SYLLABUS

COMP 110/110L Intro to Algorithms and Programming Spring 2020

Instructor: Dr. Jeff Drobman
Dr Jeff's Web Page: http://www.drjeffsoftware.com
Course Web Page: http://www.drjeffsoftware.com/classroom.html
Email: jeffrey.drobman@csun.edu
Office Hours: In Lab, immediately after class

Course Description

Introduction to ***algorithms***: their representation, design, structuring, analysis and optimization – as recipes for solving a problem or performing a function.
Introduction to ***programming***: as structured designs exhibiting *encapsulation* expressed in the high-level application programming language ***Java***. Some comparisons are given to other languages such as C and C++. *Object-oriented programming* will be covered.

Assignments and methodology: We will be applying to a series of **Lab Programs** and **Projects** the "software engineering" paradigms of *Software Development Life Cycle* (SDLC) and *Input-Process-Output* (IPO).

Finally, to enhance 2-way feedback, we will utilize an interactive *e-book* from zyBooks, including labs ("zyLabs").

Goals

This course teaches basic programming skills in *analyzing problems*, and then creating *computer programs* that solve them by:

1. Writing a set of functional ***requirements*** (1st stage of SDLC, mostly given by the instructor)
2. Creating a program ***structure*** that maps to, and covers, the requirements (somewhat given by the instructor)
3. Finding or creating appropriate ***algorithms*** to implement key requirements (given by the instructor)
4. Incorporating each algorithm inside its own structure (*encapsulated* as a *method*).
5. Adding a full complement of Input and Output (**I/O**) on both the *Console* and *GUI* (*Swing*) plus data files
6. Employing *object-oriented programming* (OOP) by defining and *instantiating* multiple *Classes*
7. *Debugging* and *testing* of programs (4th stage of SDLC)

Note that steps 2-6 encompass the ***SDLC*** stages called "Design" (stage 2) and "Implementation" (stage 3).

This course does not assume any previous experience in computer programming, and material begins at an introductory level. However, coverage is fast-paced and moves on to more advanced topics quickly. This is not a survey course for non-majors; it is a real programming course designed for students concentrating in Computer Science and related majors who need to quickly develop real programming skills.

Java

The principles and skills related to problem solving via algorithms and structured programming are general and are not specific to any programming language. However, it is imperative to also learn the modern programming paradigm (model) of *object-oriented programming* (OOP) – which is at the core of the language ***Java***. Most desktop, mobile and other client-side computer applications today are written in either Java or C++ (or its derivatives).

The CSUN CS Department has made a policy decision to use the **Java** language for lower division courses such as COMP110, so Java language details will be presented as an important part of both lecture and lab (although the policy may change in the future). For now, you must demonstrate an appropriate level of problem solving, algorithm usage and Java programming skills to successfully complete the course. (Rest assured that currently most US colleges teach Java, while some teach C++ instead.)

The lecture component (COMP110) focuses on fundamental concepts and practices with examples. The lab (COMP110L) focuses on developing working, correct and robust application programs using Java, with attendant *problem solving* along with program *debugging, testing* and *error handling* techniques.

Textbook

There is one *recommended* textbook (required in the past): Y Daniel Liang, Introduction to Java Programming, *Brief Version*, 10th, 11th or later edition, Prentice Hall. The instructor will make references to various pages and sections in this book from time to time.

There is a *required* interactive e-textbook: zyBook with zyLabs (a single book with integrated labs). Students are required to buy access (a subscription). The cost of this e-book is about the same or less than the cost of the Liang hard copy book (about $70). This e-book is *interactive*, and will require students to keep up with the presented material. Student progress will be monitored by the instructor to be in concert with lecture slides. There will be additional graded work to be performed in the zyBook (zyLabs), as embedded by the instructor.

The baseline zyBook may be in flux, with occasional instructor added material – which will appear in each student's copy.

zyBook "Activities"

zyBooks have 3 types of *Activities* – exercises that must be completed, and will be monitored for progress.

1. ***Participation*** Activities (PA) – none of these will be graded, but will be monitored

2. ***Challenge*** Activities (CA) – Five (5) chapters of the zyBook will have their CA's graded and will account for 10% of student grades (2% each).
3. ***Lab*** Activities – The first 6 Labs will be implemented in part in the zyBook (the "zyLab") – meant as "starter code" to guide the student to completing the Lab in the class *IDE* (jGRASP).

Exams
Exams, quizzes, programming assignments ("labs") and term projects are based on what is presented in the lecture. There will be one Quiz (at first quarter), a Midterm and a Final Exam. All these exams will have one part that is multiple choice entered on Scantron forms. The Midterm and Final Exams will have an additional section for writing *code segments* to demonstrate capability.

Lecture
All lectures will be delivered via instructor PowerPoint slides, coupled with zyBook material and live coding examples. Slide sets will be periodically updated and posted in PDF format to the class web page. zyBook material will also be periodically updated.

It is recommended that for the best level of preparation, students should attend every lecture and lab and participate in discussions, rather than simply reading the lecture or lab material on your own. The use of the zyBook will confer extra means of learning by *interaction* – all monitored, some graded. There will also be a limited amount of hand-drawn material on the board, for which each student is responsible for taking notes.

Lab: Assignments
All Labs and Projects are graded programming assignments. There will be 8 consecutively scheduled "Labs" plus two term projects – one for each half term. The Labs are carefully selected to integrate the current lecture material and lead the student to a mastery of traditional and fundamental programming techniques, areas and algorithms. There will also be some non-graded *exercises* intended to enhance learning (both in the Lecture and Lab sessions).

Most Labs will be issued, performed and auto graded in appropriate chapters inside the zyBook, thereby guiding students through the recommended program structure. Each zyBook Lab section will be accompanied by *starter* or *skeleton* code segments provided by the instructor. Students will need to fully understand the provided code, and then modify it and/or add their own code for completion of each section. Next, each Lab will be composed from the zyBook code segments into a complete program inside the Lab IDE (see "Lab IDE" below) and submitted for final grading on the provided *Lab Form*. Students will be graded on their submitted Lab Form. The zyBook work product is intended to indicate whether or not the student understands the code.

The two "Projects" are each designed to fully incorporate all the material learned up to the mid-point (#1) and over the entire course (#2). They are opportunities for the student to demonstrate their mastery of this course. They are intended to be realistic and fun:
(1) thermostat simulation, and (2) casino card games (poker, etc.).

Submissions of Labs and Projects will use a provided form for documentation to include each student's Requirements, Input, Output and Source Code listing. All Labs and Projects are to be completed on their dedicated forms and submitted as Word files attached to emails sent to the instructor directly (using my csun.edu address above).

Lab: Programming – Integrated Development Environment (IDE)
Programming in Java is practiced in each Lab session, putting to use concepts and constructs learned in the preceding lectures. We will use the Java *Software Development Kit* (SDK) called ***JDK***, from Oracle, for the Java *compiler*. We will use the zyBook IDE (when using the zyBook) and the CSUN provided IDE called ***jGRASP*** (freeware from Auburn University). jGRASP will be pre-installed on all classroom student workstations running Linux (from Red Hat). Both JDK and jGRASP may be optionally installed on student owned PCs or Macs. Students may thus use any combination of classroom workstations and their own machines.

Learning to use the jGRASP IDE is an important part of the lab and the course. All final source code editing, compiling, running and debugging will be done with this IDE. You will be shown how to utilize some of the extra features of it, such as generating *CSD* (Code Structure Diagram) and *UML* (Universal Modeling Language) class diagrams, along with running programs in *debug mode*.

The Lab sessions will mostly involve the instructor going over the assignments, along with the "starter code" for each. Some related general code segments, and some other example code, may also be presented during the labs. During the Lab sessions, students are encouraged to work on their assignments, and may ask for help from the instructor or the in-class tutors, or any other students.

It will usually be necessary to continue to work on your programming assignments on your own time outside of your official lab time. You will also need to work on your Projects in parallel with Lab programs as part of your lab work.

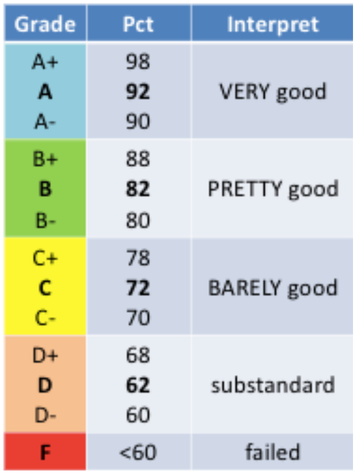
Collaboration
Collaboration among students is encouraged. However, this must fall short of copying any code segments, which is plagiarism. (That said, it is often the case that there is only one good solution to each part of an assignment, and this instructor will take that into account.)

Late Submission Policy

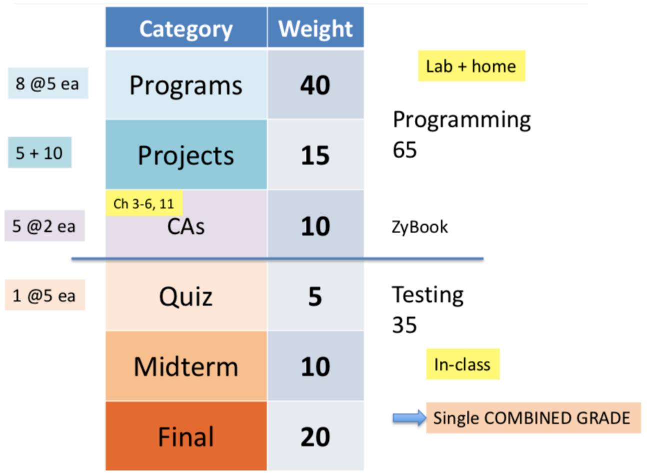
All assignments will have a due date. You will have a one-week grace period. After that, late assignments become "past due" and are assessed a "late penalty" of 10% per week – for 2 more weeks. No assignments will be accepted after that final past due period. Finally, all remaining assignments (if any) must be submitted by the last day of class (regular classes, not the final exam week). No assignments will be accepted after the last day of class.

Tutors
The School (CECS) provides designated in-class tutors available during each Lab session. Students will be expected to make use of this resource. Additionally, the School provides a bullpen of tutors available M-F in the designated tutoring room in JD1622. The outside tutor pool is to be used only for Q&A type help; they are not allowed to write your code for you.

Grading
You will receive a **single combined grade** for both 110 and 110L. **Plus/Minus** grading will be used, as shown (except no A+).



Grade	Pct	Interpret
A+	98	VERY good
A	**92**	
A-	90	
B+	88	PRETTY good
B	**82**	
B-	80	
C+	78	BARELY good
C	**72**	
C-	70	
D+	68	substandard
D	**62**	
D-	60	
F	<60	failed



	Category	Weight	
8 @5 ea	Programs	40	Lab + home Programming 65
5 + 10	Projects	15	
5 @2 ea	Ch 3-6, 11 CAs	10	ZyBook
1 @5 ea	Quiz	5	Testing 35
	Midterm	10	In-class
	Final	20	Single COMBINED GRADE

Computer Accounts
Every student registered for the course has a networked account that can be used on all CS Department computers. Your account (user id and password) is the same as the one issued by the University. The instructor does not have *sysadmin* authority over student accounts. For problems logging in, see the CECS Information Services office in Room JD 1112, extension 3919.

Software Tools
You may do your work on any Lab machine or on your personal computer (PC or Mac). Lab machines use the Linux OS, but the software applications required for class work are available for all common OS's. The Java JDK and jGRASP come pre-installed on the Lab machines. For your own computers, you will have to install these 2 tools. The tutors can help you with this.

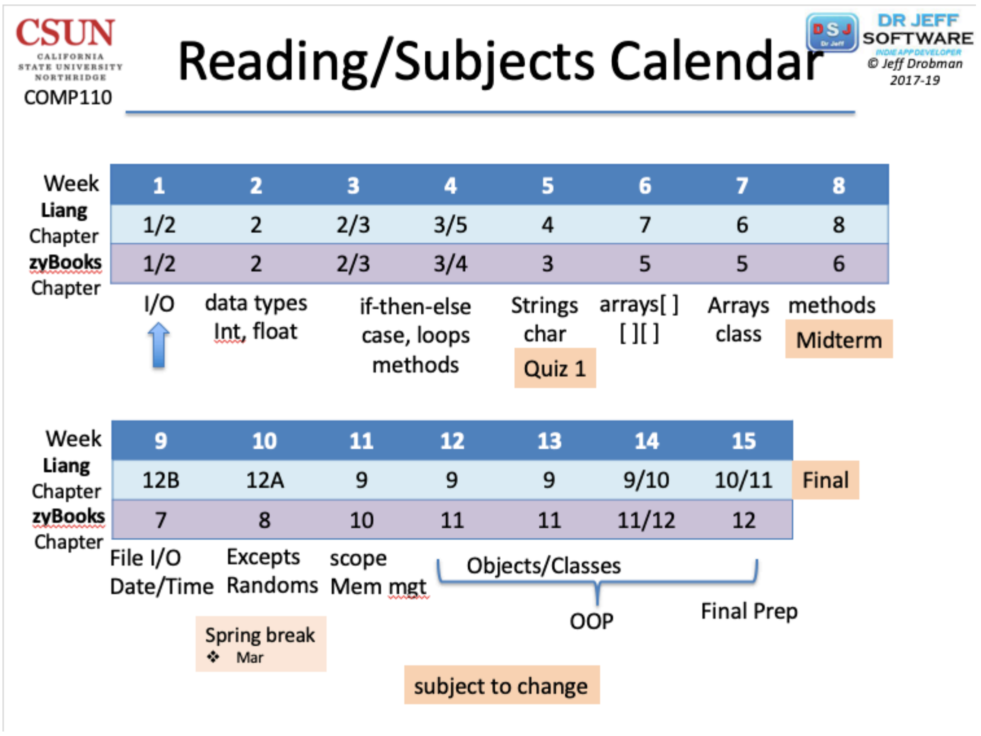
When installing on your own computers, it is important that you install the JDK first, then the jGRASP IDE. Make sure you install the Java "JDK", not the "JRE" (Java Runtime Environment), and use the "SE-8" version (not the "EE" versions).

For the **JDK**, go to:
    http://www.oracle.com/technetwork/java/javase/downloads/index.html
    Download the Java 8 SE (Standard Edition) JDK – latest version is "Java SE 8u<nnn>", where <nnn> is at least 231 now.
    Install JDK first, before proceeding to download and install JGrasp.
For **jGRASP**, go to:
    http://www.jgrasp.org
    Click on the Downloads link and download the installation for your platform (Windows, MacOS, or Linux/Unix).

Exam Materials
We will use Scantron forms (long forms 882-E, not the short ones) for all exams, so bring one plus a No. 2 pencil to the exams.

Save Your Work
Work done in the lab on classroom workstations can be saved to your personal directory on the CECS file server (the "Z drive"). Make sure to backup all your files onto your own thumb drives, since the network machines can be reformatted at any time without prior notification (and you may lose your files). Students can also backup files by sending an email to themselves with files as attachments.

Schedule of Topics (subject to change)



CSUN CALIFORNIA STATE UNIVERSITY NORTHRIDGE COMP110

Reading/Subjects Calendar

DR JEFF SOFTWARE INDIE APP DEVELOPER © Jeff Drobman 2017-19

Week	1	2	3	4	5	6	7	8
Liang Chapter	1/2	2	2/3	3/5	4	7	6	8
zyBooks Chapter	1/2	2	2/3	3/4	3	5	5	6
	I/O	data types Int, float	if-then-else case, loops methods		Strings char Quiz 1	arrays[ ] [ ][ ]	Arrays class	methods Midterm

Week	9	10	11	12	13	14	15	
Liang Chapter	12B	12A	9	9	9	9/10	10/11	Final
zyBooks Chapter	7	8	10	11	11	11/12	12	
	File I/O Date/Time	Excepts Randoms	scope Mem mgt	Objects/Classes		OOP	Final Prep	

Spring break
❖ Mar

subject to change

Attendance

For at least the first several weeks, attendance will be closely monitored in both Lecture and Lab. After the first few weeks of the semester, attendance will not be monitored as closely, but note that the scheduled Lecture and Lab time is the most appropriate time to ask questions and get help on your projects. I am not willing to spend an unreasonable amount of time outside of class (including excessive use of email) to explain class material to anyone who is not attending class. In other words, you are expected to participate in Lecture discussions by attending Lecture, and get programming help on your projects in person by attending Lab. Students are welcome to bring personal laptops with a wireless connection to both Lecture and Lab sessions.

Personal Behavior in Lecture and Lab

You are expected to arrive to both Lecture and Lab sessions on time, and to wait until dismissed before leaving. You are expected to be polite, pay attention, and participate in discussions: no web surfing, no private conversations, no cell phone interruptions, no sleeping, snoring, etc. You may quietly excuse yourself for bathroom breaks, to take important phone calls, and to address other urgent personal business, without asking for permission, but keep interruptions to a minimum. During Lab, you are expected to keep personal web surfing to a minimum (quickly check your email then get to work). You may not play music or otherwise disturb other students while in the lab. If you finish your lab work early, you may remain in the lab and work quietly. You may be asked to leave after you finish your work if your personal computer usage is disruptive to other students. If you're ahead of the rest of the class and finish your work early, kindly consider looking around and finding someone who could use your help. You may not invite friends who are not enrolled in the course to join you in the lab to "just hang out".

Plagiarism and Academic Honesty

Academic honesty is expected from all students taking the course. All work that a student submits for course credit must be performed by the student who submits it. Use of for-pay programming services such as RentACoder for programming projects for course credit, either as a buyer or a seller, is not permitted, and is considered a serious offense. Exams must be taken according to parameters established by the instructor at the start of the exam. In general terms, students are prohibited from all communication, both verbal and electronic, with anyone except the instructor during an exam. Notes and other reference aids may or may not be permitted, instructor will specify what resources are allowed, if any, when exam is announced.

For programming projects, you may discuss coding details with other students, but you cannot copy and paste large chunks of code from one student's work and then submit it as your own. Violations of this policy can result in a failing grade for the project or exam in question, and depending on the severity of the violation, may result in a failing grade for the course, as well as the violation being reported to the Dean of Academic Affairs. Students who repeatedly violate the policy across multiple courses may be suspended and even expelled from the University.