Syllabus for CSCI 235 Programming I
Spring 2011

Cary G. Gray
Office: Science 159, x5875
Home: 630-784-1540 (before 10:00 p.m., please)
E-mail: `Cary.Gray@wheaton.edu`
Office hours: M 3:10–4:30 p.m.
T 1:30–3:30 p.m.
WF 9:00–10:20 a.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.
TA: Andrew Wolfe
Andrew also works in the CS lab on Thursday evenings.

Class meetings

MWF 12:45–1:50 p.m., Sci 131
Lab: T 8:30–10:20 a.m., Sci 154
Final exam: 8:00-10:00 a.m. on Wednesday, May 4

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

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

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.

Lab activities Work during the scheduled lab will emply *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. 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.

Attendance. I expect you to be in class, and 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.)

I also expect you to be on time. If you come in late, don't interrupt class, and plan to find out what you've missed from another student *after* class.

Special circumstances and needs I will work with you if you have any kind of special need, provided you are responsible in letting me know in time to make appropriate arrangements. Accommodation of learning disabilities requires that they be documented with the registrar's office *and* that you inform me in time for me to communicate with that office and make the needed arrangements.

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		
Jan. 10		**Introduction** Intro to computing, algorithms	
11		*Lab 1:* CS lab intro	
12	1.1, 1.3	**Programming fundamentals** Java; Strings	
14	1.2, 1.4, 2.1	Types, variables, expressions, statements	
17	*MLK Day*		
18		*Lab 2:* First Java programs	Project 1 (due Jan 26)
19	2.2, 3.1, 3.2	boolean type; more strings	
21		*Lab 3:* Types	
24	3.3, 3.4	More control structures	Project 2 (due Feb 2)
25		*Lab 4:* More types, input and output	
26	6.1, 6.2	Finish control structures; start arrays	Project 1 due
28	6.3	Arrays; for loop	
31		*Lab 5:* Arrays	
Feb 1	*Faculty Workshop*		
2	6.3, 6.4	Multi-dimensional arrays	Project 2 due
4	5.1	Methods	Project 3 (due Feb 18)
7		More with methods; libraries	
8		*Lab 6:* Methods	
9			
11	11.1–3	Finish methods; Recursion	
14		Review	
15		*Lab 7:* Recursive methods	
16		**Exam 1**	
18	4.1	**Object-oriented fundamentals** Classes, user-defined types	Project 3 due
21	*Presidents Day*		
22		*Lab 8:* Classes	
23	4.3–4, 5.2–4	Class implementation details	
25	4.2	Classes and encapsulation	
28	8.1, 13.1	Interfaces, subtype polymorphism	
Mar 1		*Lab 9:* Classes, again	
2		More classes, polymorphism	
4		*Lab 10:* Subtyping	
7–11	*Spring Break*		

Date	Reading	
Mar. 14		Encapsulation, interactions
15		*Lab 11:* Simulation
16	9.1–3	Exceptions
18		*Lab 12:* Exceptions
21	15.1	Linked lists
22		*Lab 13:* Linked lists
23	15.4, 15.7	Other linked structures
25		*Lab 14:* Linked structures
28	14.1, 16.1–3	**Applied topics** Collections, iterators
29		*Lab 15:* Collections
30	17.1–5	GUI
Apr. 1		*Lab 16:* GUI
4		More GUI
5		*Lab 17:* More GUI
6		Review
8		**Exam 2**
11	18.3–4	Graphics
12		*Lab 18:* Graphics
13		
15	10.1–3	File I/O
18		
19		*Lab 19:* File I/O
20		
22	*Good Friday*	
25		
26		*Lab 20: TBD*
27		Review example
29		Review
Wed., May 4, 8:00–10:00 a.m.		**Final exam**