# How to Take This Course

## Overview

The work for this course will follow the chapters and exercises of the Comprehensive Versions of Y. Daniel Liang (2020) Introduction to Java Programming and Data Structures Comprehensive Version, **12th** Edition.  Assignments for the 11th and 12th Edition are also available. No work from earlier editions will be accepted if it differs from these two editions.

The approach of this text is rather traditional in the sense that it provides information on aspects of Java, illustrates them with examples, and asks the student to do exercises from the text.

However, this site is now paired with a [Pearson Revel site](https://www.pearsonhighered.com/revel/index.html) that contains an electronic copy of the text (once we get it working) enhanced with many features to help you Learn. These include pop ups for new words, embedded videos and interactive exercises, etc. I have only used it once before so I will be only a little ahead of you as we will be learn how to use it together. You are still responsible for the assignments regardless of how you read the book. Everything I tell you in this file may be subject to change one we become more acquainted with Revel.

It may also be far cheaper to access than the hard or soft cover version of the 12th edition. Check with the book store. I have not yet heard how you get to access it and how you opt in, or out, for this feature.

Read the documents in Supplements I of the text’s [companion web site](http://www.cs.armstrong.edu/liang/intro11e/toc.html), With the switch to Revel, it sl may still be the companion web site, but I have no information about this (the inside front cover of the text tells how to access it) to learn how to install the Java Development Kit (JDK) and the Java Runtime Environment (JRE). The instructions in the text and the companion site may appear to be out of date. More recent versions of the JDK and JRE are available from

<https://www.java.com/download/.>

Neither chapter tests nor unassigned exercises are required but might be useful for studying. The only requirements of the course are the submission of the seven assignments and the final project.

If not already done, students will need to upgrade their JDK and Runtime JRE to Java version 8 or later. These editions of the text use the latest graphical Java methods in a package known as JavaFX. The previous graphical methods, AWT and Swing, are no longer included in these editions although it is possible that they are still available and possibly part of JavaFX. You may use them in responding to the exercises but they are not the Java future. Check into the Supplements II.F-H to run JavaFX from the Windows command prompt, similar feature for Macs, and/or NetBeans and Eclipse, although Eclipse still has problems with JavaFX. Visual Studio also has features that allow to use it for Java software development. Other Integrated Development Environments may have difficulty with JavaFX but I think we can find tools to make it work. It is not required before Assignment 7.

I will be using Eclipse or the MS Windows 10 command prompt window to check your submissions. All submissions must be able to be executed in at least one of these environments. If your submission needs special actions on my part, like using a .jar file or another program like MySQL, you must provide me with **idiot proof** instructions of how to do so.

**Be sure to also read all the files in all the Modules, particularly the files in Modules 1 and 3 about user-dialog rules, Module 4 about structuring Java programs,** and **exception handling in Chapter 12.** Not following them may lead to credit loss, particularly for Module 4. For this course it is required that you follow the structures shown in the video and example files. It will be a good idea to also look at all the files attached to the assignment folders.

Students who take this course will possibly make it the only programming course in their studies or will follow it with further courses in either Management Information Systems (MIS) and/or Computer Science (CS). A set of assignments somewhat customized for these two paths have been established. If this is likely to be your only programming course, it is recommended that you pursue the one for MIS but which one you select is up to you.

## Background

Programming is the use of natural language terms, in a very specific and structured manner, to instruct computers to do logic, calculations, input, output, and other symbol manipulations for some purpose. This aspect of programming is described in the first part of Chapter 1 of the text. There are many different programming languages, and each conforms to the program structures of that language.  Program structures are similar across languages to some extent, but there are significant differences in the structure of programs written with an object-oriented (OO) language like Java from older, traditionally structured languages.

The basic work of OO programming in general, and Java programming in particular, is the creation, use, and/or extension of existing program components, called **classes**.  One of the advantages of this approach is that once a class has been created it can be used and re-used in other applications either as written or modified or extended.  Over the years many such classes have been developed and are distributed as class libraries.  Currently thousands of such classes exist and new ones appear with every new version of the Java Development Kit (JDK).  A handy but voluminous source is at <http://docs.oracle.com/javase/8/docs/api/>. There is a corresponding site for Java 7. Many programmers also make classes and entire packages of classes available to everyone as either stand-alone packages or extensions to existing ones. Much public domain software is available in Java, particularly from groups like the [Apache foundation](http://www.apache.org/).

A new layer of Java organization, modules, have recently been introduced. This edition of the text has no material on this layer. It is suggested, and mentioned in some assignments, that you do some research to determine how this modules layer affects the code of the assignment and report what you find. These reports do not have to be extensive; URLs and/or attached files may be sufficient.

## Text

The text is organized into five parts (see page vi-vii of the Preface) with 44 chapters and letters A-I appendices, hence the Comprehensive Edition designation. This allows the text to be used as documentation if your future studies and work includes Java. The first 30 chapters and the appendices are printed in the text while Chapters 31-44 are available from Revel or the [Companion Web site](https://www.pearson.com/us/higher-education/product/Liang-Companion-Website-for-Introduction-to-Java-Programming-and-Data-Structures-Comprehensive-Version-12th-Edition/9780136520085.html).

The first 20 or so chapters are also available in a briefer version, but the comprehensive version is recommended since it provides references for advanced Java programming.  Since this is an introductory course, assignments are from the first 14 chapters only. The final projects can be based on additional chapters and/or your personal interests. In the Preface, page vi, there is a tree that shows which chapters “naturally” follow from the first two parts; use these to check that you have or can acquire the background needed for work from the advanced chapters.

Each Assignment requires responses to exercises in the 11th and 12th edition, although using these may no longer work. Earlier editions are **no longer authorized** for this course. You may also find a version of the text called something like the International Edition; this is also **not authorized** as a text for this course. The topics may be similar, but the examples and problems are almost all different. If you insist on using these unauthorized editions you will have to try to match the problems to the ones assigned and you may have to copy the problems to which you are submitting. Your instructor may not be able to help you with this.

Assignments have exercises designated as being of an MIS flavor and others designated with a CS flavor; you may choose either set for each assignment or mix and match. For most of these assignments you will be asked to either modify an existing Java program or write a new one.  After you have done this, submit the modified and new programs to your instructor, for grade based on compile and run, as an attachment to the corresponding Assignment page in Canvas.

**Note well:** Attach and send only .java files (really text files with a .java extension) and any files needed for input; do not send .class files or any directories. Also, do not send files with working package designations, my directory tree is unlikely to be the same as yours.

## Submitting Responses to Exercises and Projects

Learning how to program is an individual activity. You can study in groups and share insights and practices but eventually you, as an individual, must learn how to write a program from scratch. You can copy code from existing programs but it may not fit the purposes of the program you are writing exactly so, again, you have to get the code to do the purpose of the program on which you are working

For the final project you may work in groups of up to three students. For this assignment only you will have to sign up for a group in the Project Groups tab from the People menu item of the Canvas site. Only one member of each group needs to submit assignments which must include the Group ID/Number and the names of all the students in the group. All students in the group will get the same score. It is each group’s responsibility to ensure all members benefit from each assignment. It is likely best if all members of a group wish to pursue the CS or the MIS track.

Submit all programs and files for an Assignment in a **single .zip or .rar file with no sub-directories**; submitting individual files makes reviewing them inefficient and messes up Canvas’ displays and possibly its score keeping.  **Do not send individual .java files in their own folder**. The only thing your instructor should see in the .zip or .rar file is text files with a .java extension and input, text, and/or image files for input. Do not put .txt and image files in their own directory. All files should be in a single directory, it makes locating them by the programs easier than if paths must be specified to find them. I will not create or modify my ClassPath or any of my environment variables to run your submissions. In realistic development situations other rules may apply.

**Do not include packages statements in your code**.  Some IDEs automatically structure the project into packages (usually folders or directories).  Your instructor’s folder organization (directory tree) may not be the same as yours and package statements needlessly make your programs harder to test.  **If your IDE uses package statement remove or comment them out before bundling the files into the .zip or .rar submission file.**

If you have problems with any exercise, you are welcome to submit the incomplete program with questions and I will return it with comments and suggestions.  This cycle can be repeated until you are satisfied with the program and submit it for credit.  Do not spend a long time on any one exercise (i.e., don't get bogged down); ask questions and submit partially completed programs sooner rather than later.

Whether complete or not, the .zip or .rar file should be submitted as attachments to Assignment pages. These pages need to have a score entered which might be quite low for programs with problems, but you will have chances to resubmit for a higher score. Take the instructor’s comments into consideration when you resubmit. Keeping the conversation within the Assignment pages allows all the dialog associated with an assignment to stay in one place making back references easy to find.

As mentioned, you should only submit the .java files, **do not send the .class files.** You may also have to attach test files and images.  See the start of the Assignment pages, and the Assignments themselves, for more instructions on what to submit and how to submit it. You are welcome to provide any comments and/or questions in the Assignment page itself in the text area.

**Please** **submit the Assignments as you finish them**, do not wait to accumulate several assignments before submitting them. In this way mistakes and poor programming practices can be caught before they become a habit. Also, if you resubmit assignments based on my comments, send only the files you want me to subsequently review; **remove all other files** from the Assignment attachments. It is easy to become confused as to which version of a file you want me to check.

Assignment due dates are designated in the Assignment pages. Note that assignments, if not accepted, can be resubmitted but **note the 3-week resubmission window in the Syllabus: assignments not resubmitted within three weeks of the due date will receive their latest score.**

The first eight chapters concern themselves with two aspects of programming:

1. the basic syntax and semantics of Java (covered in 3-5 weeks of a standard programming course). There is a chapter on multidimensional arrays; there are no assignments for this chapter, but these arrays often form the basis for interesting projects, games and puzzles. This chapter can be included in a final project.
2. The use of program components, called classes, attributes, and methods in OO languages, is formally introduced in Chapters 9-13, although they are used and partially described in earlier chapters.  It is the use of classes and hierarchies of classes that many consider to be an important part of what makes OO languages object oriented.

Students who are using this course as an introduction to programming should set themselves the goal of completing the **first five assignments as the minimum requirement for this course** (this will earn a C grade – see below).

For all applications one of the main tasks is to interpret and parse user input often as text (even numbers when entered are first read as text although this may not be obvious). Text, after being entered, usually becomes a String in a programming variable; Strings are considered in multiple chapters. Strings also result from text typed into boxes or areas that are part of Graphical User Interfaces (GUI) for which the 11th and 12th Edition now contain material for the 3rd generation of Java GUI resources called JavaFX, further described in Chapter 14. The material in this and later chapters can be included in your project.

The course then ends with your choice of one of several projects, or a project that you suggest.  If you choose to do a project of your own design, let your instructor know as soon as possible so that appropriate boundaries can be put on it to make it possible to complete the project within the time of the semester.

Other important programming concepts, such as event handling, exception handling, arrays, collections, inheritance, string processing, sequential files, database programming, and web applications are considered in the remaining chapters.  Some Assignments and Projects are suggested from these chapters.

## User Dialog

Many programs require users to enter data and, after the program does its processing, will provide output. One of the common practices is to prompt the user with a message indicating what input is expected, such as an integer or the name of a student. Assignment 1 and 2 include instructions for prompts.

Be sure to read the files in Assignments 1 and 2 for user dialog rules in this course. These are specific to an introductory programming course like CS501 and will make testing the programs somewhat easier than if you do not follow them. This is a requirement of this course. For production applications clear and precise input instructions are necessary for successful programs.

One of programming skills you will acquire primarily through trial and error is how to structure the user-computer dialog. Regardless of how intelligent messages and instructions generated by a computer appear to be, whatever intelligence they exhibit comes from the program designer and programmer. Since these people will likely not be available when a user interacts with the program, the user-computer dialog needs to be carefully structured and instructions for input must be

* self-contained,
* easy to read and understand,
* accurate,
* complete, and
* very specific.

One of the basic “rules” of dialog is that all **input to a program must be validated and echoed back to the user**, preferably in the same window with the output. This is not always practical for large amounts of input but for all except maybe one or two exercises in this course this rule should be followed. If the output does not make sense to the user, it is frequently because of input errors. **Being able to see the input right along with the output and error messages in a single window helps users to diagnose the problem and correct their input**.

In addition, all input, even if only implicitly, has rules for validity (triangle sides cannot be negative, passwords often must be greater than a certain length, contain some specified number of letters, numbers, and punctuation marks, *etc*.). These rules should be communicated to the user explicitly, best as part of the prompt, example, and/or in associated documentation. For applications in the real world all are often supplied. Your program should check the user input for such validity and, when input fails to follow these rules this must be communicated to the user and allow the user to re-enter the input. Starting with Assignment 4 you are expected to use the **exception** constructs of Java. They should then be used for all subsequent assignments.

After the first assignment you should consider using popup windows for both input and output. The sooner you master these the more appealing your programs will be. Your instructor will provide suggestions as you go along.

The above “dialog rules” are also in a separate document for easy reference.

## Exceptions and Debugging

There are additional topics mentioned in the text:

* exception handling, and
* use of the debugger for finding/fixing program errors.

Exception handling is well structured in Java; there are 5 keywords that implement it. You will be asked to use them starting with Assignment 4. This is a useful tool to validate input, however it can be overused; it takes additional time and if not structured well it can be confusing to the user. Starting with Assignment 4 the user should not even be aware that the program uses exception handling unless something does not go right. All exceptions should be caught by the test program, communicated to the user, and suggestions for new input sought.

Debuggers are very useful when you start creating your own programs from scratch and you should learn to use it on the examples and the exercises before then.  While no exercises that deliberately call for the use of the debugger have been assigned, it will be worth your while to study its use and to try it. One effective way to do so is to deliberately introduce errors in a fully working program and then see how the debugger can help you find them.

## UML

UML stands for Universal Modeling Language.  It is a rich set of standards for application analysis, planning, design, and communication that is popular among some software design communities.  Only the class diagrams of the UML standards are used in the text, but the entire language is worthwhile learning.  A quick and easily digested source on UML is Rosenberg's *Use Case Driven Object Modeling with UML* from Addison-Wesley, ISBN 0-021-43289-7.  A fuller exposition can be found in Larman's *Applying UML and Patterns* by Prentice Hall, ISBN 0-13-092569-1. More recent editions and other books may also be available.

## Grades

The Syllabus contains the grading scheme, check it out. Your grade will depend on both how satisfactory the responses to assignments are and how many you complete satisfactorily. You may resubmit assignments that are not considered satisfactorily but the total score available on re-submitted assignments may be reduced when compared to the score available to the first submission. Check the details in the Syllabus.

## Finally

There are many more exercises than assignments.  You are welcome to do as many of the exercises as you wish (the more exercises the more you learn) but you should send your instructor only those extra exercises with which you have difficulty.

Programming can be a lot of fun, but it also can be frustrating.  Computers are very picky, and problems can be found to be a misplaced comma, or a misspelling buried deep in the code (debuggers can help with these types of errors).  Error messages are often less than helpful, although with the use of debuggers they can usually pinpoint the problem to a small part of the code.  Do not get bogged down in debugging.  After you have made a reasonable effort and still cannot find the problem, submit the non-working code to your instructor for assistance.  The availability of an instructor to assist you is the advantage of taking a course as opposed to learning programming entirely by yourself.  After the early chapters, topics become somewhat independent of each other, so you are welcome to go on to the other chapters while waiting for a response.

You are also welcomed to form on-line study groups with your classmates.  Assignment groups have been mentioned. These groups may collaborate, which helps learning. A good way to do this is to post a message in the Latest News or Mail facility of Canvas and see who responds.  If you are geographically near each other, you can even set up joint sessions.  However, all submissions must be from individuals except for the final projects and require that all members of such collaboration be mentioned in all submissions.

So, get into it and enjoy yourself!