CSC140 Foundations of Computer Science

Professor. Leon Tabak

Table of Contents

Calendar

	MON	TUE	WED	THU	FRI
Week 1	12	13	14	15	16
Week 2	19	20	21	22	23
Week 3	26	27	28	29	30
Week 4	02	03	04	05	06

Our meeting times and places

My office is in Law 206C.

You may call me in my office at (319) 895 4294.

You may send me electronic mail at `<l.tabak@ieee.org>`.

I will be in my office and available to meet with you Monday through Friday from 3:00 p.m. until 3:30 p.m.

We will all meet together in the classroom in the mornings and in the laboratory in the afternoons.

	Where	**When**
Classroom	Law Hall 121	1 p.m. to 3 p.m.
Laboratory	Law Hall 113	9 a.m. to 11 a.m.

Textbooks

Introduction to Programming in Java: An Interdisciplinary Approach [http://www.cs.princeton.edu/IntroProgramming] . Robert Sedgewich and Kevin Wayne. Addison-Wesley. Copyright © 2008. 978-0-321-49805-2.

GridWorld AP Computer Science Case Study—Student Manual [http://apcentral.collegeboard.com/apc/public/courses/teachers_corner/151155.html] . Chris Nevison. Barbara Cloud Wells. Cay Horstmann. The College Board. New York . Copyright © 2007.

Etiquette for the Classroom

Please show respect to your classmates, to me, and to the seriousness of our enterprise by exercising the following courtesies:

- Please give your attention to whomever is speaking. That'll be me some of the time, but it will be you some of the time too.

  Did you bring a computer to class? Good. We will find ways to use your computer to accomplish the goals that we have set for ourselves in this course.

  Now, please turn off the games. Close windows that are displaying news, electronic mail, and scores from the world of sports. Put the earbuds away.

- Please do not interrupt and distract the class by late entries, early departures, or by coming less than fully prepared to make your contribution to the class. If you anticipate a need to be absent or late, please notify me in advance of your anticipated absence. With all due respect to Admiral Grace Murray Hopper, excuses after the fact will not succeed.

- Tastes in music vary. Some people need more quiet than others in order to concentrate. I and your classmates would like to know that we, and not an MP3 player, have your attention when we speak to you. Keep these things in mind.

- Please refrain from bringing food or drink into the classroom or laboratory.

- Please refrain from the use of vulgar language.

- Please do not wear clothing or buttons that bear vulgar messages or images. Indeed, it is best to avoid wearing messages of any kind. Outside of the classroom, I will be happy to discuss with you any issue of the day. Inside of the classroom, it is rude to broadcast opinions unrelated to our subject because our work there does not allow anyone else to respond to your challenge (or to decline your invitation to debate).

- Please keep shoes on and hats off in the classroom. Leave your pajamas and bathrobe at home. Test the limits of social conventions if you must during the more than 80% of the week during which we shall be apart.

- Please demonstrate your love for your fellow man or woman with kind words and gracious gestures but delay other physical expressions of tenderness until our work is done and you have found a more private setting.

Policies

Cornell College is committed to providing equal educational opportunities to all students. If you have a documented learning disability and will need any accommodation in this course, you *must* request the accommodation(s) from the instructor of the course and no later than the third day of the term. Additional information about the policies and procedures for accommodation of learning disabilities is available on Cornell College's Web site [http://www.cornellcollege.edu/academic-support-and-advising/disabilities/academic-accommodation/index.shtml].

Please also familiarize yourself with the college's statement on academic honesty [http://www.cornellcollege.edu/registrar/pdf/Academic%20Honesty.pdf] and its policies for dropping courses [http://www.cornellcollege.edu/registrar/gb-resources-student/add-drop.shtml]

Description of the course

This course introduces students to problems that engage the interests of computer scientists and define the field. The course introduces students to object-oriented design, a principal discipline that computer scientists use to solve problems. Students learn to divide large problems into small problems, bundle related data with methods that operate on that data, and incorporate into new designs elements of previously completed designs. The course emphasizes creative expression using an abstract notation. Students practice designing, writing, testing, and presenting programs. Success in the course does not require previous programming experience.

Objectives of the course

Upon the successful completion of this course, students will be able to:

- Identify important problems in computer science and important contributors to the field.

- Translate simple algorithms (including iterative and recursive algorithms and algorithms for searching and sorting) into working code.

- Analyze and compare alternative methods for the solution of a given problem (by, for example, counting the number of instructions that the computer must execute using each method).

- Design, write, and test a class that models a number that has two parts. Examples of numbers that have two parts include fractions (numerator and denominator), lengths (feet and inches), weights (pounds and ounces), times (hours and minutes), vectors (x and y components), and complex numbers (real and imaginary components).

- Present programs that they have written to their peers using the Javadoc tool to extract comments from code and UML class diagrams.
- Describe the attitudes and habits that define professional practice in software engineering.

Grades

Written work will be due on each day of the term except for the first day and the last day. Printed copies and electronic copies of your papers will be due at 9 a.m.

Experience presenting work to peers will be a central part of the course. Practice asking your teammates questions during their presentations, critiquing their decisions, and suggesting improvements to their code will also be an important part of your education during this term. We will schedule one day in each week of the term for you to present your work.

	Activity	**Points**
	Daily problems and writing	16
	Code reviews (4 presentations)	20
	Examination 1 (Friday, January 16)	16
	Examination 2 (Friday, January 23)	16
	Examination 3 (Friday, January 30)	16
+	Examination 4 (Wednesday, February 4)	16
		100

Daily schedule

Monday, January 12

Read.

Read Section 1.1 on pages 3–13 (11 pages) in *Introduction to Programming in Java*.

Discuss.

What you can learn in this course that will help you in other courses and in your career, even if you never study computer science again.

How to write a program, prepare a program for execution, and run a program.

- What is computer science?
- The role of programming in the study of computer science.
- Why the Java programming language?
- The kinds of tools that programmers use in their work.

- Kinds of errors in computer programs.
- Knowledge, skills, habits, and attitudes that lead to success.

Church-Turing Thesis—what all computers and all programming languages have in common.

Components of a computer:

- microprocessors
- memory
- busses
- input and output devices

Software tools:

- operating systems
- compilers
- interpreters and virtual machines
- editors
- Integrated Development Environments (IDEs)

Write.

Write solutions to Exercises 1.1.1, 1.1.2, 1.1.3, 1.1.4, and 1.1.5 (on page 13 of *Introduction to Programming in Java*).

These exercises will familiarize you with tools we will use throughout the term and with the form of programs written in the Java language.

Return to the top.

Tuesday, January 13

Read.

Read pages 14–34 (21 pages) in Section 1.2 in *Introduction to Programming in Java*.

Discuss.

How (and why) to associate types with data, how to combine data arithmetically and logically, how to convert data from one type to another.

- Literals, identifiers, and variables.
- Integers, floating pointing numbers, characters, strings, and Boolean values,
- Precedence and associativity of operators.

- Conversions between data types.

We will look more closely at the parts of a class.

- Comments.
- Constructors, accessors, and mutators.
- Instance variables, class variables, and parameters.

Programmers always write for two audiences. They write for the machine—they must write code that the compiler or interpreter will accept. They also write for their human collaborators and clients. Other people will build upon the code, use the code in other projects, test the code and correct errors in the code. Programmers have an obligation to help their teammates by making their code as easy to understand as possible.

We will review conventions for naming classes, methods, variables, and constants. Conformity to these conventions will make it easier for other people to understand what we have written.

Write.

Write solutions to Exercises 1.2.4, 1.2.10, 1.2.11, and 1.2.22 (on pages 39–44 of *Introduction to Programming in Java*).

You will learn how arithmetic on a computer differs from the arithmetic done in a mathematics classroom.

Return to the top.

Wednesday, January 14

Read.

Read pages 35–45 (11 pages) in Section 1.2 in *Introduction to Programming in Java*.

Discuss.

- The “software crisis”.
- Re-usable components in computer programs.
- Applications Programmering Interfaces (APIs)

Practice with declarations of variables, initializations of variables, expressions that include variables, literals, operators, and calls to methods. We will get some practice using the methods found in the Math class.

A variable is a named location in the computer's memory. A variable has six attributes:

1. name
2. type
3. value
4. address

5. scope

6. lifetime

We will introduce types. We will define the differences between integers and floating point numbers.

- What is a type?
- Why did the designers of Java include type-checking in the language?

The interpreter can produce values by evaluating expressions. We will learn how to write expressions that include numeric literals, calls to methods, and arithmetic operators.

Write.

Write solutions to Exercises 1.2.31 and 1.2.33. (on pages 43 and 44 of *Introduction to Programming in Java*).

Your programs will compute functions found in Geographic Information Systems. A Geographic Information System is a tool for creating, editing, sharing, interpreting, and analyzing maps and other geographic data.

Return to the top.

Thursday, January 15

Read.

Read pages 46–73 (28 pages) in Section 1.3 in *Introduction to Programming in Java*.

Discuss.

How to direct the computer to choose between two alternative sequences of statements. How to direct the computer to execute a sequence of statements repeatedly.

- computing square roots
- producing the binary representation of an integer
- simulating gambler's ruin
- if statements
- while statements
- for statements
- break statements
- do-while statements
- switch statements
- compound statements

- examples to illustrate the use of loops:

We will learn how to instruct the computer to make decisions. We will learn how to describe the condition under which the computer should take a prescribed action by writing expressions that evaluate to true or false. This will require the introduction of Boolean operators.

Write.

Write solutions to Exercise 1.3.3, 1.3.4, 1.3.5, and 1.3.6 (on page 77 of *Introduction to Programming in Java*).

You will begin to learn how to guard against errors and make your programs more robust.

Return to the top.

Friday, January 16

Read.

Read pages 74–85 (31 pages) in Section 1.3 in *Introduction to Programming in Java*.

Discuss.

Practice with if statements, for loops, and while loops.

Learn about several famous and curious mathematical problems. Discover that your new knowledge of programming enables you to solve them!

Write.

Write solutions to Exercises 1.3.43 and 1.3.44 (on page 85 of *Introduction to Programming in Java*).

In these exercises you will see a little of how computer science has provided new methods for discovery and experimentation in the natural sciences and mathematics.

Examination 1.

Return to the top.

Monday, January 19

Read.

Read pages 86–111 (26 pages) in Section 1.4 in *Introduction to Programming in Java*.

Discuss.

How to create and use variables that hold multiple values. How to declare and initialize arrays, assign a value to an element of an array, retrieve a value, and exchange two values.

Write.

Write solutions to Exercises 1.4.18 and 1.4.19 (on page 115 of *Introduction to Programming in Java*).

Return to the top.

Tuesday, January 20

Read.

Read pages 112–119 (8 pages) in *Introduction to Programming in Java*.

Discuss.

Practice interpreting and writing loops that manipulate arrays. How to recognize and avoid easy to make mistakes.

We will look more closely at looping and decision-making constructs. We will complete our list of Java's control statements. We will practice rewriting Boolean expressions, loops, and branches to make our programs easier to read and understand. Once again, we will emphasize that we are writing for both the machine and other human beings—we must make our intentions clear to both!

Write.

Write solutions to Exercises 1.4.12, 1.4.15, and 1.4.17 (on pages 114 and 115 of *Introduction to Programming in Java*).

In these exercises, you will learn more about how to work with matrices. Matrices have important applications in computer graphics.

Return to the top.

Wednesday, January 21

Read.

Read pages 120–151 in Section 1.5 (32 pages) in *Introduction to Programming in Java*.

Discuss.

How to read, write, and format data. How to create drawings and sounds.

Write.

Write solutions to Exercises 1.5.5, 1.5.6, and 1.5.7 (on page 154 of *Introduction to Programming in Java*).

In these exercises, you will see some simple ideas for the design of error detection and correction codes and data compression codes.

Return to the top.

Thursday, January 22

Read.

Read pages 152–161 (10 pages) in Section 1.5 in *Introduction to Programming in Java*.

Discuss.

Write.

Write solutions to Exercises 1.5.19 and 1.5.21 (on page 157 of *Introduction to Programming in Java*).

You will write programs that draw geometric patterns.

Return to the top.

Friday, January 23

Read.

Read Section 1.6 on pages 162–180 (9 pages) in *Introduction to Programming in Java*.

Discuss.

Java is more than a programming language. It is a family of technologies, tools, and libraries of software. Within Java's libraries, the classes that model buttons, labels, menu items, scrollbars, and other elements of graphical user interfaces rank among the most important. These classes are components (designed, written, and tested by other people) that we can plug into our own programs. We do not have to write everything from scratch!

Write.

Write a solution to Exercise 1.6.13 or 1.6.17 (on page 180 of *Introduction to Programming in Java*).

Examination 2.

Return to the top.

Monday, January 26

Read.

Read Section 2.1 on pages 183–217 (35 pages) in *Introduction to Programming in Java*.

Discuss.

- How mathematicians define functions and how computer scientists define methods.
- How (and why) to define our own static methods.
- How to give a method a parameter that is an array.
- How to define a method that returns an array to its caller.
- How to represent harmonic tones in a musical note.

Write.

Write solutions to Exercises 2.1.7, 2.1.8, 2.1.19, and 2.1.21 (on pages 209 and 212 of *Introduction to Programming in Java*).

Return to the top.

Tuesday, January 27

Read.

Read Section 2.2 on pages 218–253 (36 pages) in *Introduction to Programming in Java*.

Discuss.

- How and why programmers share code.
- How to create our own libraries (collections of related methods).
- The special importance of testing and documenting code that is to be shared.

Write.

Write solutions to Exercises 2.2.1, 2.2.3, and 2.2.11 (on pages 248–249 of *Introduction to Programming in Java*).

Return to the top.

Wednesday, January 28

Read.

Read Section 2.3 on pages 254–285 (32 pages) in *Introduction to Programming in Java*.

Discuss.

- How to write recursive methods.
- Reasons for sometimes prefering recursion.
- Cases in which recursion does not provide an advantage.
- Classic examples of recursive algorithms.

Write.

Write solutions to Exercises 2.3.1, 2.3.2, 2.3.7, and 2.3.30 (on pages 278, 279, and 284 of *Introduction to Programming in Java*).

Return to the top.

Thursday, January 29

Read.

Read Section 2.4 on pages 286–313 (28 pages) in *Introduction to Programming in Java*.

Discuss.

Write.

Write solutions to Exercises 2.4.1, 2.4.4, 2.4.9, and 2.4.20 (on pages 308, 309, and 312 of *Introduction to Programming in Java*).

Return to the top.

Friday, January 30

Read.

Read Parts 1 and 2 on pages 3–15 (13 pages) in *GridWorld Case Study*.

Discuss.

The Waterfall Model, although flawed and outdated in important ways, is a good starting point for a discussion of the ways in which software engineers create new products. An examination of the Waterfall Model (and better models that followed the Waterfall Model) will also give us a clearer picture of where opportunities lie within the software engineering profession.

- Requirements. (Determine what the clients need. No Java here! The clients probably do not know very much about computer science and we probably have a lot to learn about our clients' work.)
- Specification. (Quantify the required functionality, cost, and performance. Establish priorities. Negotiate with clients.)
- Design. (Recruit a team, choose a programming language and other tools, develop a budget and schedule, divide the problem into pieces, and define the relationships among components.)
- Code. (Translate the design into a working program that meets the specifications.)
- Test. (There are many kinds of tests: alpha, beta, white box, black box, unit, modular, and system. We will need testers who have a deep understanding of software and testers who have a deep understanding of how our clients work.)
- Release. (Make the case to potential customers that this is a product that can make their work easier. Help customers learn how to use the software effectively.)
- Maintenance. (Correct errors as they are discovered and reported. Add new features. Adapt the program to run on other platforms.)
- Retirement. (Give notice that support for the product will end. Meet and conclude all contractual obligations. Maintain and build good will so that customers will want to buy the next product.)

Which of these roles most appeals to you? How do you think that this sequential approach might fail in practice? How could we improve upon this formula for the development of software?

Write.

Complete the exercises in Part 2 of the case study (on pages 13, 14, and 15). These exercises ask you to extend the Bug class in several different ways. Each of your several kinds of bugs will move in its own special way.

Examination 3.

Return to the top.

Monday, February 02

Read.

Read Parts 3 and 4 on pages 16–38 (23 pages) in *GridWorld Case Study*.

Discuss.

We will consider ways in which software engineering differs from other kinds of engineering. How does software differ from other kinds of products (or services)?

The world needs computer scientists with many different talents and interests. We will review some of the talents and interests that could be the foundation for satisfying and successful study of computer science.

We will survey the variety of special fields within computer science, other academic disciplines, and careers in which students might apply their knowledge of computer science.

Write.

Design, write, and test a program by following the instructions found in the Group Activity of Set 6 in Part 3 of the case study (on page 26).

Design, write, and test a program by following the instructions found in the Group Activity of Set 9 in Part 4 of the case study (on page 36).

Return to the top.

Tuesday, February 03

Read.

The ACM's and IEEE Computer Society's Software Engineering Code of Ethics and related articles that we will find online.

Discuss.

Identify the special responsibilities of a professional software engineer.

Outline opportunities for further study of computer science.

Prepare for the final examination by reviewing what we have learned during the term.

Write.

An abstract of an article that considers the responsibilities borne by professional software engineers.

Evaluate the course.

Return to the top.

Wednesday, February 04

Read.

Review previously assigned readings, your notes, and your programs.

Discuss.

There is no discussion scheduled for today.

Write.

Final examination.

Return to the top.

Thursday, February 05

Read.

There is no reading assigned for today.

Discuss.

We will not meet today.

Write.

There is no writing assignment for today.

Return to the top.

Friday, February 06

Read.

There is no reading assignment for today.

Discuss.

We will not meet today.

Write.

There is no writing assignment for today.

Return to the top.