Courant Institute of Mathematical Sciences
Computer Science Department
CSCI-UA.0102 Data Structures

Syllabus

Course Number

CSCI-UA.0102

Course Title

Data Structures

Course Description

The subject of this course is data structures and algorithms. A data structure is an arrangement of data in a computer's memory (or sometimes in a disk).

In this course *you will learn ways to structure and arrange data in computer's memory that includes array, linked lists, stacks, trees, hashtables, graphs, among others*. Algorithms manipulate that data in these structures in various ways, such as searching for a data item and sorting a set of data elements. In this class you will learn several concepts including how organize data elements, how to delete, insert, edit and search for a data element in a specific data structure and how to sort a set of data elements. You will also learn the differences between different data structures and when to use to use the right ones to solve problems.

An engineer is constantly solving problems. **You will be introduced in this class to practical problems to solve using data structures** such as utilizing data structures such as linked-lists and arrays to implement a course registration tool, storing and sorting courses and students' information data, modeling social network data using a graph data structure, and applying sorting and graph algorithms to analyze social network data (e.g. potential online community detection)

The programming language adopted in this class is Java, at the beginning of the semester **you will have a review of the object oriented programming paradigm** and some of its features such as inheritance, abstraction, encapsulation and polymorphism.

Number of Course Credits

3 Credits

Pre-requisites

CSCI.UA.001

Strong foundation of programming and the object oriented programming paradigm.

Course Website

https://cs.nyu.edu/~abari/CS102.htm

Instructor

Anasse Bari, Ph.D.
Email: abari@nyu.edu
Office: 425 WWH
Phone: 8-3227
Office Hours: (see https://cs.nyu.edu/~abari/CS102.htm)

Required Textbook

Data Structures and Algorithms in Java, 6th edition,

Michael T. Goodrich, Roberto Tamassia, Michael H. Goldwasser,

Sixth Edition

ISBN-13: 978-1-118-77133-4 (paperback), 978-1-118-80314-1 (e-text)

Wiley

Data Structures and Algorithms in Java
Book by Michael T. Goodrich and Roberto Tamassia

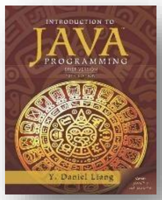
Course Learning Objectives

At a high level, at the end of this class students shall be able to:

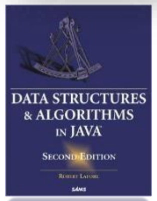
Computer Science Learning Outcomes	CS102 Course Learning Objectives
an ability to design a system, component, or process to meet desired needs	• Leverage data structures and algorithms learned in class to build a Java programs that will solve a practical problem in the real world. • Apply *the object oriented paradigm (OOP)* to break the problem into modules and utilize oop's features (inheritance, abstraction, encapsulation..) to solve the class programming work.
an ability to function on multi-disciplinary team	• Experience working in a team collaborating in some in-class assignments and some programming homework.
an ability to identify, formulate and solve engineering problems	• Use algorithms and data structures to model and solve real-world problems.
an ability to communicate effectively	• Acquire communication skills through conducting in class presentations of some specific course assignments and learning from the audience's feedback.

Recommended Textbooks:

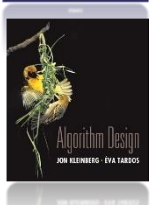
(CS101 Textbook)
Introduction to Java Programming, Brief Version, 10/E
By Y. Daniel Liang



Data Structures and Algorithms in Java (2nd Edition), 2002, by Robert Lafore (Author)



Algorithm Design Hardcover, 2005 by Jon Kleinberg (Author), Éva Tardos (Author)



Topics

More details will be communicated via email.

Motivation. Importance of understanding data structures and algorithms, the emerging field of data science and its intersection with data structures in computer science, the job market and acquiring a skill of computer programming.

Introduction. Overview of data structures, Overview of algorithms. Overview of Software Engineering lifecycle. Review of basic object oriented programming using Java.

Features of the Object Oriented Paradigm and Related Concepts. Abstraction. Encapsulation. Inheritance. Polymorphism. Interfaces including comparable interfaces. Abstract Classes.

Abstract data types (ADT). ADT for list, stack, queue. Array-based implementation of lists, stack and queue.

Algorithms: Prefix, Postfix, Infix notations and expression evaluation.

.

Recursion. Thinking recursively to solve a problem using algorithms. Practical recursive algorithms. Stacks data structure.

Searching & sorting. Linear search. Binary search.

Bubble sort. Selection sort. Insertion sort. Quick sort. Merge sort. Radix sort.

Introduction to Algorithms Analysis Tools. Big-O analysis: a methodology to predict the resources that the algorithm requires in terms of computer memory and computational time. Counting primitive operations. Algorithm growth rate.

Exception Handling in Java. Writing your own exception classes. Catching ALL possible exceptions.

Arrays as a data structure. The basics of Arrays in Java. Array's creation, initialization and elements' access. One dimensional, n-dimensional arrays. Array of objects. The ArrayList class in Java.

Linkedlist as a data structure. The concept of linked data representation. Organic implementation of linkedlist and its operations (add, delete, and search), implementation of Iterators.. Linkedlist provided in the Java standard library. Usage of iterators to traverse a linkedlist. Linkedlist implementation of Stack and Queue data structure.

Tree as a data structure. Trees' terminology. different types of trees, different types of traversals. Finding, inserting and searching for a node in a binary search tree. The efficiency of binary search tree. Heaps as an instance of a tree's data structure.

Graph as a data structure. Introduction to Graphs. Graphs' terminology. Programmatic Representation of Graphs. Modeling real-world data as graphs: Social Networks Data as a graph, Airport & flight connections as a graph. Graph traversals: depth first search and breath first search. Graph connectivity. Algorithms to find paths between two nodes. Bi-connected Components (application: potential communities in a social graph). Articulation Points (important individuals in a social graph). Topological sort.

Hash Table. Different types of conflict resolution and different choices for hash functions.

Optional (if time permits):

Balancing binary search trees and self-balancing binary search trees. Priority queues and heaps.

Homeworks and Programming Assignments

Full homework text will be communicated via email and NYU classes and the class website.

Course Policies

Office Hours Class Policy

You are very encouraged to visit the instructor during the office hours mentioned in the course website without notice. You will need to get ready for meeting your instructor by preparing focused and targeted questions. In case there is a line of several students waiting, the instructor will limit the meeting to a maximum of 10mins per student. In some cases, the instructor would schedule an appointment outside office hours. You are also encouraged to meet with tutors, graders, and posting questions in the Piazza online platform. You must also visit your tutors. See schedule in the class website here.

Attendance Policy

You are expected to attend every class. Attendance and punctuality are basic requirements for an effective computer science class. Attendance will be taken randomly at times. Attendance will be taken into consideration while calculating the final grade.

I am assuming that you are now a very active member of a learning community. Thus, you are expected to comport yourself as a professional individual as we are preparing you for real world job market. For instance, being on time for class, not leaving the class while it is in progress other than for emergencies, turning off cell phones and personal computers unless they are used for programming, being respectful of other classmates even if you disagree with them.

Make-up Exams and Quizzes Policy

Midterm and final exam dates are set as indicated in the class website (the date will be posted by the first week of classes). The exact date of the final exam will be communicated during the first two weeks of classes. The midterm will be announced three weeks in advance. There will be no individual make-ups in case of invalid excuses (invalid excuses mentioned below). You will need to notify the instructor two weeks in advance when you are not attending an exam.

Example of Invalid Excuses (from past experiences) include the following:

- Scheduled flights or trips
- Scheduled non-emergency doctor appointments
- Scheduling a job interview during class time (this case can have discussed with the instructor)
- WIFI/Internet connection problems leading to failure to submit homework, etc.
- Scheduled commitment during a class when you are required to take a quiz or an exam, or if scheduled to deliver a presentation, you will receive a zero for that grade.

Examples of "Excused" absences include the following:

- Illness or injury that is documented by a letter from a physician or health professional approved by the Department of Computer Science and the instructor.
- Emotional stress caused by a family problem (divorce, death of friend or family member) that is documented in writing and submitted to the instructor and the Department of Computer Science.
- Officially sponsored university athletic, music that is documented by a letter by the appropriate university official submitted and approved by instructor and the Department of Computer Science.
- Students are authorized to take excused absences each academic year for religious observances required by the student's faith. Students who wish to request more than two excused absences must contact their instructors for permission.

Homework and Assignment Submission Policy

- All assignments must be submitted through NYU classes. Assignments submitted by email will NOT be taken into consideration.
- Assignments must be submitted on the specified deadline. There is will be a -10pts per day for late submission. Submissions submitted after four days from the deadline will NOT be accepted.
- **We use automatic accredited tool that detect plagiarism and identify duplicates. You submission MUST be your own. I adopt the department academic integrity rules. Cheating will result be reported to the department for disciplinary action.**

Grading Policy

Quizzes

There will be NO make-up scheduled for missing quizzes. If you miss a quiz it will be a zero.

There will be a number of quizzes in this class. 50% of the quizzes will **NOT be announced**, 50% of the quizzes will be announced with a one session notice. For instance, you will be notified of a Thursday quiz on Tuesday (in case the class meets on Tuesdays and Thursdays). It is your responsibility to keep up with class announcements.

Your grade will be based on:

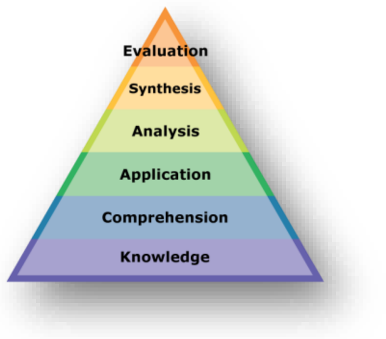
Final grades for the course will be determined using the following weights:

20% Programming assignments and homeworks
32% Midterm
35% Final Exam
10% Quizzes
3% Recitation attendance, class participation attendance and exercises

The following scale will be followed on assigning the final grade:

A	95-100
A-	90-95
B+	87-90
B	84-87
B-	80-84
C+	76-80
C	72-76
D	65-72
F	less than 65

Bloom's Taxonomy and Exams Questions Philosophy



I adopt bloom's taxonomy in this class as a framework for cognitive engagement, assessment and design of exams. The exam questions are designed in such a way to provide balance between basic learning of the computer science content learned in class *and high-level skills (beyond basics) such as analysis, synthesis and evaluation.*

Some examples in this section are from CS101.

Bloom's levels of assessment and computer science examples

Knowledge: At this cognitive level, the student is expected to simply recall information's, ideas and principles in the exact form in which they were learned in class. Exercises that mention verbs like: label, list, name, order, select, define, and among others, are of the knowledge level.

Computer science examples (knowledge level):

Define the object oriented programming paradigm.

Identify if the following statement is true or false: "An Abstract class can be instantiated."

Comprehension: In this stage the student is required to translate or interprets the material learned in class. Exercises that mention verbs like: explain, describe, give examples, rewrite, paraphrase and among others are of the comprehension level.

Computer science examples (comprehension level):

Rewrite code from do-while to while loop. (more details will be provided)

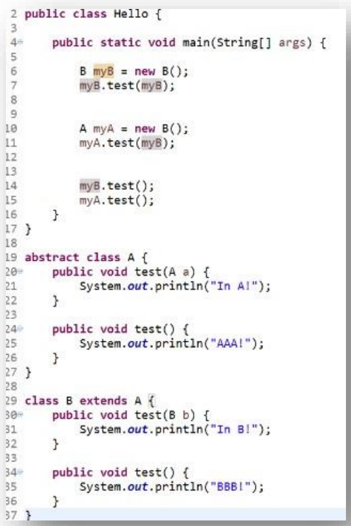
Explain in your own words the difference between "a class" and "an object". *Give two examples.*

Application: At this level, the student is expected to selects and uses principles learned in class to complete a task with minimum direction. Exercises that mention verbs like apply, change, compute, relate and among others are of the application level.

Computer science examples (application level):

Apply what you learned from the example we covered in class about implementing an ATM machine using methods in Java, to design a *Point of Sale* application in Java. (more details will be provided)

Analysis: At this level, the student should be able to relates assumptions, hypothesis, evidence or structure of a statement of question. Exercises that mention verbs like compare, contrast, relate, use, demonstrate, modify and among others are of the analysis level.



```java
public class Hello {

    public static void main(String[] args) {

        B myB = new B();
        myB.test(myB);


        A myA = new B();
        myA.test(myB);


        myB.test();
        myA.test();
    }
}

abstract class A {
    public void test(A a) {
        System.out.println("In A!");
    }

    public void test() {
        System.out.println("AAA!");
    }
}

class B extends A {
    public void test(B b) {
        System.out.println("In B!");
    }

    public void test() {
        System.out.println("BBB!");
    }
}
```

Computer science examples (application level):

Compare and contrast the procedural programming paradigm and the object oriented programming paradigm.

Synthesis: At this level, the student is expected to originate and combine ideas into a new product. Exercises that mention verbs like develop, create, explain, synthesize and among others are of the synthesis level.

Computer science examples (synthesis level):

Design an algorithm that sorts a singly LinkedList (Combining sorting and LinkedList)

You are helping your friend to develop a software application that manages products they sell in their store. Develop a software application using Java, OOP and a suitable data structure to full-fill the requirements of the following menu: (more details will be provide in real case scenario for this exercise)

The menu for the application looks like the following:

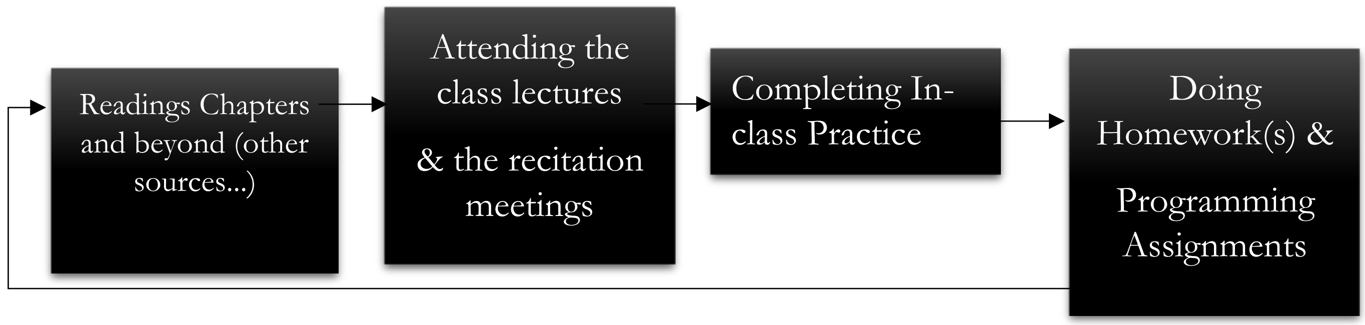
1. Add a Product
2. Edit a Product
3. Upload Inventory
4. View All Products that needs to be ordered

Evaluation: At this level, the student appraises, assesses or critique on a basic of specific learning outcomes, standards and criteria. Exercises that mention verbs like summarize, conclude, explain, interpret, select, recommend and among others are of the evaluation level.

Computer science examples (evaluation level):

Assess the code in the figure in the right. Will this code compile? If Yes, provide the output that will be printed. If the answer is NO, briefly explain the error.

Student's Basic Requirements to Succeed in 102


Readings Chapters and beyond (other sources...)
Attending the class lectures & the recitation meetings
Completing In-class Practice
Doing Homework(s) & Programming Assignments

Basic requirements for an effective computer science class include *reading chapters and other sources* prior to the lecture: You will be assigned readings from the book, you are encouraged to read beyond the book and be **proactive**. You might be asked to *hand-write* summaries of some chapters from the book (they will be counted as bonuses). You will need to make sure to *attend every class*, *complete the in-class practice*, *attend the recitation*, and *spend substantial time in doing your homework and the programming assignments*.