

COMPUTER SCIENCE S-111A

Syllabus and Other Important Information

I. Overview

Computer Science S-111a is an intensive, 4-week introduction to computer science. The course's primary goal is to teach problem-solving and programming through the design and implementation of *algorithms*. CSCI S-111a is designed for students who are contemplating advanced study in the field of computer science, as well as for mathematicians, biologists, engineers, economists, physicists, linguists and others who need to make serious use of computing for various reasons. It is *not* a "computer literacy" course!

CSCI S-111a covers most of the topics in the Harvard Extension School courses CSCI E-50a and CSCI E-50b. Relative to Harvard College courses, it includes a substantial fraction of CS-50.¹ The principal programming language used is *Java*, which is utilized widely in software development.

Prerequisites

There is no formal computer science prerequisite. Although many of you may have programmed before — perhaps in *HTML* or *BASIC* — there will members of the class who have never written a program. Therefore, this course "starts from scratch," and we offer a lot of resources to help you get quickly up to speed.² There is also no

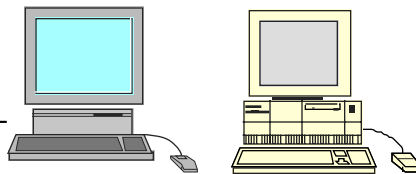
¹ Note, however, that the Harvard College course CS-50 teaches the C programming language, not *Java*. CS-51 currently uses Lisp and C++.

² Those of you who successfully complete this course, and then continue on with CSCI S-111b, may become so skillful that you may be able to go out and land a programming job at the end of the summer!

specific math prerequisite, but we will occasionally use algebraic notation. If you have taken math courses up through pre-calculus, then you should more than adequately prepared.

This course requires a lot of time!!! Plan on at least 20 to 30 hours a week outside of class. Working at a half-time job is inadvisable; working more than that is a virtual guarantee of disaster. *You have been warned!* Computers have a tendency to sometimes “crash” right before a critical deadline, so remember to get started on your homework assignments EARLY, and to save your work often. If you don't have your own personal computer, that's fine: the Science Center's computing facilities are available 24 hours a day; the 53 Church Street labs are available from 9 AM until midnight (Sunday through Thursday), and until 10:00 PM on Friday and Saturday.

Software/Hardware



Each student will write a number of computer programs using

- The *JKarel* software simulator, which currently runs on both the Macintosh OS X and Windows 98/2000/XP platforms. This software package is available from our course website, <http://www.fas.harvard.edu/~libs111>
- *javac* (a Java compiler that runs on the UNIX operating system). You can download the latest version for your own personal computer from <http://java.sun.com/j2se/downloads.html>. We recommend you install version 5.0 of J2SE (Java 2 Standard Edition).

One can, **at one's own risk**, use other *Java* development environments (e.g., Metrowerk's *CodeWarrior* for Macintosh/Windows or Borland's *JBuilder*), but the S-111a course staff will only be able to support *javac*. If you use any of these other software packages, you **MUST** still test your programs out on the UNIX platform using *javac* before turning in your work.

Course Goals

The emphasis of CSCI S-111a is on learning problem-solving strategies and a programming style that places a high value on programs that are easy to read, maintain, and modify. These issues are important in any type of programming activity, regardless of the specific language used. A second goal is to introduce elements of modern software-engineering methodology (in particular, object-oriented analysis) that are essential to the successful design and implementation of large and complex projects.

We will also survey the landscape of computer science as it exists today, with some reference to its past and future. This will enable us to touch on a variety of topics of some universality within computer science, as outlined below:

- 🖥️ *Computer Systems:* representation of information inside a computer, major hardware components, and systems software.
- ✈️ *Applications of Computers:* simulation and modeling, text processing, computer graphics, data management and analysis.
- 🚲 *Data Structures:* All information processed by a computer is ultimately encoded as a sequence of bits. We will consider how to impose order and structure on those bits so that the encoded information is readily available and easy to manipulate. The study of data structures also includes the design, implementation, and analysis of structures and techniques for information processing at all levels of complexity: from individual bits, characters and words, to aggregates such as objects, arrays, and files; and from abstract structures such as stacks and queues, to their concrete realization.
- 🖥️ *Analysis of Algorithms:* for searching, sorting, and other tasks.

In covering the above subject matter, we hope to impart to the student a substantial understanding of both the theory and practice of computer programming.

II. Course Staff

Faculty: Henry Leitner 51 Brattle St., rm. W-725
(617) 495-9096
leitner@harvard.edu

Dr. Leitner will generally be available for consultation (and sometimes consolation) immediately following lecture, at 11:00 A.M. He can also be reached at his Brattle St. office on Tuesday and Thursday afternoons, from approximately 2:30 PM until 4:00 PM.

Teaching Fellow: Mr. David Egli deggli@gmail.com
Course Assistant: Mr. Michael Leonardo mikeleo@wpi.edu

Note that the teaching fellow and course assistant are responsible for grading all of the homework and for helping students, in general, with the material covered in CSCI S-111a.

Mr. Egli will be holding daily “section meetings” in **53 Church Street room 203** which you are all required to attend (more about that below), and will also have regular “office hours” which will be announced later. David will use Science Center room 101e as an office for meeting privately with students. In addition, both David and Michael will assist Dr. Leitner with a number of very important administrative matters (such as maintaining our course website, administering “unit quizzes,” and so on).

If the course enrollment gets very large, then David will teach an additional section at another time (either in the afternoon or in the early evening).

III. Course Activities

Lectures: Held on Monday, Tuesday, Thursday and Friday mornings from 8:30 AM until 11:00 AM in Science Center hall A, and will begin promptly at 8:35. We may have a few Wednesday lectures, but most of those meetings will be optional, and mainly for review purposes.

Participation: From time to time, the last half hour of lecture may be used for writing a short program collectively. Everyone (or almost everyone) will be called on to make small contributions. Feel free to interrupt and ask questions whenever you feel confused about anything going on in class.

Section Meetings: In addition to the 4 or 5 lectures per week, David Egli will conduct a daily (i.e., Monday through Friday) 60-minute section meeting at Noon on days when there is no Unit Quiz, and at 1 PM on days when there is a quiz. These section meetings will start officially on Tuesday, June 28, and will generally not introduce new material, but will instead focus on matters that are especially relevant to the problem set you are currently working on.

Students who have never used computers before, or who would like a hand-on introduction to the Harvard computing environment (including JKarel) should plan to attend a special, optional section meeting at Noon on Monday, June 28, at 53 Church Street.

Unit Quizzes: For the first 4 units, Mr. Leonardo will proctor a compulsory, open-book *unit quiz* at Noon (see the syllabus at the end of this handout for the precise schedule). This means the actual section meeting will start around 1 PM, right after the quiz ends on those days. If you score less than 70% on a unit test (i.e., a 17.5 or lower), you may repeat it one or two days later at 1 PM for a maximum score of 18. A practice test for each unit will be distributed in advance. **Quizzes are held in 53 Church Street, room 203.**

We recommend that each CSCI S-111a student attend a private 10-minute conference with one of the teaching staff, once per week. The purpose of these informal meetings is, of course, to help students with problems they may be having with the homework or with specific

topics. We will arrange the exact time and place for these meetings (but they will ideally take place right after lecture has ended).

Attendance at both the daily lectures and section meetings is essential, as this course moves very fast. If you **do** miss a lecture, be sure to get copies of whatever handouts you may have not received. The handouts are numbered sequentially, and extra copies will be available in one of the turnstiles in hallway just outside Science Center room 110. Your TF may also have a couple of extra copies of handouts, in case you are missing some.

Important course announcements — e.g., class lists, solutions to homework problems, etc., may occasionally be sent to you via *e-mail* (electronic mail). Please pay careful attention to these notices. Some will be of greater and more immediate importance than others. Those of you who intend to make heavy use of computers other than our network of PCs take note of this!

IV. What to Read

First, you must immediately purchase the CSCI S-111a “Coursepack.” Students can purchase the CoursePack from a special website,

http://www.uis.harvard.edu/digital_print_services/coursepacks/

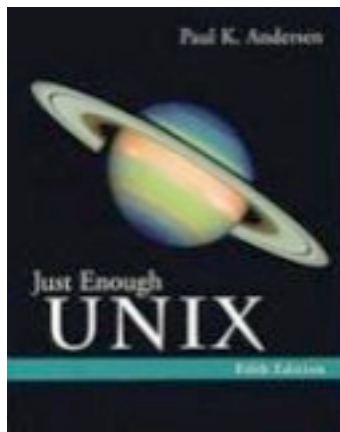
with a Visa or Mastercard. Once the order is placed, students can pick up their CoursePack at the Science Center in the Technology Showcase when they receive their order complete email. Students wishing to pay with cash or check must do so at our main production facility at 219 Western Ave. in Allston. The cost of the CoursePack will be approximately \$25.00 or so.

The various textbooks for this course are for sale at the Harvard COOP, but there is only one book that is required for purchase:



- ✓ *Java: An Introduction to Computer Science and Programming* (fourth edition), by Walter Savitch. Published by Prentice-Hall, 2005. ISBN #0-13-149202-0. Note that this book is bundled with two other items at the Harvard COOP bookstore: a useful paperback reference entitled *The Essential Java Class Reference for Programmers*, by Brian Durney; and *Borland J-Builder* software under ISBN #0-13-153786-5.

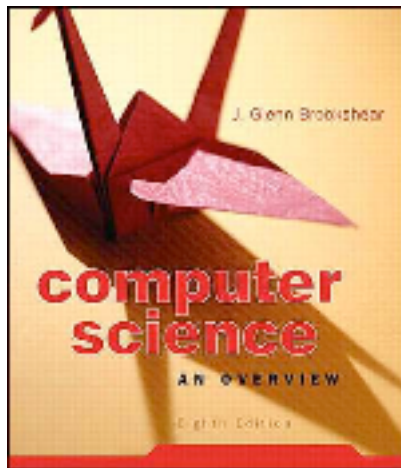
In addition, there are several supplementary textbooks which are NOT required for purchase. They should be for sale at the Harvard COOP and may be available at the Cabot Science Center Library:



- ✓ *Just Enough UNIX*, by Paul Andersen (5th edition). Published by McGraw Hill, Inc., 2005. ISBN #0-0729529-70.



- ✓ *On to Java²* (3rd edition), by Patrick Winston and Sundar Narasimhan. Published by Pearson Education, 2001. ISBN #0-201-72593-2.



- ✓ *Computer Science: An Overview* (7th or 8th edition), by J. Glenn Brookshear. Published by Addison Wesley, 2004. ISBN #0-32124726-4.




We do not recommend that you acquire any of the supplementary books at this time.

V. *Homework and Grading Policy*

The majority of the homework will be programming problems that require use of a Windows 98/2000/XP computer with a 500 mhz (or better) processor, a Macintosh OS X computer, or a UNIX timeshared computer (which can be accessed via a Mac or PC). These systems are available for student use at the Science Center (rooms B-14 & B-11, in the lower level, and rooms 226 and 229). At 53 Church Street there are computing facilities consisting of roughly 85 IBM PCs and 30 or so Macs. A small satellite computing facility located in the Sever Hall Grossman Library (on the third floor) may be available during weekday daytime hours as well.

If you own a personal computer with the right type of *Java* development environment (version 5.0), you may wish to copy all the lecture demonstration programs off of the Harvard network, and on to your own machine. At lecture and at section meetings we will explain how to accomplish this.

Computer Science S-111a is divided into 5 distinct “units” or parts. Each unit will require roughly 3 days of lecture. Each unit has a problem set worth 50 points. These five homeworks are each broken down further into two or three individual parts:

-  A pencil-and-paper assignment
-  A *main programming* assignment. These programs will be graded “critically” ... in other words, both for correctness as well as programming *style*.
-  A set of *supplementary programming* problems for which you will have a choice as to which you work on. You can earn up to an additional 5 points (maximum) of “extra credit” by solving more supplementary problems than required. These programs will be evaluated only on the basis of whether or not they “work.”

Electronic submission of all programming assignments is due PRIOR TO 9:00 PM on the date announced. Refer to the syllabus at the end of this document. No paper or printouts need be turned in, except

in the case of paper-and-pencil exercises.

Collaboration: You will learn a lot by discussing assignments with your classmates, but confine yourself to exchanging ideas, not portions of programs. What you hand in must be your own work! In doing assignments, feel free to copy and modify anything relevant that you find in the course software on the network, but NEVER copy anything from another student's program, even with the intention of making major changes to it. *Having said this, we may (from time to time) allow each student to collaborate on particular homework problems with one other currently enrolled CSCI S-111a student!*

Lateness Policy: All problem sets for a given unit are included in the *Course Packet*, which you should purchase right away. Due dates will be strictly enforced, and no late assignments will be accepted, subject to the following caveats:

- If you are ill, or should some other emergency arise, you must make special arrangements with your TF to turn your work in late.
- We will allow the following exceptions: once during the course you may hand in a programming assignment at most 24 hours late without being penalized. Don't fritter away these extensions too early — you may need them when the going gets tough!
- Any other homework (including all pencil-and-paper assignments) can be turned in up to 24 hours late for a 10% deduction.

Do **not** attempt to finish up a homework during lecture or during section! At section meetings, your assigned TF will return your graded homework to you. Please, please

Do NOT fall behind on homeworks!!!

Just as you cannot expect to learn how to drive a car by reading about it or by watching other people do it, the same holds true for programming a computer. Do your work on time — this is one course

you simply cannot “cram” for at the last minute, so don't even try! We cannot stress this strongly enough. Remember that the homework will be quite time consuming, so please reconsider your other commitments before you decide to continue with Computer Science S-111a.

Please note that **extensions** of time in CSCI S-111a will be granted only in extreme circumstances (e.g., illness), and only when appropriate documentation is provided. Such cases must be cleared (as early as possible) with your teaching fellow, who will consult with Dr. Leitner.

We will require ELECTRONIC SUBMISSION of all work, and ask that you follow these guidelines when submitting:

- Include your full name, course, date and assignment number in a comment line at the beginning of every program.
- For the programming assignments, be sure you have submitted your files using the *UNIX* **submit** command. The precise procedure will be explained to you.
- Remember that your homework will be graded on the basis of *correctness* and *clarity* (e.g., the use of meaningful identifiers, appropriate indentation of statements, modular design, comments). We'll have more to say about this later on.

Exams

In addition to the 4 quizzes, there will be a standard (paper-and-pencil) 3-hour, open-book final exam on **Friday, July 22**.

Grading

There will be 500 points of credit to be earned: 250 points from the 5 homework assignments, 100 points from the 4 quizzes, and 150 points from the final exam.

If you earn 400-450 total points, that will probably translate into some sort of “B” grade, while 450 or more points will likely be some sort of “A”. Graduate-credit students will have their final grade computed a bit differently from undergraduate-credit students because of some additional required work, which we will say more about later on.

V9. Syllabus and Schedule

The following outline will give you an idea of how Computer Science S-111a will progress. The material is divided into five basic “units”:

- Unit 1 — *JKarel*: Introduction to top-down design, editing and debugging, and basic control structures (conditional statements, iteration, while loops and loop invariants). Based upon Richard Pattis’ *Karel the Robot*, *JKarel* will enable students to absorb a large number of useful, important, and sophisticated programming concepts quickly.³
- Unit 2 — Introduction to *Java*: The structure of a standard *Java* program; integer and floating-point variables and constants, and arithmetic operators. Standard output using **System.out**; keyboard input using the *SavitchIn* methods. Parameterless methods, conditional statements, **for** loops and **while** loops. Notion of integer “overflow.” Introduction to classes, class instantiation, instance variables and methods. Computing the Fibonacci sequence. Printing interesting patterns; testing programs. The *boolean* data type; formulating complex conditions using the logical operators.
- Unit 3 — The **char** data type; **do-while** loops, **break** and **continue**; the **switch** statement; Class methods and variables; Monte-Carlo simulation using “random numbers.” Parameter-passing mechanisms. Constructors, getter and setter methods. Scope and lifetime of variables and identifiers.
- Unit 4 — Simple recursion; use of library functions for character graphics. Classes revisited as a means for creating data abstractions. Single and multi-dimensional arrays. *Strings*, *StringBuffers*, and the string manipulation library functions.
- Unit 5 — Inheritance, exceptions, and basic file I/O. Interfaces. Introduction to GUIs using the “Swing” classes.

Below is a more detailed view of the schedule, including the dates on which unit quizzes are held. There is no lecture on most Wednesdays.

³ Paraphrased from Pattis, Richard E. 1981. *Karel the Robot: A Gentle Introduction to the Art of Programming with Pascal*. New York: John Wiley & Sons, p. vii (the Preface).

		Homework			
Date	Lecture	P & P	Main	Supp	Unit Tests
Mon, 27-Jun-05	Unit 1, day 1				
Tue, 28-Jun-05	Unit 1, day 2				
Wed, 29-Jun-05	Unit 1, day 3	#1			
Thu, 30-Jun-05	Unit 2, day 1		#1		
Fri, 1-Jul-05	Unit 2, day 2			#1	#1 at NOON
Mon, 4-Jul-05	<i>No lecture!</i>	#2			
Tue, 5-Jul-05	Unit 2, day 3				#1 retest at 1 PM
Wed, 6-Jul-05	<i>No lecture!</i>		#2		
Thu, 7-Jul-05	Unit 3, day 1			#2	
Fri, 8-Jul-05	Unit 3, day 2				#2 at NOON
Mon, 11-Jul-05	Unit 3, day 3	#3			#2 retest at 1 PM
Tue, 12-Jul-05	Unit 4, day 1		#3		
Wed, 13-Jul-05	<i>No lecture!</i>			#3	
Thu, 14-Jul-05	Unit 4, day 2	#4			#3 at NOON
Fri, 15-Jul-05	Unit 4, day 3		#4		#3 retest at 1 PM
Mon, 18-Jul-05	Unit 5, day 1			#4	
Tue, 19-Jul-05	Unit 5, day 2	#5			#4 at NOON
Wed, 20-Jul-05	<i>No lecture!</i>		#5		#4 retest at 1 PM
Thu, 21-Jul-05	Unit 5, day 3			#5	
Fri, 22-Jul-05	<i>Final Exam</i>				

Beatles in the new millennium, for computer people!

Yesterday

Yesterday, All those backups seemed a waste of pay.
 Now my database has gone away.
 Oh I believe in yesterday.

Suddenly, There's not half the files there used to be,
 And there's a milestone hanging over me
 The system crashed so suddenly.

I pushed something wrong
What it was I could not say.

Now all my data's gone
and I long for yesterday-ay-ay-ay.

Yesterday, The need for back-ups seemed so far away.
I knew my data was all here to stay,
Now I believe in yesterday.

Eleanor Rigby

Eleanor Rigby
Sits at the keyboard and waits for a line on the screen
Lives in a dream
Waits for a signal
Finding some code
That will make the machine do some more.
What is it for?
All the lonely users, where do they all come from?
All the lonely users, why does it take so long?

Guru MacKenzie
Typing the lines of a program that no one will run;
Isn't it fun?
Look at him working,
Munching some chips as he waits for the code to compile;
It takes a while...

All the lonely users, where do they all come from?
All the lonely users, why does it take so long?

Eleanor Rigby
Crashes the system and loses 6 hours of work;
Feels like a jerk.
Guru MacKenzie
Wiping the crumbs off the keys as he types in the code;
Nothing will load.

All the lonely users, where do they all come from?
All the lonely users, why does it take so long?

Unix Man (Nowhere Man)

He's a real Unix Man
Sitting in his Unix LAN
Making all his Unix plans For nobody.

Knows the blocksize from du(1)
Cares not where /dev/null goes to
Isn't he a bit like you And me?

Unix Man, please listen(2)
My lpd(8) is missin'
Unix Man The wo-o-o-orld is at(1) your command.
He's as wise as he can be
Uses lex and yacc and C
Unix Man, can you help me At all?

Unix Man, don't worry
Test with time(1),
don't hurry
Unix Man
The new kernel boots, just like you had planned.

He's a real Unix Man
Sitting in his Unix LAN
Making all his Unix plans For nobody ...
Making all his Unix plans For nobody.

Write in C ("Let it Be")

When I find my code in tons of trouble,
Friends and colleagues come to me,
Speaking words of wisdom: "Write in C."

As the deadline fast approaches,
And bugs are all that I can see,

Somewhere, someone whispers: "Write in C."

Write in C, Write in C, Write in C, oh, Write in C.
LOGO's dead and buried,

Write in C.

I used to write a lot of FORTRAN,
For science it worked flawlessly.
Try using it for graphics!
Write in C.

If you've just spent nearly 30 hours,
Debugging some assembly,
Soon you will be glad to
Write in C.

Write in C, Write in C, Write in C, yeah, Write in C.
BASIC's not the answer. Write in C.

Write in C, Write in C Write in C, oh, Write in C.
Pascal won't quite cut it. Write in C.

Something

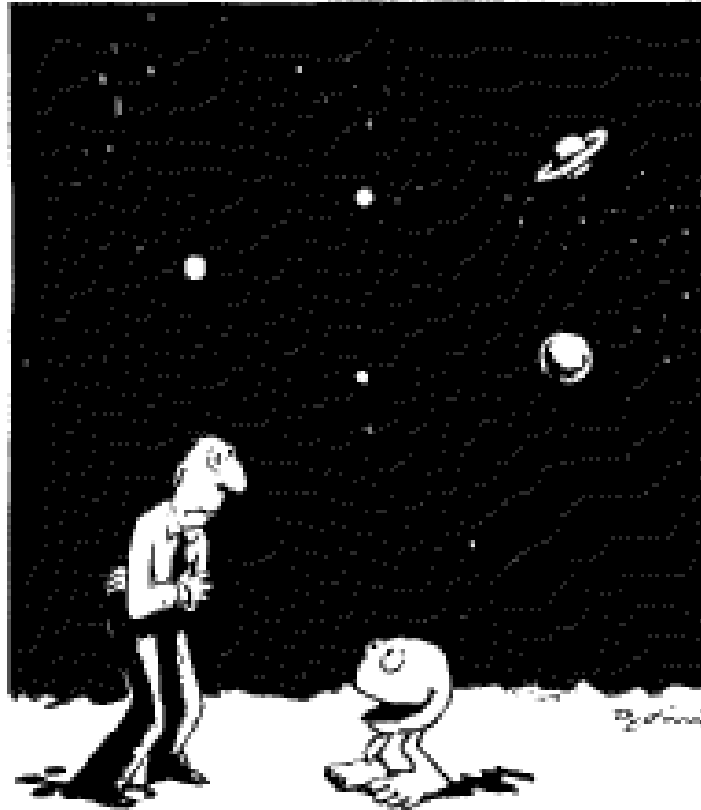
Something in the way it fails,
Defies the algorithm's logic!
Something in the way it coredumps...
I don't want to leave it now I'll fix this problem somehow

Somewhere in the memory I know,
A pointer's got to be corrupted.
Stepping in the debugger will show me...
I don't want to leave it now I'm too close to leave it now

You're asking me can this code go?
I don't know, I don't know...
What sequence causes it to blow?
I don't know, I don't know...

Something in the initializing code?
And all I have to do is think of it!
Something in the listing will show me...

I don't want to leave it now -- I'll fix this tonight I vow!



Think back ... which keys did you press?

Important Information on Compiling and Submitting Assignments

Although you are free to develop programs on your own personal computer, you are required to have your programs compile and run under the UNIX platform, and you must also submit these programs electronically via the UNIX operating system. Below we describe how to use the **make** utility for the development and submission of programming assignments.

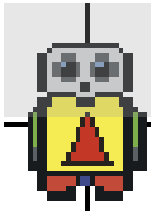
The **UNIX make** utility is used by programmers to manage software projects. We will use it to help compile, print, and submit programming assignments. To run the make program, type **make** at the prompt, followed by optional arguments as described below.

When starting on the main programming assignment for a new unit, follow these

steps:

- Make a new directory for the assignment. For example, for unit 3, you would type:
% **mkdir unit3**
- Change into the new directory. For example, type: **cd unit3**
- Copy the assignment files from the **libs111** account. For example, type:
% **cp ~libs111/unit3/assign/***.
- The directory will now contain files in which you will have to fill in some code. For example, if you need to write the programs *foo* and *bar*, you will need to edit the files *foo.cc* and *bar.cc*.
- When you are done with the assignment, submit your code by typing: **make submit**. Note that you can submit multiple times, in which case only the last submission counts. Note also that the submission time is recorded electronically - make sure that you submit your work before the deadline.
- If you wish to print the code for yourself, type: **make print**. This prints your code to the default printer in the basement of the Science Center. However, you will not need to turn in hardcopy for grading.

If you have any questions about this procedure, please don't hesitate to ask any member of the course staff!



Best of Luck from your friend, JKarel