CS 85A (Sections 3, 4): Introduction to Programming Workshop, Spring 2023

David Scot Taylor
Associate Professor
Dept. of Computer Science
San Jose State University
212 MacQuarrie Hall
Phone: (408) 924-5124 (email works better)
Email: david.taylor "at" sjsu.edu

Spring 2023 office hours:

- Monday, 1:30-2:30 (in person).
- Tuesday, 12:15-1:15 (in person).
- Other times available, especially on Tuesdays. Set up an appointment by email.

Office hours will generally be in-person. When they are virtual, use this zoom link for zoom office hours, and you will enter the waiting room (I will see one at a time in the order you enter, occasionally updating the waiting room through chat about the size of the queue.)

Note: for this class, rather than using my general office hours above, you should use special office hours listed below.

Class Meetings:

- Section 3: Fri 8:00-8:50am, MacQuarrie Hall 225.
  Office Hours: Fri 8:50-10:30am, MacQuarrie Hall 225.
- Section 4: Fri 11:00-11:50am, MacQuarrie Hall 225.
  Office Hours: Fri 11:50-1:30am, MacQuarrie Hall 225.

COVID-19 and Monkeypox Safety Training

Students registered for a College of Science (CoS) class with an in-person component should view the CoS COVID-19 and Monkeypox Training slides for updated CoS, SJSU, county, state and federal information and guidelines, and more information can be found on the SJSU Health Advisories website<\a>. By working together to follow these safety practices, we can keep our college safer. Failure to follow safety practice(s) outlined in the training, the SJSU Health Advisories website, or instructions from instructors, TAs or CoS Safety Staff may result in dismissal from CoS buildings, facilities or field sites. Updates will be implemented as changes occur (and posted to the same links).

Prerequisite Courses

You must show me that you are currently enrolled in, or have already taken, CS 46A.

Course Format

The class will generally consist of 50 minutes of a lecture of some sort, followed by 140 minutes of exercises during "office hours". You should be able to complete all homework if you stay for office hours and participate

in the exercises.

Course Website

The main course website will through SJSU's Canvas website at https://sjsu.instructure.com/. Basic course information, including a link to this greensheet, is at http://www.cs.sjsu.edu/faculty/~taylor/term/current/CS85A/.

Course Description

Designed to help all students excel in Introduction to Programming. Students work in groups on challenging problems to help them understand programming concepts more deeply and lay the groundwork for success in future courses.

Faculty Web Page and MUSJSU Messaging

Course materials such as syllabus, handouts, notes, assignment instructions, etc. can be found on CanvasCanvas Learning Management System course login website. You are responsible for regularly checking with the messaging system through MySJSU on Spartan App Portal to learn of any updates.

Course Learning Outcomes

To assist students in material from CS 46A and promoste teamwork within the workshop. See CLOs for CS 46A.

Required Texts/Readings

Worksheets may be provided to go along the textbook from the parent course CS46A. (Anything needed will be in Canvas.)

Other Technology Requirements

This course will meet in person. All students are required to have access to a wireless laptop (running OSX, Windows, or some version of UNIX), with a camera and microphone, upon which you can install required software. You will need it for all classes, labs, and exams. Technology used will include Canvas, programming in Java, an IDE (Integrated Development Environment) and program submission through a website called Web-CAT.

Library Liaison

Anamika Megwalu, email: anamika.megwalu@sjsu.edu, website: https://libguides.sjsu.edu.

Course Requirements and Assignments

See general SJSU policies at the University's Syllabus Information web page.

The expectation is that you can complete all "homework" for this class in the 100 minutes immediately following your section. There will not be any exams.

Students are required to participate actively throughout the semester, and participation will be assessed at every class meeting. Insufficient participation will result in either being dropped or receiving a grade of NC (no credit). This is a one-unit Lab that meets 3 hours per week.

SJSU classes are designed such that in order to be successful, it is expected that students will spend a minimum of three hours per unit per week, including preparing for class, participating in course activities, completing assignments, and so on. More details about student workload can be found in the University's Syllabus Information web page.

Grading Information

This workshop is graded Credit/No Credit. The grading policy for the workshop will be based on classwork given during each workshop meeting, as well as performance in CS 46AX. To receive a "CR" grade for this class, you must earn at least 70% of the participation points.

The classwork is based on topics that you are currently studying, have studied, or will study shortly. In order to receive credit for the classwork you must be actively participating in the coursework. You may use your books and notes and work with others in a group to complete assignments. A student who does not participate adequately in more than 5 classes will receive a grade of NC.

Classroom Protocol

Please put your phone away during class.

Some Tips for Succeeding in CS 46A

Why CS 46AW (CS 85A)? In order to learn your first programming language, you need to overcome several barriers:

- You need to understand the organization and logical steps required to accomplish whatever task you are supposed to solve with your program. You have learned many "algorithms" over your life: something like the process to multiply two long numbers against each other is an algorithm. But, in programming, sometimes we are expected to come up with an algorithm to solve a problem, rather than just following an algorithm. That is, rather than being asked to follow a given algorithm to solve an instance of a problem, you are asked to come up with an algorithm that can solve all instances of some problem. Consider: if you didn't know the algorithm for long multiplication, inventing that method would be much more difficult than just following it.
- You need to learn the syntax of the programming language (Java here), in order to be able to tell the computer how to follow that logic. Like any language, small changes in a program can greatly change what it accomplishes, but it tends to be more extreme: in a Java program, one misplaced character might change a perfectly working program into one that is faulty, or one that the computer can't even recognize as a program.
- Because these first two tasks are so difficult, many tools have been developed to help you to program. To help with the fact that the syntax can be so finicky , your Integrated Development Environment (IDE) can highlight the parts of your code that are not legal Java syntax. It can even make suggestions to you of how to fix your code. And, once the program is in running order, they can help you to find errors in your logic. These tools have grown in usefulness and sophistication over the years, and will help to make your time coding much more fruitful…eventually. Unfortunately, like many tools, it takes some familiarity with them before they start to really pay off. And, while you are already struggling with all of the new material from the two issues above, adding new tools to the mix can just make programming seem overwhelming at first. <\il>

Because of this, the start-up learning curve for programming can be quite steep. Consider this: some students program their first line of code in college, and less than 4 years later, they are able to get jobs as professional programmers. That is quite a quick ramp-up from novice to professional. I suspect there aren't too many music majors who take up their first instrument in college, expecting to be professional musicians 4 years later.

So, why do students get discouraged? Because, at first, it is difficult. They don't get enough practice. They think that they can't do it. And, sometimes, they compare themselves to other students in the class, and feel like they don't measure up. Oftentimes, that is a very unfair measurement: many students in CS46A have programming experience. Even if it was in a different language, frequently, the issues from the first point above are language independent. So, even if they programmed in a different language, they have already seen much of the logic, and now only have to learn the syntax. And tools…except they have likely already seen some similar tools too. And, some of them have done this in Java: maybe they were introduced to Java in the CS Principles AP, which doesn't give credit for CS46A. Maybe they took a CS A AP course, which would give credit for CS46A, but they were unable to take the exam, or didn't score well enough on the exam to get credit. In any of those cases, it is not reasonable to compare your knowledge, six weeks into your first programming class, with the knowledge of students who have seen it before, last year.

We are developing CS46AW (CS 85A this semester)specifically to give students with less programming experience extra opportunities to practice computational thinking, and extra chances to do lots of simple programs. Like most things in life, we get better at programming by practicing. For CS46AX, much of that practicing will be at home, on your own, but in CS46AW, you will get the chance to work on more problems, with peers and supervision, to help keep you from getting stuck. Students with programming experience are certainly allowed to take CS46AX/AW instead of CS46A, but our expectation is that this extra practice will be most beneficial to students who haven't programmed before.

Effective Study Habits

- Read the assigned sections before class so that you will have some idea of the day's topic.
- Work through the examples in the sections covered in class.
- Make use of additional references and resources.
- Seeing somebody else code can help to teach you something about coding, but it will not make you a good coder. Only writing your own code can do that. Similarly, watching somebody else play piano/tennis might teach you something, but it will not make you a good piano/tennis player. Students assume that, if they can follow someone else's code, that they understand the topic, but if you don't practice writing it yourself, it isn't the same.
- Get all the help you can from your instructor or the tutoring service.

Major Causes for Failure

- Lack of motivation, unwilling to commit the time and personal effort necessary to master course details.
- Students do not spend sufficient time on the material outside class, or they don't try to get help from instructors and tutorial services.
- Students get answers from others, confusing "getting the answer" with "understanding the material".
- Getting behind, letting things pile up, and trying to catch up after it is too late. Do homework and study at a steady pace.
- Early difficulties lead to students getting discouraged, thinking that they are "naturally bad" at programming. A little secret: at first, almost all of us were bad at programming.
- Not reading the book and relying completely on the lecture or relying completely on the book and not taking notes.
- Not going over tests and homework and correcting mistakes.

University Policies

Per University Policy S16-9, university-wide policy information relevant to all courses, such as academic integrity, accommodations, etc. will be available on Office of Graduate and Undergraduate Programs' Syllabus Information web page at http://www.sjsu.edu/gup/syllabusinfo/.

Drop and Add Dates

Note that for this semester, the last day to drop without consequence is ***Monday, February 20***. That is also the last day to add. After that date, it becomes very difficult to drop or add a class, so be sure you are where you want to be before these dates arrive! If you are going to drop the class, figure that out ahead of time, so that there is time for you (and others) to shuffle courses as needed.

Class Format

This class is scheduled to be an in-person class.

Recording Lectures or Sharing Course Materials

You can make audio recordings of class for your own personal use. Perhaps you want to want to have my dulcet tones lull you to sleep at night instead of only during class, that is fine. Weird, but fine. Perhaps you want to torture your neighbors by blasting it on your porch, that is not fine: aside from possible violations of the Geneva Convention, recordings should not be reproduced, distributed, or publicly broadcasted. If you want to make video recordings, please discuss it with me.

Course material developed by the instructor is the intellectual property of the instructor and cannot be shared publicly without his/her approval. You may not publicly share or upload instructor generated material for this course such as exam questions, lecture notes, or homework solutions without instructor consent.

Tentative Class Schedule

A precise schedule will be online in the school's Canvas system. Below is the planned outline, which will be modified as we go.

Date **Subject to change**	Planned Topic	Actual Topic (if different) or more detail
January 27	Introductions, Specifying Algorithms for Every Day Tasks	
February 3	Method Calls and Overloading	
February 10	Scope and Object Oriented Programming (OOP)	
February 17	Debugger	
February 24	Instance Variables and Object Persistence	
March 3	Strings and Nested Methods	
March 10	More Strings and Nested Methods	
March 17	Algorithms and Flow Charts	
March 24	Loops	
April 7	ArrayLists	
April 14	More ArrayLists	
April 21	More ArrayLists	
April 26	Designing your own Classes. Reading from Input Files	
May 5	Inheritance	

May 12	Strings?

There is no final exam for CS85A. There might be a "final project" that would just be a somewhat more complex program, that we can still work on in class.