 Programming I
Fall 2013

Cary G. Gray
Office: Science 159, x5875

Office hours: MWF 2:30-4:00 p.m.
Th 2:00-3:30 p.m.
I am typically in my office much more than the posted times, and you are welcome to stop by whenever my door is open. Check with me ahead of time if you want to be sure that I'll be there outside scheduled times.
TAs: Kayley Lane, Josh Wright

Class meetings

MWF 12:45–1:50 p.m., Sci 129
Lab: Section 2 — T 11:15 a.m.–1:05 p.m., Sci 154
Section 0 — T 1:15–3:05 p.m., Sci 154
Final exam: 1:30–3:30 p.m on Wednesday, Dec. 18

On-line resources

Additional (and updated) course information will be available at the class page at

http://cs.wheaton.edu/~cgray/csci235/

Textbook

Savitch, *Absolute Java*, third to fifth edition, Addison Wesley, 2005–2010

Description

This course is the first half of a two-semester introduction to programming, emphasizing the process of solving computational problems and expressing those solutions in an object-oriented language. An outline of the two-semester sequence appears at the end of this syllabus. Here is the official catalog description for this semester:
CSCI 235 Programming I: Problem Solving
A first course in computer programming for beginners. Structured and object-oriented programming in Java or a similar programming language. Types, control structures, methods, and recursion; objects, classes, interfaces, encapsulation and polymorphism; exceptions, library classes, file I/O, linked lists, and graphical user interfaces.

Objectives

By the end of this class, you should be able to:

1. Design basic algorithms and organize their description:
    - organize data in terms of types and variables;
    - arrange expressions and statements in their logical order;
    - use the control structures of branching, iteration, and recursion; and
    - encapsulate functionality in methods.

2. Express algorithms in the Java programming language:
   - correctly use the features of the Java language to implement the concepts above; and
   - use the standard Java terminology in describing the parts and process of a computer program.
3. Practice the basics of object-oriented design:
   - design new types;
   - encapsulate data and functionality;
   - design the relationships among types.

Assessment

1. Quizzes are intended as encouragement to keep up with terminology and concepts; they also help students identify the ideas that need more attention. Specific quizzes will test students' ability to sketch algorithms, to write small pieces of code, and to understand short segments of code.
2. Labs provide practice working with the material from class, and especially emphasize correct program syntax.
3. Projects are the opportunity to put the ideas from this class together, solving problems of increasing size as the semester progresses. Specific projects will require designing an algorithm to solve a problem, designing types to model information, and implementing the types and algorithms in Java.
4. Examinations, including the final, will evaluate students' mastery of the course material.

Grades will be weighted as follows:

component	*weight*
mid-term exams	5 (total)
final exam	4
quizzes	1
labs	2
projects	8

Policies

Attendance You are responsible for what happens in class, whether you are present or not. If you are sick or must miss because of other school responsibilities, let me know in advance. If there is an emergency, let me know as soon as practical. (You will do well to think of this as practice for keeping a job. Note that you can reach me by email, and my office phone takes messages at all hours.) Being in class means being on time: arriving late is highly disruptive. If the door is locked when you arrive, do not disturb the class.

Labs and exams will happen as scheduled. Make-up or other alternative arrangements will be made only if arrangements are made in advance (or, for emergencies, if notification is provided promptly).

Because lab work will be done in pairs, it is especially important that you arrive on time and that you provide me advance notice about absences.

Your presence in class calls for your full attention. That implies no use of networked devices to be virtually somewhere else. *If* you need to use a computer of some sort for access to your textbook or to take notes, you should do so with the networking turned off, and you are implicitly promising that you will not use it for any other purpose during class.

Preparation. The course schedule (below) lists sections of the textbook to be read *before* class meetings. What we do in class will be planned with the expectation that you have done your reading. Take notes as you read so that you can recall the most important ideas and anything that you are unsure about.

Labs Work during the scheduled lab will employ *pair programming*. Two students will work together at one computer, producing a single program, by alternating roles. The *driver* controls the mouse and keyboard, entering or editing the program; the *navigator* watches the driver, catches simple mistakes, and thinks about ways to test what is currently being programmed and how to approach the next task. Students in a pair trade roles for each sub-task within the lab, about every ten minutes. Pair work will be done using in-class accounts instead of your personal accounts.

Academic integrity. Because you are encouraged to work together—especially during labs—and provide each other assistance, you do risk inadvertent plagiarism. Be cautious, especially when you ask for or provide assistance. Make sure that you don't let someone else do your work for you, and make sure that you don't do someone else's work.

Because projects are your individual work, what you must not share is code. You may not program together or watch each other programming either to give or receive help. You may, on the other hand, discuss the problem in abstract, work through examples, or even draw diagrams or sketch small bits of pseudocode. Sharing test cases is another good form of collaboration.

If you are stuck trying to understand a compiler error or a program misbehavior, you may ask for help. The rule is that you should not show each other *working* code. When you give assistance, you should avoid giving the correction; instead, help the other student to better understand the problem or to find helpful material in the book.

There is a lot of code and other information available in books and on the Internet—and some of it is even correct. Looking up information is acceptable, but you should be careful to give credit to sources you use. As when you are working with each other, copying *code* is not permitted, nor is asking for solutions. If you inadvertently find a resource that is too helpful, be honest and acknowledge it.

I expect you to conduct yourself honestly in this course. When you submit work, you assert that it is your own. If you use an outside source or receive assistance, acknowledge it. Deliberate misrepresentation will result in no credit for the assignment; a second offense will result in failing the course. All offenses will be reported and are subject to college disciplinary action as well.

Late assignments You are granted two free late days during the semester; that translates to being two days late with one assignment or one day late with each of two. Don't waste that slack, because other late work will not be accepted.

Special circumstances and needs Wheaton College is committed to providing reasonable accommodations for students with disabilities. Any student with a documented disability needing academic adjustments is requested to contact the Academic and Disability Services Office as early in the semester as possible. Call 630.752.5941 or email jennifer.nicodem@wheaton.edu for further information.

Inclusive language By vote of the faculty in April 2012, all syllabi are required to contain the following statement:

> For academic discourse, spoken and written, the faculty expects students to use gender inclusive language for human beings.

Acknowledgement Dr. VanDrunen has done the largest share of the work in developing CSCI 235 and 245. Significant portions of this syllabus and many other class materials are his work or derived from his work.

Schedule

Here is an initial class schedule; additions and changes will show up online.

Date	Reading	
Aug 28		Introduction to computing, algorithms
30		More algorithms
Sep 2	*Labor Day*	
3		*Lab 1: Intro to the lab; making and running programs*
4	1.1, 1.3	**Programming fundamentals** Java programs; strings
6	1.2, 1.4, 2.1	Types, variables, expressions, statements
9	2.2, 3.1, 3.2	The Boolean type; more strings start Project 1 (due Sep 16)
10		*Lab 2: First Java programs*
11	3.3, 3.4	More control structures
13	6.1, 6.2	Finish control structures; start arrays start Project 2 (due Sep 23)
16	6.3	Arrays; for loop
17		*Lab 3: More strings*
18	6.4	More arrays, multidimensional arrays
20		*Lab 4: arrays*
23	5.1	Methods start Project 3 (due Oct 2)
24		*Lab 5: methods*
25		More with methods
27		Methods; libraries; return; catching exceptions
30	11.1–3	Recursion
Oct 1		*Lab 6: Recursion*
2		review start Project 4 (due Oct 9)
4		**Exam**
7	4.1	**Object-Oriented Fundamentals** Classes, user-defined types
8		*Lab 7: First classes*
9	4.3–4, 5.1–4	Implementation of classes start Project 5 (due Oct 18)
11	4.2	Classes and methods; encapsulation
14	8.1, 13.1	Interfaces, subtype polymorphism
15		*Lab 8: subtyping (chess)*
16		Interaction of objects start Project 6 (due Oct 30)
18	15.1	Linked lists

Date	Reading	
21	*Quad break*	
22	*Quad break*	
23		More linked
25		*Lab 9: linked 1*
28	15.4, 15.7	other linked
29		*Lab 10: linked structures*
30		more linked start Project 7 (due Nov 13)
Nov 1		Review
4		**Exam**
5		*Lab 11: simulation*
6		*Lab 12: continuation of simulation*
8		debrief; review linked
11	14.1, 16.1–3	**Applied Topics** Collections, iterators
12		*Lab 13: collections*
13		more collections start Project 8 (due Dec 11)
15	9.1–3	*Lab 14: Catching and classifying exceptions*
18	17.1–5	GUI and events
19		*Lab 15: GUI*
20		More event-handling
22		*Lab 16: GUI2*
25	18.3–4	Graphics
26		*Lab 17: Graphics*
27	*Thanksgiving*	
29	*Thanksgiving*	
Dec 2		Files and I/O, Exceptions
3		*Lab 18: File I/O, Exceptions*
4		Defining exceptions
6		Wrap up exceptions
9		Review example
10		*Lab 19: Putting it all together*
11		Review
13		Review
1:30-3:30pm, Wed 18 Dec		**Final exam**