CS 1A Object-Oriented Programming in Java

Course Syllabus

jump down this page to: schedule | final grades | assignment grades | late policy | collaboration | disabilities | getting started | style conventions

Instructor

Dave Harden
Email: dharden@santarosa.edu
Response Times: I almost always respond to emails and discussion posts within 24 weekday hours, always within 48 weekday hours. I rarely respond on weekends or evenings. My primary times for communication are MWF mornings.

Please use the class discussions for all questions whenever possible. For private questions that are not appropriate for the class discussion, email me. Do not attempt to contact me using the Canvas messaging system (or "inbox"). I don't monitor the Canvas inbox. **Please include "CS 1A" in the subject when you email me!!**

Two Most Important Announcements!

- You **must** post an introduction in the "introductions" topic of the class forum by Monday, Jan 14, or I am required to drop you from the class. When you begin working your way through the modules, you'll see that this is the first task for week 1.
- Absolutely no assignments will be accepted for any reason after Wednesday, Mar 20.

Textbook

Title: Introduction to JAVA Programming Comprehensive Version 11th Edition
Author: Y. Daniel Liang

As far as I know, the 10th edition should be fine, and I have no objections to you using it, but you are responsible for any discrepancies that might arise.

Computer Science Department Announcements

Opportunities for CS students is a blog that contains announcements of internships, scholarships, free software offers, pertinent public lectures, etc. Announcements will be posted here often during the quarter.

The STEM Center, in room 4213, has CS tutors at various times each day. It is also a place on main campus where students without their own computers can do their lab work, and it also provides online tutoring for CS students. I strongly encourage you to take advantage of this valuable resource. See the STEM Center Website for more information.

Official Course Outline and Student Learning Outcomes

You can access the official course outline of record for all CS courses at http://www.foothill.edu/schedule/catalog.php. From that page, select Dept: Computer Science -> Search, and from there, select any CS course whose official outline you want to review.

Here is a link to student learning outcomes for this and other CS courses.

Tentative Schedule

Assignment	Topic	Text Reading	Suggested Start Date	Assignment Due Date
a1	Java Basics 1	ch 1	Monday, Jan 7	Monday, Jan 14
a2	Java Basics 2	ch 2	Monday, Jan 14	Monday, Jan 21
a3	Decisions	ch 3&4	Monday, Jan 21	Monday, Jan 28
a4	Loops	ch 5	Monday, Jan 28	Monday, Feb 4
a5	Methods	ch 6	Monday, Feb 4	Monday, Feb 11
a6	Classes 1	ch 9	Monday, Feb 11	Monday, Feb 18
Midterm	Thru Methods	ch. 1 - 6	Monday, Feb 18	Monday, Feb 18
a7	Classes 2	ch 9	Monday, Feb 18	Monday, Feb 25
a8	Arrays 1	ch 7	Monday, Feb 25	Monday, Mar 4
a9	Arrays 2	ch 7	Monday, Mar 4	Monday, Mar 11
a10	GUI's	NA	Monday, Mar 11	Monday, Mar 18

Final	6 - 10	7, 9	Monday, Mar 25	Monday, Mar 25

Final Grades

Your final score will be made up of the following components.

Component	**points each**	**points total**
Assignments (9)	95	855
Discussions (9)	5	45
Midterm	40	40
Final	60	60
total		**1000**

Grades will be assigned as follows: 900 points for an "A", 800 points for a "B", 700 points for a "C", 600 points for a "D". Grades of + and - are not given

Assignment Grades

Any program submitted that does not work as specified in ways that are more than trivial will receive a score of 0. Your scores on programs will be based only on issues of style and presentation. In order to understand how to get a good score you must read the Style Conventions section of this document carefully. You should also pay close attention to sample solutions that are given and instructions in the lessons and text. Because of this emphasis on issues of style and presentation, students are often surprised at their low scores on programming exercises. Be careful not to let this be you. A working program may receive a failing score!

Programs will be scored according to the percentages in the following table. Note that the number in the first column corresponds to the number in the Style Conventions section, which appears later in this document.

1 Comments	20%
2 Appearance (e.g. Whitespace, Wraparound)	10%
3 Identifier Names	10%
4 Decomposition	20%
5 Indentation	10%
6 Simple Code/No Repeated Code	20%
7 Miscellaneous	10%

Exams

There will be a midterm and a final. Both are taken online, and you can take each exam at any time of your choosing during the day on which it is scheduled (see the schedule). Once you start, the midterm must be completed within 60 minutes, and the final must be completed within 80 minutes, without exception, so ensure that you will not be interrupted once you begin. All exams are multiple choice with possibly one matching question. The midterm covers the topics in assignments 1 through 5. The final is comprehensive but strongly emphasizes the topics in assignments 6 through 9. Assignment 10 is not covered on the final.

You will take the exams on the honor code. The tests are available for one full day for your convenience, but the validity of the tests relies heavily on your academic integrity. Don't take advantage of the flexibility by sharing questions with students who have not taken the test.

You are expected to simulate a class environment when you take the exams. The exams are open book and open notes, and you may even use your compiler, but you cannot receive any help from another person. The rules are summarized below. Email me if you have any questions:

- Allowed: Textbook
- Allowed: Notes
- Allowed: Past assignments
- Allowed: Compiling your answers
- Not Allowed: Browsing non-class websites
- Not Allowed: Accepting assistance from another person.
- Not Allowed: Sharing questions with other students after the test.

Late Policy

This late policy is for assignments only. Late exams are not accepted.

Assignments are due at 11:59pm on the date indicated in the schedule. However, assignments may be submitted up to 48 hours late with no penalty. This is is the "final deadline". This does not mean that the due date is extended! For example, if your assignment is not done by the original due date, and you get severely ill between the due date and the final deadline, so that you cannot complete your assignment by the final deadline, it will be considered late. In addition, failing to complete projects by the original due date will put you behind in the class, and may delay the grading of your assignment significantly. You should make every effort to complete the assignment by the original due date.

Beyond the final deadline, assignments will be accepted until 2 weeks after the original due-date (except that no assignments will be accepted after Wednesday, Mar 20). They will be considered late and will receive a 50% deduction, with no exceptions. Assignments are not accepted more than 2 weeks late.

To repeat: absolutely no assignments will be accepted for any reason after Wednesday, Mar 20.

Collaboration

I'm letting you know up front that I am very serious about detecting and penalizing inappropriate collaboration. Please decide now that you will not engage in this. Instead get help when you need it using course discussions, email to me, tutors, etc. Last quarter over 20 assignments were given grades of 0 and five students received an F in the course because of this policy.

Inappropriate collaboration first offense = ZERO on assignment and formal Academic Dishonesty Incident Report.

Inappropriate collaboration second offense = F in the course

Any variation of copying or collaborating on programming assignments, or parts of programming assignments, is prohibited. Every assignment must be 100% your own work.

You may not use even one line of code that you find online, even if you modify it. You may use websites for reference purposes (for example: how does a particular language feature work?). But you may not get information specifically related to a problem you are trying to solve (for example: what's an algorithm for reducing a fraction?). Don't ask for help from online forums. They will almost invariably do the problem for you, or give you bad information.

I recommend that you get all of the information you need for the course from the text and lessons. If you need help, ask in the discussion. As a Foothill student you can also use NetTutor for free. As often as not, information that you find online will lead you in the wrong direction anyway.

Other than participating in the course discussions, you may not work together on assignments.

Examples of specific actions that violate this policy:

- Using chegg.com or coursehero.com for any reason
- Using an online tutor
- Viewing another student's code
- Allowing another student to view your code
- Viewing or copying code online that is specifically related to a course assignment

If you submit code that shows similarity to another student's code or to code that is available online, you will be found in violation of this policy.

Learning Disabilities:

To obtain disability-related accommodations, students must contact Disability Resource Center (DRC) as early as possible in the quarter. To contact DRC, you may:

- Visit DRC in Room 5400
- Email DRC at adaptivelearningdrc@foothill.edu
- Call DRC at 650-949-7017 to make an appointment.

If you already have an accommodation notification from DRC, please contact Teresa Ong privately to discuss your needs.

Style Conventions

[How To Get Good Grades On Your Programs]

In the real world, programmers usually work in teams and often the company that they work for has very precise rules for what kind of style to use when writing programs. For this reason, and also to encourage good programming style, we will be adopting the following style conventions for this class. This is not to say that these rules represent the only good style for writing computer programs (although in most cases they do). After you finish this class, you may decide that you prefer a different style than what is required here. However, in order to get good grades on your programming assignments in this class, you must follow these guidelines.

Each style convention is labeled with a code in square brackets. This code refers to the first assignment in which you are required to follow this style convention. For example, if the style convention is labeled with [a3], that means you don't have to worry about following this convention until assignment 3. If the word "special" appears in the square brackets, it means that only those students who might use a Java feature not covered in this class need be concerned.

1. **Documentation:**

   A. Initial File Comment [a2]: Your programs should be well-documented. Each program should begin with an initial file comment. This comment should start by indicating your name, class, date, instructor, name of file, etc. Next it should describe in detail what the program does and how the code works. Any input expected from the user and any output produced by the program should be described in detail. **You should**

expect your initial file comment for the first few assignments to be at least 50 words, and by assignment 7 you should be seeing over 90 words.

Important local variables should be commented at their declaration. Aside from this, in most cases it should not be necessary to place comments in the body of a method. This usually clutters up your code and ends up making the method more difficult to read. If you find yourself needing to explain something in the middle of a method, perhaps you should look for a clearer way to write it!

B. General Advice [a2]: Your comments should be directed toward a reader who is an expert Java programmer. You should not explain features of the language!

C. Method Comments [a5]: Just above each of your method definitions you must provide a comment describing what the method does. **A simple method might have a 15 word comment, while a more complex method should have a comment of at least 50 words. Make sure to explain the role of each parameter in your method comments, and refer to them by name.**

2. **Appearance:**

   A. General [a1]: Use lots of whitespace (blank lines and spaces) to separate the different parts of your program!! When I look at your program my first impression should not be a page crammed with code. Get rid of wraparound. Put a blank line between your declarations and your statements. Put a space before and after each operator. Make sure your lines aren't too long, no more than 80 or 90 characters.

   B. With Methods [a5]: **Put 6 blank lines between method definitions.**

3. **Identifier Names:**

   A. General [a1]: Choose your identifier names very carefully. Variable names should precisely represent what the variable is storing. Do not use abbreviations unless you feel the identifier name would otherwise be so long that it would hinder the readability of your program. Don’t use one letter variable names except, perhaps, as a counter in a for loop.

   B. With Methods [a5]: Choose your method names so that as much as possible your program reads like English and the names describe precisely what the method does. Void method names should start with an action word (readString, getData, etc.).

4. **Decomposition** [a5]:

   Any time there is a sequence of statements in your program that performs a specific, nameable sub-task, you should consider making that sequence of statements into a method. A nice length for methods is about 10 lines, although they can be longer if they are simple (for example, lots of output statements) or if there is just no logical way to break it up. Consider making complex methods (for example, nested loops) even shorter. A goal: when you are done with your program, I ought to be able to look at any particular method and have a general understanding of what is does and how just at a glance.

   In this class you should rarely write a method longer than 10 lines.

5. **Indentation** [a1]:

Indents must be exactly 4 spaces.

You may follow the indentation scheme used in the textbook or you may use the scheme used in the lessons. No others. For example, every statement must appear on a line by itself, every close curly brace must appear as the first (or only) item on a line, and every open curly brace must appear as the last (or only) item on a line.

6. **Simple Code/No Repeated Code** [a1]:

   Make sure that your code is as simple as possible and that there is no unnecessary repeated code.

7. **Miscellaneous:**

   1. [a2] In most cases no numbers other than 1 or 0 should appear in your program. Other numbers should usually be declared as constants using the "final" keyword.
   2. [a2]**Do not use any class variables in your main class!!** These are variables that are declared in the main class but outside of any method definition. The exception to this rule is the Scanner variable. Violating this guideline will cost you a lot of points!
   3. [a5] You must use a value-returning function if (a) there is exactly one value being communicated to the calling function, and (b) there is no input or output occurring in the function.
   4. [a1] The characters "== true" or "== false" should never occur in your code.
   5. [a5] You should never have

      ```
      if (x) {
          return true;
      } else {
          return false;
      }
      ```

      in your code. This can be replaced with simply

      ```
      return x;
      ```