CSCI 152 Computer Science I Using Java  
David M. Keil, Framingham State University, Fall 2013  
SYLLABUS DRAFT 4/13

Invitation
Why did you choose to take a course in computer programming and software development? Do you like to solve puzzles and mysteries? Do you plan to build computing systems as a career, or to be a user, or to determine requirements for systems that others build?

Are you attracted by the challenge of analytical reasoning and complex problem solving, the top skills that employers are reported to want college graduates to have more of in 2013 (Hart Research Associates report to the Association of American Colleges and Universities)?

In any of these cases, this course offers ways to understand how professionals construct reliable computing systems.

To deliver what it offers, this course emphasizes demonstrations and discussion in the classroom, including collaboration among learners. The approach emphasizes active learning within an environment that relies on independent thinking and problem solving.

Course description (FSU catalog)
An introduction to problem solving using the Java programming language. The course stresses algorithms, object-oriented programming in graphical environments, documentation, testing, and debugging. Topics include hardware basics and number systems, classes, methods, control structures, types, virtual-machine concepts, Internet and client-server computing, human-computer interaction, social, professional, and ethical issues, and general features of programming languages.

Prerequisite: CSCI 120 Introduction to Information Technology. Corequisite: MATH 123 College Algebra or minimum score of 2 on the FSC math placement examination.

To contact me
Office hours (Hemenway Hall 318A):
M 12:30-1:30 p.m., W 9:30-10:30 a.m.,  
Th 1:30-2:20 p.m., others by appointment  
Telephone: (508) 626-4724  
Email: dkeil@framingham.edu  
URL: www.framingham.edu/~dkeil

Overview
Computer science is a scientific discipline with mathematical and empirical aspects. Computation is the manipulation of abstract symbols, but can only occur in the physical world. We use abstraction and reasoning when we design solutions to computational problems, and we test these solutions on actual computing devices.

Computer science is much more than programming, but some concepts of computer science can be presented through computer programming. You will learn by hands-on programming activities, mainly in Java, an object-oriented programming language widely used in the software industry. In each topic of this course, the central focus is designing solutions to problems. Almost all require loops (repetition).

This course looks at, or provides some knowledge of, a number of languages, and encourages a comparison among them. Among these, in addition to Java, are the algebraic language of spreadsheet formulas; the languages of database querying; the HTML web markup language; Javascript, which is embedded in HTML web pages; machine and assembler codes; and design notations such as pseudocode, flowcharting, module diagrams, and the UML specification language.

Programming begins with an emphasis on design. We direct attention to design of both algorithms and non-algorithmic, interactive computational processes. Students will learn to trace the operations diagrammed in flowcharts and to design algorithms using flowcharts. The three control structures used in algorithm design are sequences, branches and loops.

We will introduce some hardware concepts, centered around the notion of a processor that fetches and executes instructions in machine language. In discussing hardware we also look at the binary (0,1) representation of data.

In this course, you will learn problem specification, solution design, and the coding, documenting, testing, and debugging of programs.

The notion of data type (e.g., integer, real, string, class) is part of CS I. The central organizing element of the object-oriented design and programming paradigm,
and of Java programs, is the class. A class is an abstract specification of objects: data items with associated attributes and operations. A method is a Java subroutine that implements an operation for a class.

This course is designed to challenge you and may offer and demand new ways of thinking about problem solving, including logical inference, the creation of data types, and the writing of subprograms.

It emphasizes hands-on learning on a computer. Our tools will include a Java compiler (Sun’s Java Development Kit). Exercises will require extensive use of a compiler.

Strongly recommended readings


Handouts with examples and explanation.

Slides: As hard copy and at course web site.

I like this textbook because studying text material related to the course provides explanations that learners need for understanding. In college courses, textbook work is expected outside of classroom time. The slides and the study questions provide a summary and a guide to which concepts are most important to this course.

Basic skills (presented in CSCI 120, Introduction to Information Technology)

1. Explain and use spreadsheet formulas.
2. Explain and use built-in functions in a spreadsheet.
3. Recognize or explain simple database concepts.
4. Show knowledge of the basic terminology of computer hardware.
5. Show knowledge of the basic terminology of computer software.
6. Identify the steps in the software development life cycle.
7. Explain the notion of an algorithm.
8. Trace a looping and branching computation specified in a flowchart or pseudocode.
9. Explain the concepts of syntax, logic errors, and debugging.
10. Explain how technological changes have created new ethical, social, and legal issues.

To review CSCI 120 materials, see www.framingham.edu/~dkeil/iit-mats.htm.

CS I learning objectives

0a. Participate in class activities throughout the semester
0b. Solve problems as part of a team
0c. Present results in the classroom
0d. Present written results
0e. Show knowledge of facts and concepts
0f. Summarize the semester’s learning
1. Describe and apply principles of system specification and design
2. Explain and use the binary system of numerals and simple machine operations
3. Write, document, and test a simple Java program
4. Use standard Java data types in documented, tested programs
5. Use and debug a variety of Java branch and loop statements
6. Define and test Java methods and classes with object-oriented features
7. Define and safely manipulate arrays, including collections of objects
8. Explain technical and social issues associated with professional software development

Classroom format and grades

The essay, “What we do in my classroom,” is part of this syllabus. It has guidelines for assignments, collaboration, and grading. As it explains, for each topic, we have presentations, group work, discussion, assignments, and quizzes. Assigned work and quiz questions help to assess attainment of learning objectives.

Our classroom environment emphasizes active inquiry, participation, respect, and support among all participants. Learning is seen as the interactive construction of knowledge by the learner. We ask each other questions and investigate problems together.

Work includes small groups and blackboard work and report backs from each student. A semester project brings together the learning from the different topics and assignments. Frequent assignments and quizzes monitor progress and enable second chances.

Grades assess learning based on attainment of the stated objectives of the course. I score each item of work, or grading criterion, on a scale of 0 to 1.0.
Programming project
You will build a Java programming project step by step as the course proceeds, making use of control structures, methods, arrays, file input/output, and class design. It will provide experience in coding, testing, and documentation of specifications, design, and code.

Semester grading weights
The following categories group course objectives and outcomes (see previous page), which are assessed by means of exercises, quizzes, exams, and records of classroom discussion, presentations, and group work.

<table>
<thead>
<tr>
<th>Category</th>
<th>Weight</th>
</tr>
</thead>
<tbody>
<tr>
<td>Application of concepts</td>
<td></td>
</tr>
<tr>
<td>core topic objectives</td>
<td>35</td>
</tr>
<tr>
<td>other topic objectives</td>
<td>10</td>
</tr>
<tr>
<td>Knowledge of facts</td>
<td>10</td>
</tr>
<tr>
<td>Written contribution</td>
<td>20</td>
</tr>
<tr>
<td>Presenting results in person</td>
<td>10</td>
</tr>
<tr>
<td>Group activity</td>
<td>5</td>
</tr>
<tr>
<td>Summary and reflection</td>
<td>5</td>
</tr>
<tr>
<td>Attendance</td>
<td></td>
</tr>
<tr>
<td></td>
<td>5</td>
</tr>
<tr>
<td></td>
<td>100</td>
</tr>
</tbody>
</table>

Backing up data
All electronic files that students submit for this course should be kept in at least two physical locations. This means routine backup of all work to the FSC network (your Y: drive), to a memory stick, or to your Blackboard Content Area. This matters. Loss of data is not an acceptable reason for missing assignments.

Tutoring
Tutors who are eager to help CS I students are available via the CS Department and CASA.

Accommodations
“Students with disabilities who request accommodations are to provide Documentation Confirmation from the Office of Academic Support within the first two weeks of class. Academic Support is located in the Center for Academic Support and Advising (CASA). Please call (508) 626-4906 if you have questions or if you need to schedule an appointment.” (See http://www.framingham.edu/CASA/Accommodations/accomm.htm.)
# Course Plan

<table>
<thead>
<tr>
<th>Dates</th>
<th>Topic</th>
<th>Reading</th>
</tr>
</thead>
</table>
| 9/2-9/4   | Introduction                                       | *Handout*
| 9/9-9/24  | 1. Problem solving and system design              | Savitch-Carrano, Sec. 1.1; *Handout*                                    |
| 9/23-10/4 | 2. Computer organization and hardware             | *Handout*                                                               |
| 10/2-10/9 | 3. Introduction to Java                            | Savitch-Carrano, Sec. 1.2-1.3, 2.1, 2.3-2.4, Appendix 1; *Handout*     |
| 10/14     | **Optional objectives quiz**                       |                                                                        |
| 10/15-10/22 | 4. Standard Java data types: numeric, character, string, file stream | Savitch-Carrano, Sec. 2.2, 10.1-10.2; example handouts                  |
| 10/23-11/2 | 5. Branch and loop statements                      | Savitch-Carrano, Chs. 3-4; example handouts                             |
| 11/6-11/18 | 6. Methods and classes                             | Savitch-Carrano, Chs. 5-6; example handouts                             |
| 11/19     | **Optional objectives quiz**                       |                                                                        |
| 11/20-12/2 | 7. Arrays and collections                          | Savitch-Carrano, Ch. 7; example handouts                               |
| 12/3-12/9 | 8. Building GUls and file-maintenance applications (advanced topic) | Savitch-Carrano, Ch. 8; Sec. 1.4, 2.5, 3.4, 4.3, 5.4, 6.8, 7.6, 8.4, 10.6 |
| 12/10-12/11 | Review                                           |                                                                        |
| 12/14     | Pre-final quiz (problem solving, topics 1-8)       |                                                                        |
| Thurs., 12/17, 9:00am | Final exam (multiple-choice and short answer, topics 1-8) |                                                                        |

3. D. Keil, “Problem solving and system design”
4. D. Keil, “Using Java development tools”; example handouts