# CSCI 252: Computer Science II

David M. Keil, Framingham State University, Spring 2014

# SYLLABUS

## Invitation

Now that you've taken CS I and have some skills with the Java language, would you like more expertise and confidence in software development skills? Would you like to have the capability to build larger programs? Does *object-oriented design* interest you?

## Course description (FSU catalog)

An intermediate programming course that emphasizes debugging, documentation, and modular and object-oriented design with tools such as the Unified Modeling Language. Topics include event-driven programming, string and array manipulation, sorting and searching, file operations, dynamic memory allocation, inheritance, polymorphism, and exception handling.

*Prerequisites:* MATH 200 Precalculus (may be taken concurrently) and CSCI 152 Computer Science I Using Java.

## Course overview

This course continues CS I's study of problem solving, algorithm design, and debugging. The central content of CS II consists of new programming and programming-language concepts and intermediate-level software development skills such as program design and debugging.

We will focus on object-oriented design and on four key concepts, *encapsulation, containment, inheritance*, and *polymorphism.* We'll apply them in Java.

CS II starts with an expanded review of material from the CS I introductory course: Java data types, loops, debugging, and files.

This course introduces *nested loops* and *searching and sorting of arrays*. CS II students will acquire skills in building and verifying larger programs.

*Debugging* is a crucial programming skill, developed in this course. We will emphasize techniques like *tracing* that help locate errors.

We will define *collections* stored on disk and in memory, including *arrays of objects* to implement the database concept of a *relation*.

The main principles on which this course is based are: structured design; object-oriented design, including containment and inheritance; nested loops; concern for time efficiency.

We will connect the topics together with a programming project that will be part of the ongoing course activities.

CS II is a rigorous and demanding course, with heavy emphasis on lab work on intermediate-level software-development problems.

## Textbooks

Reading a textbook makes a *lot* of difference, because everyone needs *explanations* of concepts.

Use your CS I textbook or another Java textbook by an academic publisher. The course plan (p. 3) cross-references course topics with three Java texts with which I'm familiar.

## Meeting times

We meet Monday and Wednesday, 8:30-10:20, in Hemenway Hall 229.

## To contact me:

Office hours (Hemenway Hall 318A):
Mon. 10:30-11:30 a.m.; Tue. 5:30-6:30 p.m.;
Wed. 3:30-4:30 p.m.
Telephone: (508) 626-4724
Email: *dkeil@framingham.edu*
URL: *www.framingham.edu/~ dkeil*

I like talking with students about what we're studying. Even if everything is clear enough, please check in with me at least twice in the semester.

## How the course will deliver what it offers

My goal is to create a *natural critical learning environment*. In CS II, this means working on *solving realistic software problems.* We all learn at our own pace, and we are all together in this class learning.

For each of the six topics, I'll speak about the topic for a few minutes, with slides and with examples to compile, run, and discuss; we'll make space for group work; and we'll have in-class and out-of-class written exercises. *Practice exercises* and *quiz questions* help me track what students learn.

For each topic, we'll have two to four two-hour sessions. I'll ask you to solve some topic practice problems and to let me see your solutions by the end of the session *before* the last one on the topic. I'll look at them and provide comments.

In the *last* session on the topic, we'll have a review, with problem solving by students; a multiple-choice quiz; and a set of problems in quiz form.

For every topic, there'll be two or three more chances to solve problems in make-up quizzes. What I track is a student's *best* work on problems that assess course objectives.

See the essay, "What we do in my classroom."

## Programming project

Step by step, as the course proceeds, you will build a two-part Java programming project that will make use of control structures, debugging, class design, arrays, and file input/output. It will provide experience in coding, testing, and documentation of specifications, design, and code. The first part is a file-maintenance program that manages a collection that you will specify. The second part is an even larger program, either to design and code your own application, or to modify a larger application written by others; an even larger project with documented group contribution is an option.

## Learning outcomes

The course objectives are summarized and measured by several *learning outcomes* per topic; see next page. Some outcomes, mostly from CS I, are *essential* for success. All students who pass the course will have shown success with these capabilities.

I will work to help students reach the outcomes listed on the back page. *Outcomes* are a kind of very specific learning objective. Some of our outcomes are especially central; these I call "priority outcomes" and they are indicated by asterisks.

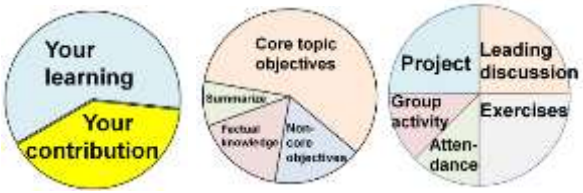
If you're tracking my opinions, it may help you to use that sheet as a score card, writing on it the numbers (from 1 to 4) that I write beside your quiz answers. Update these numbers as you answer questions on make-up versions of the quizzes; your highest score on an outcome is the one that matters.

## Topic objectives

1. Define and test Java methods and classes with object-oriented features
2. Define and safely manipulate arrays, designing nested loops and applying search and sorting algorithms
3. Explain, design, and implement a multi-class application that manages a disk-based collection
4. Explain and implement the notions of an inheritance hierarchy and of polymorphic behavior
5. Explain event-driven GUI development and use Java graphics libraries to implement such a GUI
6. Describe concurrent features of mobile and web environments and implement concurrency with Java threads

## Grading and assessment of learning

In evaluating the work of individual students, evidence of two kinds of accomplishment matter to me: *learning of course objectives*, and *contribution to the learning of others*. The graphs below show their relative importance to me and the relative importance of their components or methods of assessment.




Overall Learning Contribution

I evaluate answers to quiz questions, as well as project work and some exercises, using the rubric below. 4 means, roughly, "excellent"; 3 means "good"; 2 means "OK"; 1 means "weak success." An answer without one of these scores is considered not yet successful; try again.

| Code | Meaning | % |
|---|---|---|
| 4 | Solves problem thoroughly and accurately. Applies relevant concepts adeptly and insightfully. Fully supports claim of mastery of outcome. | 100 |
| 3 | A mostly successful solution with some omissions or errors. Generally accurate application of concepts. Gives strong support for claim of success with outcome. | 87 |
| 2 | Solution shows some grasp and application of relevant concepts, reflecting significant partial achievement of outcome. | 73 |
| 1 | A solution that shows some idea about relevant concepts, meeting minimum standards for outcome. | 61 |

## 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

| | | Relevant chapters | | |
|---|---|---|---|---|
| **Dates** | **Topic** | **Horstmann[1]** | **Reges[2]** | **Downey[3]** |
| 1/22 -2/3 | *Introduction and review:* data types; algorithms; files | 1-2, 4-6 | 1-6 | 1-2, 4.1-4.5, 7-8, Appx. B-D |
| 2/5 – 2/19 | 1. Class and method design | 3, 8, 11-13 | 8, 12 | 3, 4, 6, 9, 11 |
| 2/24 – 3/5 | 2. Arrays | 7, 14 | 7, 10, 13 | 12 |
| 3/10 – 3/12 | 3. Collections | | 11, 15-16 | 13-15 |
| 3/24 – 3/26 | 4. Inheritance and polymorphism | 9, 18 | 9 | |
| 3/31 | *Review and quiz make-ups* | | | |
| 4/2 – 4/7 | 5. Event-driven GUIs and Java graphics | 10, 15-16, 19 | 3G | Appx. A |
| 4/9 – 4/14 | 6. Concurrency in mobile and web computing | 21-22 | | |
| 4/16 – 4/30 | *Summary and review* | | | |
| Fri., 5/9, 8:00-11:00 | *Final exam (presentations)* | | | |

Ver. 1/17/14

[1] Cay Horstmann *Big Java Early Objects*, 5th ed.
[2] Stuart Reges and Marty Stepp, *Building Java Programs*, 3rd ed., Pearson, 2014
[3] Allen Downey, *Think Java*, 2012, free download.

# Subtopic outcomes for CSCI 252 Computer Science II

## Expanded CS I review

____ 0.1a Compile and test a Java program**
____ 0.1b Use logical operators in a program*
____ 0.1c Evaluate an expression that uses logical operators*
____ 0.1d Debug a program with type errors *
____ 0.1e Use bitwise operators
____ 0.2a Trace a looping flowchart**
____ 0.2b Design a looping algorithm**
____ 0.2c Argue for the correctness of a loop design**
____ 0.2d Solve a numeric loop problem in Java**
____ 0.2e Solve a loop problem with strings**
____ 0.2f Trace a Java loop**
____ 0.2g Debug a defective loop**
____ 0.2h Describe an instance of the testing and debugging process*
____ 0.3a Explain streams and sequential file I/O**
____ 0.3b Read a file using a loop**
____ 0.4a Describe Java syntax for statements**
____ 0.4b Describe Java syntax for expressions**
____ 0.4c Explain how the Java virtual machine works*

## 1. Class and method design

____ 1.1a Explain procedural abstraction**
____ 1.1b Define a Java method**
____ 1.2a Explain method signatures, overloading, and scope of variables*
____ 1.2b Derive a loop from a recursive function definition
____ 1.2c Write a method with parameters and return values**
____ 1.2d Debug a method
____ 1.3a Describe Java data abstraction**
____ 1.3b Contrast memory allocation for simple types and objects**
____ 1.3c Define a Java class**
____ 1.4a Describe Java encapsulation, cohesion, decoupling*
____ 1.4b Write and document a class with interface and implementation
____ 1.4c Describe class debugging concepts
____ 1.4d Test and debug a class*
____ 1.4e Locate a fault in a multi-method class
____ 1.5a Explain exception handling*
____ 1.5b Use exceptions

## 2. Arrays

____ 2.1a Describe Java arrays**
____ 2.1b Define and use an array**
____ 2.2a Write array code with boundary checking*
____ 2.2b Write a simulation using a random number generator
____ 2.2c Write an encryption/ decryption program
____ 2.2d Define a two-dimensional array
____ 2.3a Give the output of a nested loop*
____ 2.3b Debug a nested loop*
____ 2.3c Explain a search algorithm
____ 2.3d Search an arra y*
____ 2.4a Explain a sorting algorithm
____ 2.4b Sort an array*

## 3. Collections

____ 3.1a Explain principles of object-oriented design*
____ 3.2a Explain what a collection is*
____ 3.2b Define and test a collection*
____ 3.2c Define and use an iterator
____ 3.3a Explain the design of file-maintenance applications*
____ 3.3b Write and test a file-maintenance application*
____ 3.3c Describe random-access files
____ 3.4a Distinguish references from objects*
____ 3.4b Explain linked lists
____ 3.4c Define a linked-list class

## 4. Inheritance and polymorphism

____ 4.1a Distinguish containment from inheritance*
____ 4.1b Design an inheritance hierarchy
____ 4.1c Explain multiple inheritance and the diamond inheritance problem, with solutions
____ 4.2a Define a derived class*
____ 4.2b Explain overloading, overriding, and shadowing
____ 4.3a Explain polymorphism, late binding and virtual methods*
____ 4.3b Implement polymorphism

## 5. Event-driven GUIs; graphics

____ 5.1a Explain event-driven programming*
____ 5.1b Write event-driven code
____ 5.2a Describe elements of a graphical user interface*
____ 5.2b Use an application class in an application framework
____ 5.3a Describe Java graphics tools
____ 5.3b Write a graphics application

## 6. Concurrency

____ 6.1a Distinguish concurrency from serial processing*
____ 6.1b Describe multi-threading
____ 6.1c Use Java threads
____ 6.2a Describe features of software for mobile computing
____ 6.2b Design a multi-threaded robotic program
____ 6.3a Describe issues in web software development
____ 6.3b Write a Java applet
____ 6.4a Describe ethical issues for computer professionals
____ 6.4b Defend a position on an ethical issue

* *Priority objective*
** *Essential objective*

`