Introduction

to Data Structures, Fall 2022

Instructors: Marina Barsky, Sam Taggart
The course is about

- Data structures
- Algorithms
- Java
The course is about

- Data structures
  - Algorithms
  - Java
What is a Data Structure?

• Each program works on data: takes input data and produces output data

• There are sophisticated ways to structure data in memory – that makes program efficient

The choice of a suitable data structure can make all the difference between a working and a failing program

Example: Most Frequent Word
Many Data Structures exist

- Simple: *arrays, linked lists, stacks, queues*
- More intricate - but still very useful: *heaps, search trees, hash tables*
- Advanced - *Bloom filters, union-find* ...

- Why do we need so many? Because different data structures support different sets of operations and are good for different types of tasks.
We need to know what exists and what it is good for

• We will discuss the pros and cons of each data structure for a particular task
• The fewer operations the data structure supports - the faster these operations will be

The skill: think about the operations that you need for solving a problem

Choose the best data structure - the one that supports only required operations, and not more.
Four levels of Data Structure Proficiency

- Level 0: ignorance
- Level 1: cocktail party awareness
- Level 2: solid literacy: know which data structures are appropriate for which types of tasks and comfortable using them
- Level 3: hardcore programmers and computer scientists: understand the internals of existing and implement new data structures

We aim here...
The course is about

• Data structures

➤ Algorithms

• Java
Why algorithms?

• **Algorithm** is a sequence of steps that converts input data into a desired output

• We will get familiar with algorithms for operating on different data structures

• We will study the basics of *Algorithm Analysis*, and compare performance of different data structures for a given task
The course is about

- Data structures
- Algorithms
- Java
Why Java?

- Simple **typed** language
- Fully **Object-Oriented**
- Takes care of memory with **Garbage collector**

- Contains multiple implementations of **ready-to-use Data Structures**
- We will learn which structures are available in the Java Developer Kit, so you won’t waste time reinventing the wheel
- But we will also **implement our own Data Structures** from scratch
Why Java? Useful on its own

• High-level language – concentrate on a task not on the machine
• Building programs from interacting objects → large projects with short schedule: divide work into components
• Java is used to build long-lived, reliable, modifiable software
Course Outline

• **Java** first:
  • Principles of object oriented program design

• **Algorithms** second:
  • Sorting and searching
  • Recursion
  • Analysis

• **Basic structures**
  • Arrays, Lists, Queues, Stacks

• **Advanced structures**
  • Trees, Heaps, Maps, Graphs
Course Mechanics
Where is everything!?

• BLACKBOARD: LINK
  • Announcements
  • Lab submissions
  • Your grades

• WEBSITE
  • https://cs.oberlin.edu/~mbarsky/classes/cs-151/f2022/
  • Links to (virtually) all course content
  • Schedule by weeks
Textbook

Zybook:
https://learn.zybooks.com/zybook/OBELINCSCI151Fall2022

Participation activities: due before class
Help

• Discussion forum: Piazza
• Instructor office hours
• Weekly problem-solving sessions with OWLs
• Lab-helper hours
• Individual tutoring

The details and links are in the Course Syllabus: LINK
Grading

• ~10 programming assignments – 40 %
• Preparation exercises – 10 %
• Class participation – 10 %
• Midterm exam – 20 %
• Final exam – 20 %
Late submission policies

• 3 grace tokens: 3 days late with no penalty
  Must fill in the form on blackboard before the due date

• 2 resubmissions
  Can earn up to 50% of lost points
Honor code

• The course grade is largely based on programming assignments, all must be your own work.

• We believe that you are here because you want to become a skillful Computer Scientist:

  Be honorable: do not copy solutions from each other

Plagiarism detection with MOSS (Measure of Software Similarity): http://theory.stanford.edu/~aiken/moss/

Punishments: zero grade, penalty grade, suspension (no jail time)
Your typical weekly workflow

• Before coming to class - read 3-5 sections of the book and answer questions (this does not start before week 3)
• Come to the lecture, listen, ask questions, and engage in at least 75% of class activities
• Come to the lab and finish at least the first part during lab time with the help of the instructor (the weekly labs start on September 12)
• Continue working on the lab and submit your solution on the due date
• Have fun!

Our first class activity: class profile

Section 10 AM

Section 11 AM
Hello Java!

https://github.com/mgbarsky/cs151_data_structure_demos/tree/main/0.hello
public class Hello {
    public static void main(String[] args) {
        System.out.println("Hello, CS151!");
    }
}

/*
 * Hello.java
 * Author: CS 151 staff
 * Fall 2022
 * Prints a welcome message to the terminal
 */
Edit/Compile/Run cycle

• Edit: Save Java source code in file Hello.java
• Compile: `javac Hello.java`
  • Produces Java bytecode file named `Hello.class`
• Execute: `java Hello`
  • Searches `Hello.class` for a method with signature `public static void main(String[])`
  • Executes that method (if it exists)
Hello1.java

/*
 * This program prints two first program arguments to the terminal.
 */

public class Hello1 {
    public static void main(String[] args) {
        System.out.print(args[0] + " ");
        System.out.print(args[1]);
        System.out.println();
    }
}

Hello2.java

/*
 * This program echoes the arguments provided on the command line.
 */

public class Hello2 {
    public static void main(String[] CLParams) {
        int i = 0;
        while (i < CLParams.length) {
            System.out.print( CLParams[i] + " ");
            i++;
        }
        System.out.println();
    }
}
Notes

• Changed args to CLParams

• Every array knows its size: CLParams.length
  • It’s a data member, not a method call

• Java while loop

  \[\text{\textit{initialization}};\]
  \[\text{\textnormal{while (continuation) }} \{
  \text{\textit{statement}}; \ldots \text{\textit{statement}};\]
  \[\text{\textit{update}};\]
  \}\]

• Equivalent to Java for loop

  \[\text{\textnormal{for(\textit{initialization}; \textit{continuation}; \textit{update})}}\]
  \[\{ \text{\textit{statement}}; \ldots \text{\textit{statement}}; \}\]
Hello3.java

/*
 * This program echoes all arguments provided on the command line.
 */

public class Hello3 {

    public static void main(String[] CLParams) {
        for (int i = 0; i < CLParams.length; i++) {
            System.out.print( CLParams[i] + " ");
        }
    }

    System.out.println();

}
Hello4.java

/*
 * This program echoes all arguments provided on the command line.
 * It also prints a message suggesting how to properly use the program.
 */
public class Hello4 {

    public static void main(String[] CLParams) {
        if(CLParams.length == 0) {
            System.out.println("Usage: java Hello5 string1 ...");
        } else {
            for(int i = 0; i < CLParams.length; i++) {
                System.out.print(CLParams[i] + " ");
            }
            System.out.println();
        }
    }
}

Hello4.java

/*
 * This program echoes all arguments provided on the command line.
 * It also prints a message suggesting how to properly use the program.
 */

public class Hello4 {

    public static void main(String[] CLParams) {
        if (CLParams.length == 0) {
            System.out.println("Usage: java Hello5 string1 ...");
        } else {
            for (int i = 0; i < CLParams.length; i++) {
                System.out.print(CLParams[i] + " ");
            }
            System.out.println();
        }
    }
}

{} can be omitted for single-statement blocks
Notes

• Multi- and single-line comments: /* .. */ or //

• Code must be wrapped in a class declaration
  Everything is (in) a class in Java

• File name should be same as declared class name

• System is a Java object holding another object called out. out is of type PrintStream
  • PrintStreams provide many methods, including print() and println()
Notes (cont.)

• We can pass String values into the program through the *args parameters of the *main method

• The parameter *args is an array of String
  • It is passed to the *main method from the *command line
  • Contains every string on the command line after *java Hello

• The name *args can be replaced with any other variable name…

• More about String [] *args
  • Every array has an associated variable (instance variable) called *length, which holds the size of the array
  • Array indexing, as in C and Python, starts at 0
  • String, unlike int, is a class-based type, not a primitive type
    • More on this soon…. 
To do list

• **Register** for the course (if not already registered)
  
  We can discuss your individual situation on Wednesday during office hours

• Locate the course on the **blackboard**

• Register for the **Piazza** forum and post something fun

• Carefully read the **syllabus** and prepare questions

• **Read the code** for Hello Java

• Optional: **read Handout 1 “Java essentials”**