Syllabus for CSCI 235 Programming I
Spring 2015

Cary G. Gray
Office: Science 159, x5875
Home: *omitted online*
E-mail: *omitted online*
Office hours: MW 1:00–3:00 p.m.
F 2:00–3:00 p.m.
and by appointment (esp. Tuesday mornings)
You are welcome to stop in when my office door is open.
TA: Johnny Edman *omitted online*

Class meetings

MWF 9:15–10:20 a.m., Sci 131
Lab: Th 8:30-10:20 a.m., Sci 154
Final exam: 8:00–10:00 a.m on Thursday, May 7

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*, fifth edition, Addison Wesley, 2012. (Fourth edition is also fine.)

Note that this textbook—a reference, actually—is also used for CSCI 245.

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

Note that your course grade may be lowered up to one letter for disruptive behavior or for recurrent absence or tardiness.

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).

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. Talk with me beforehand if you think you need to do this.

Preparation. The course schedule (below) lists handouts and 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.

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.

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.

Gender-neutral language 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		Projects
Jan 12		Introduction to computing, algorithms	
14		More algorithms	
15		*Lab 0 Intro to the lab*	
16	1.1, 1.3	**Programming fundamentals** Java programs; strings	
19		*MLK Holiday*	
21	1.2, 1.4, 2.1	Types, variables, expressions, statements	start Project 1
22		*Lab 1 First Java programs*	
23	2.2, 3.1, 3.2	The Boolean type; more strings	
26	3.3, 3.4	More control structures	
28	6.1, 6.2	Finish control structures; start arrays	start Project 2
29		*Lab 2 More strings*	
30	6.3	Arrays; switch statement	Project 1 due (Feb 3)
Feb 2	(pre-lab)	*Lab 3 Arrays*	
4	6.4	More arrays, multidimensional arrays	
5		*Lab 4 First methods*	
6	5.1	Methods Library methods	
9	sorting	More with methods; sorting	Project 2 due (Feb 10) start Project 3
11	9.1	Methods; catching exceptions	
12		*Lab 5 Methods*	
13	11.1–3	Recursion	
16		*Presidents Day*	
18		More recursion; classes as modules	start Project 4
19		*Lab 6 Recursion*	
20		Review sample exam	Project 3 due
23		**Exam**	
25	4.1	**Object-Oriented Fundamentals** Classes as types	
26	(pre-lab)	*Lab 7 First classes*	
27	4.2, 4.3–4, 5.1–4	Classes, methods, encapsulation	start Project 5
Mar 2	8.1, 13.1	Interfaces, subtype polymorphism	Project 4 due (Mar 3)
4		Interaction of objects	
5		*Lab 8 Subtyping*	
6		More interaction examples	Project 5 due start Project 6

Date	Reading		Projects
Mar 9-13		*Spring break*	
16		Simulation	
18	15.1	**Linked structures** linked lists	
19		*Lab 9 Simulation*	
20		Lists, again	
23	15.4	*Lab 10 Linked lists*	
25		Other linked structures	Project 6 due start Project 7
26		*Lab 11 Trees*	
27		Review sample exam	
30		**Exam**	
Apr 1	14.1, 16.1–2 5e/pp. 277–279 4e/pp. 271–273	**Applied Topics** Collections Collections API	
2		*Lab 12 Using collections*	
3		*Good Friday*	
6	16.3	More collections; iterators	
8		Collections, for-each; handling exceptions	
9		*Lab 13 More collections*	
10	17.1–2, 17.4–5 GUI classes	Handling exceptions; GUI events	Project 7 due start Project 8
13		*Lab 14 GUI event handling*	
15	17.3	More GUI, layouts	
16		*Lab 15 More GUI*	
17		More GUI, layouts	
20		*Lab 16 Graphics*	
22	graphics 18.3–4	Graphics	
23		*Lab 17 File I/O and exceptions* *File I/O classes* *File I/O overview*	
24		Files and I/O	
27		Review Example	
29		Review	Project 8 due
30		*Lab 18 Putting it all together*	
May 1		Review	
8:00-10:00am, Thurs 7 May		**Final exam**	