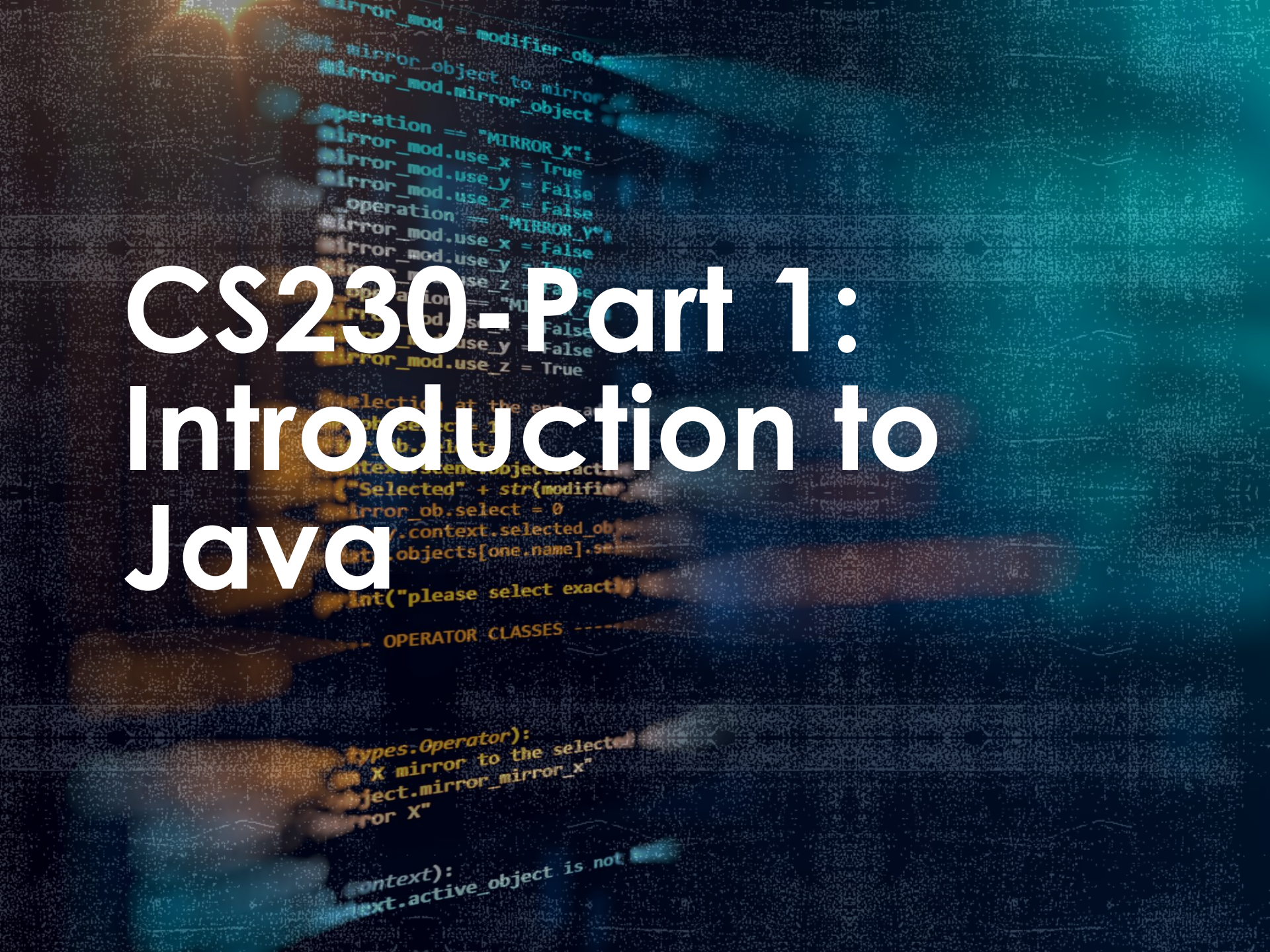
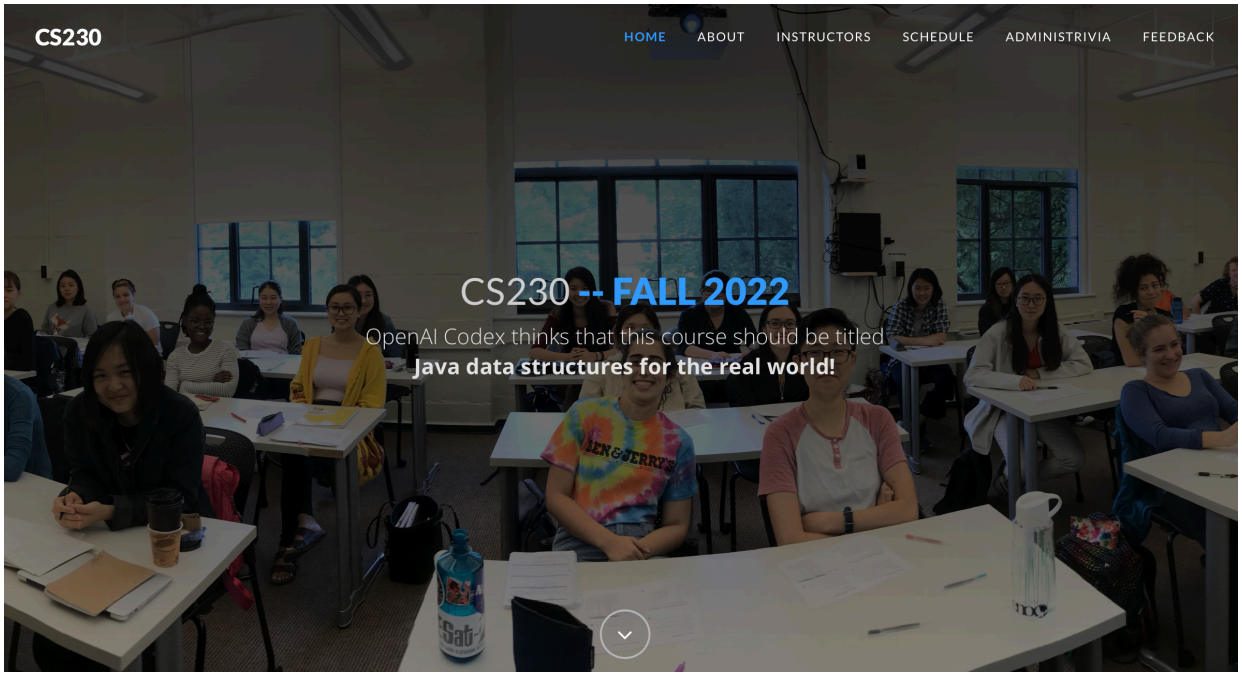


CS230-Part 1: Introduction to Java

Welcome to CS230

- Email cs230instructors@wellesley.edu to get the quickest answer for all course related questions
- Class website: https://cs.wellesley.edu/~cs230/

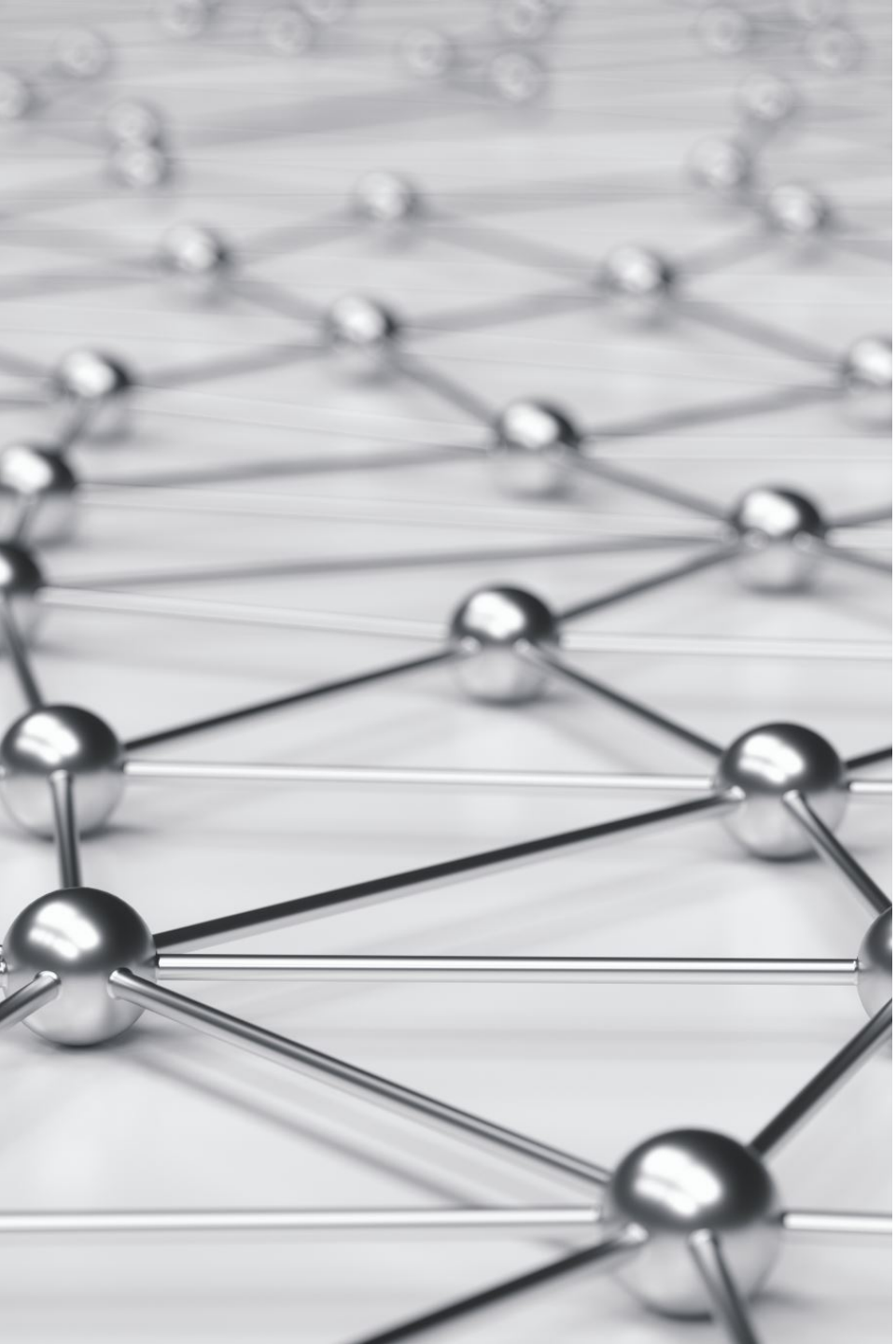


Class sections

Section 01 (11:20am T-F)
Section 02 (12:45pm T-F)
Section 03 (2:20pm T-F)
WITH REQUIRED LAB (Wednesdays)

MONDAY	TUESDAY	WEDNESDAY LABS	THURSDAY	FRIDAY
SEP05	**SEP06**	**SEP07**	**SEP08**	**SEP09**
	Classes start L1: Intro to CS230, Java data types **Reading:** LDC Ch 1 & 2 (slides / code)	Lab 0: BlueJ, First java program, compilation, Strings, Primitive data types. Java API	CS230-M first lecture Location TBD	L2: Using Java classes (Math, Random, String), operators, conditionals and loops **Reading:** LDC Ch 3 & 4. (Optional: 3.6-3.7) (slides / code)

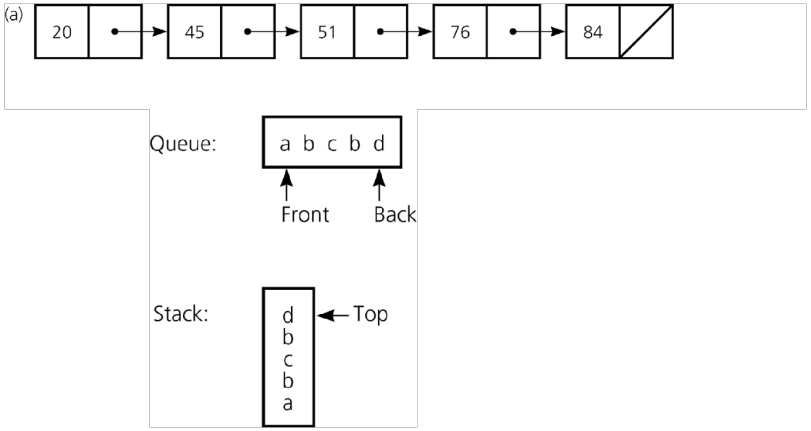
Section M (2:20 M-Th, no lab)
For students with prior Java experience
sign-up here

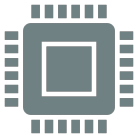


Why take CS230?

- You will learn the "big picture" of programming
  - Data abstraction
  - Modularity
  - Performance Analysis
  - Basic abstract data types (ADTs)
- You will become a more competent programmer
  - You will also become a designer, tester, analyzer, debugger, team member
- You will have fun in the process!

Why Abstract Data Types (ADTs)?





Allow you to write complex programs easier

To keep mental track of complex data interaction

To reuse code (yes!)

To improve code performance

Allows modularity of large projects

Easier to understand large chunks of code

Easier to collaborate with large teams

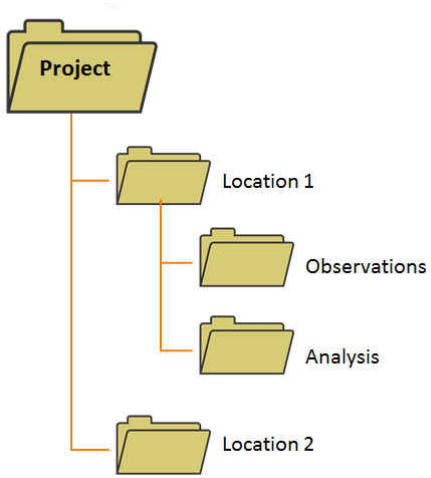
Some basic ADTs:

- Collections
- Linked list
- Stack
- Queue
- Table
- Priority queue

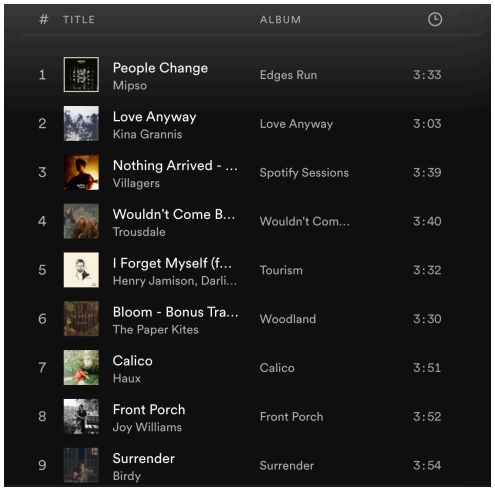
Not so basic:

- Tree
- Set
- Graph

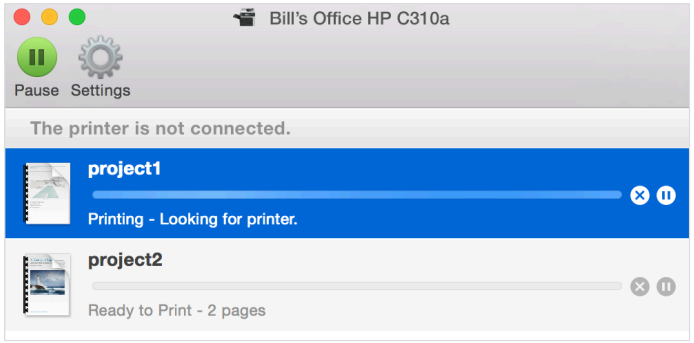
Application examples of ADTs



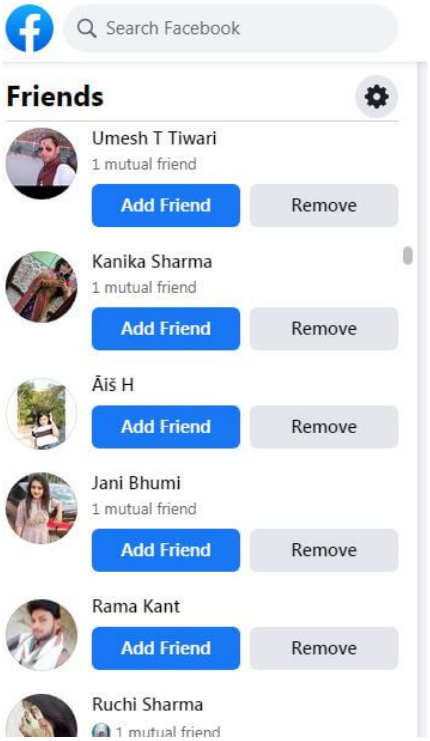
Directory tree



Music playlist list



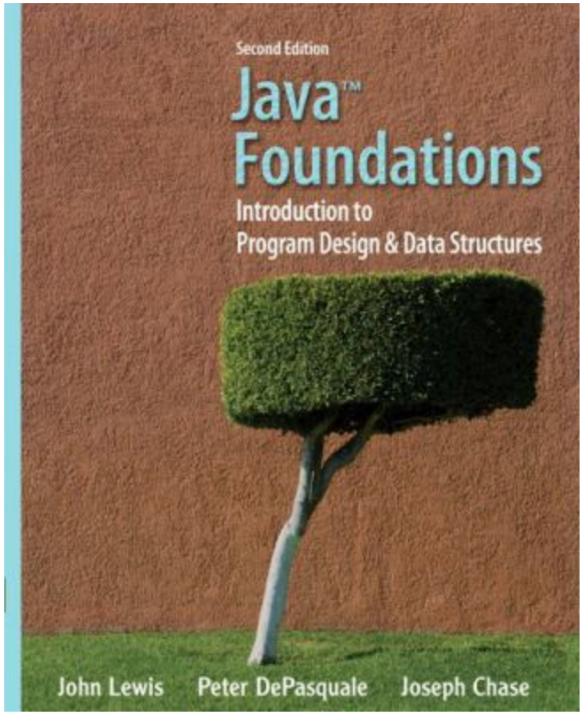
Printer queue

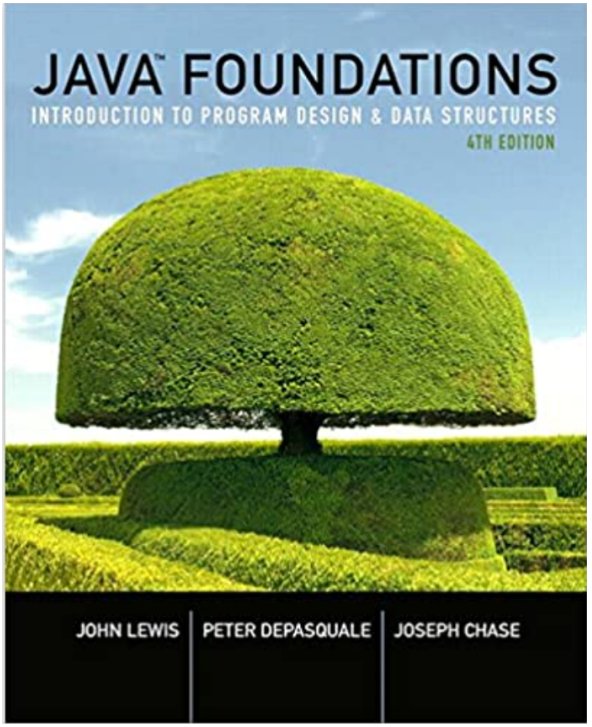


Social media friendship graph

Textbook

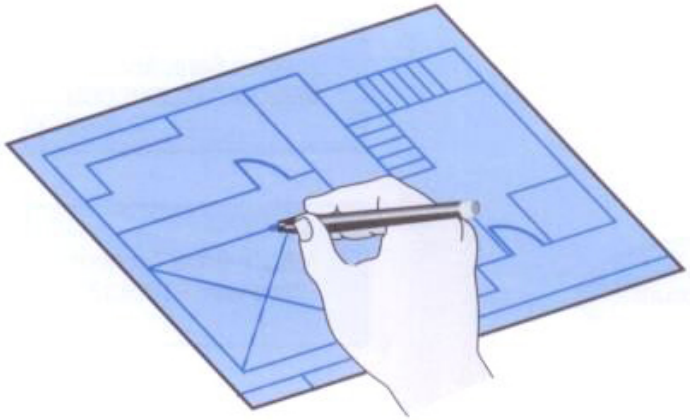
- Java Foundations
  - by Lewis, DePasquale and Chase
  - 2nd edition or higher





Java has Classes and Objects

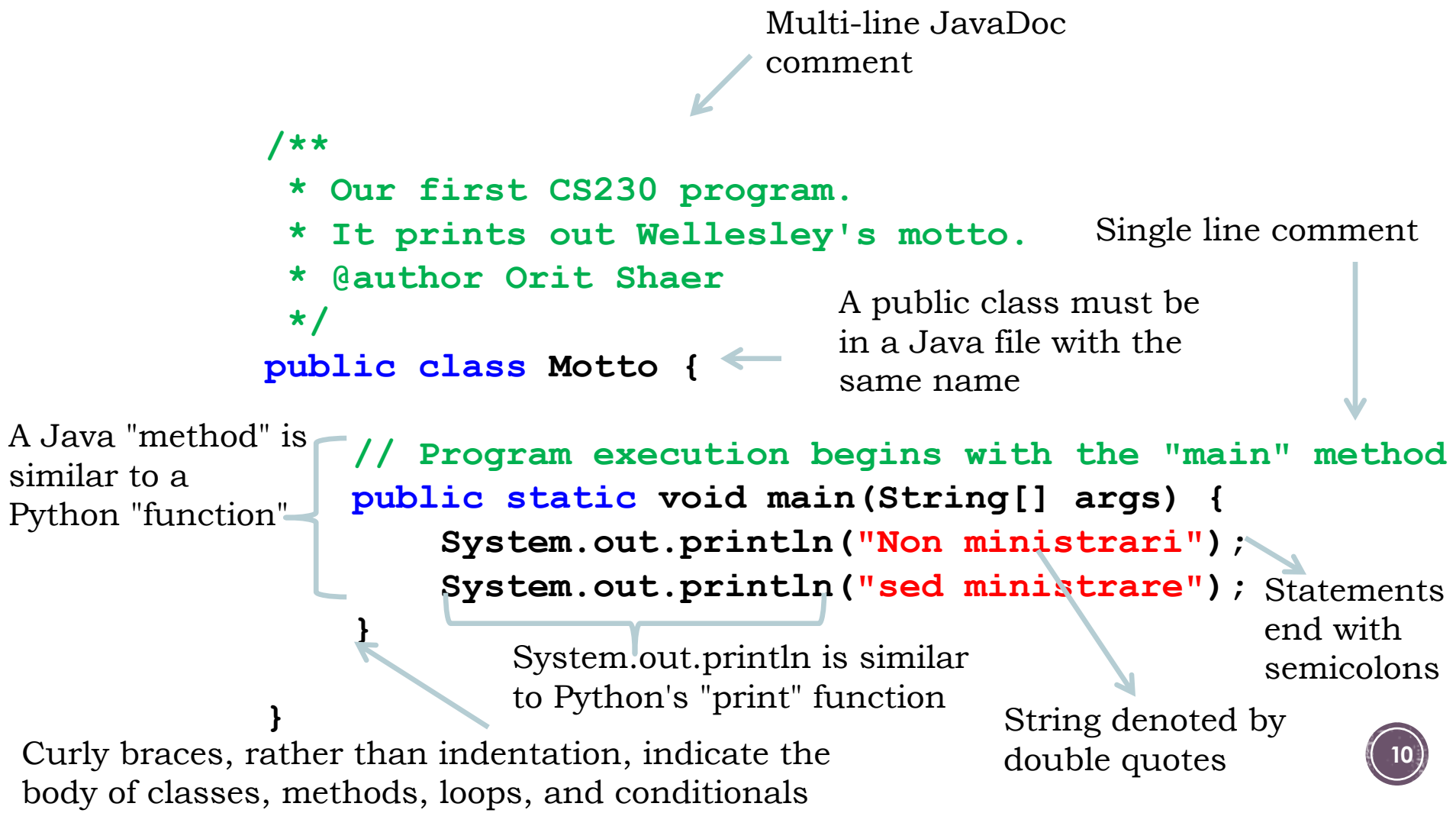
- A class is like a blueprint from which you can create many of the "same" house with different characteristics



Java syntax

Your first program

A First Program: Motto.java



Taking the High Road

- The native language of a computer is a low-level language. E.g.,

```
# Store the sum of a and b in c
load r4, a
load r5, b
add r4, r5
store r4, c
```

- Java is a high-level language designed for people. E.g.,

```
// Store the sum of a and b in c
c = a + b;
```

- To get from high to low a translator is needed.

Interpreters

Source code

```
/* Our first CS230 program.
 * It prints out Wellesley's motto.
 */
public class Motto {

    // Program execution begins with the "main" method
    public static void main(String[] args) {
        System.out.println("Non ministrari");
        System.out.println("sed ministrare");
    }

}
```

Interpreter

```
Non ministrari
sed ministrare
```

Results

Compilers

Source code

```
/* Our first CS230 program.
 * It prints out Wellesley's motto.
 */
public class Motto {

    // Program execution begins with the "main" method
    public static void main(String[] args) {
        System.out.println("Non ministrari");
        System.out.println("sed ministrare");
    }

}
```

Compiler

```
01110100101
01001111010
00100011110
11010001011
00010001100
00100111100
00000111111
01111110011
```

Object code

Object code interpreter

```
Non ministrari
sed ministrare
```

Results

First compiler was created by Admiral Grace Murray Hopper in 1952.
(3 mins) https://www.youtube.com/watch?v=E3PjvadIlXE

Java Does Both

Source code (.java)

```
/* Our first CS230 program.
 * It prints out Wellesley's motto.
 */
public class Motto {

    // Program execution begins with the "main" method
    public static void main(String[] args) {
        System.out.println("Non ministrari");
        System.out.println("sed ministrare");
    }

}
```

Compiler

```
jtklew0er#2
^720[l8scWq
kls;wkjjh3?
nnmsllw7y0*
y%#*&jk23=}
(*kd1*8,<vV
p+}ke56&8)6
kls;wghjh3?
```

Java byte codes (.class)

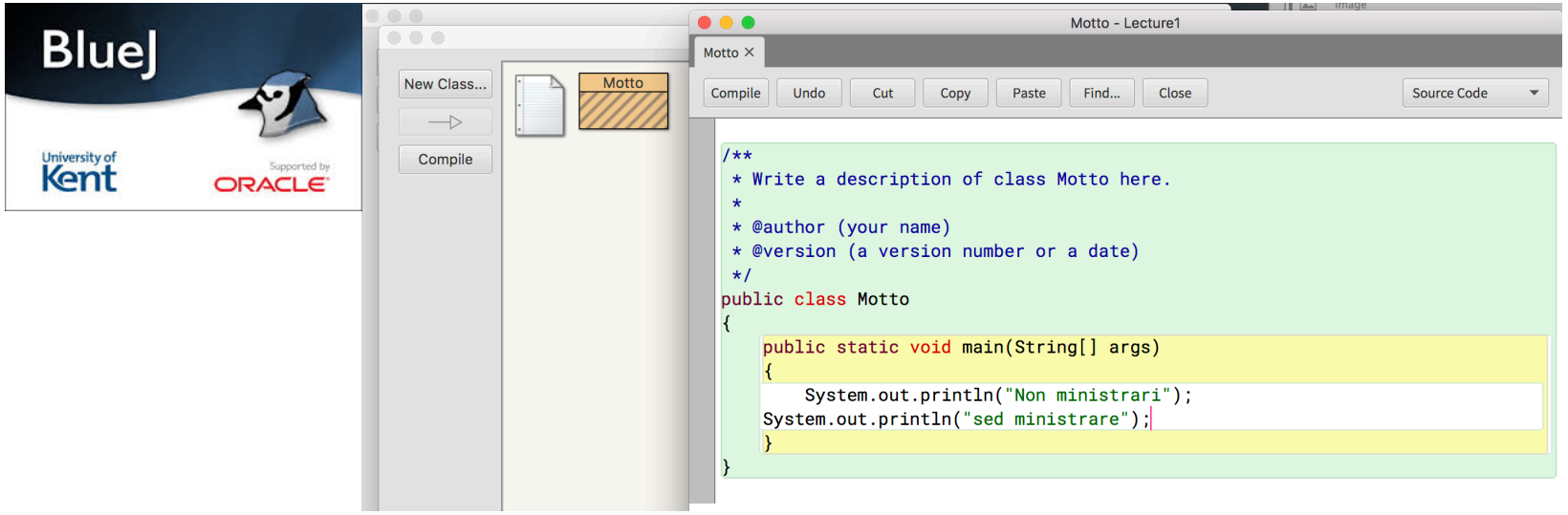
Interpreter (JVM = Java Virtual Machine)

```
Non ministrari
sed ministrare
```

Results

Using Java and BlueJ

- You can study **data structures** using *any language*
  - in this course we use **Java**
- You can **write and execute Java programs** in many ways, on the command line or with an IDE
  - in this course we will use a simple IDE called **BlueJ**



Variable Declaration in Java

- A **variable** must be declared once before it can be used
- The **type** of a variable cannot be changed after declaration
- The **value** of a variable can be changed many times

```
int x;
int y;
int z;
x = 7;
y = 5;
z = x + y;
System.out.println(z);
```

Variables declared and initialized in separate statements

```
int x = 7;
int y = 5;
int z = x + y;
System.out.println(z);
```

Variables declared and initialized in single statement

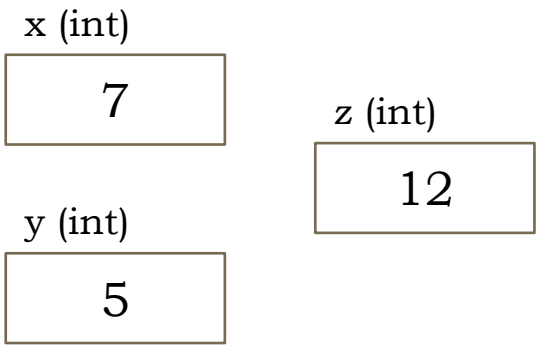
Memory models

- There is no one standard for writing memory models in CS, but in this course, we will follow a consistent diagram-based model.

- For example

```
int x;
int y;
int z;
x = 7;
y = 5;
z = x + y;
System.out.println(z);
```

- Will result in this final model



- And will result in “12” being printed out on the screen

Control flow

- Or in other words,
  - In what order does the code run to produce such an outcome?

- Java is a sequential language
  - Each line of code is executed in the order that it is written
    - For example, lines 1-7 below are executed one after the other
  - The compiler can jump from one part of the code to another if it encounters
    - Method calls
    - Conditionals
    - Loops

```
1. int x;
2. int y;
3. int z;
4. x = 7;
5. y = 5;
6. z = x + y;
7. System.out.println(z);
```

Control flow + memory models

- Again, for the same example,

```
1.  int x;
2.  int y;
3.  int z;
4.  x = 7;
5.  y = 5;
6.  z = x + y;
7.  System.out.println(z);
```

Line number	Line 1	Line 2	Line 3	Line 4	Line 5	Line 6	Line 7
Memory model of the current state of the program	x (int)	x (int) y (int)	x (int) y (int) z (int)	x (int) 7 y (int) z (int)	x (int) 7 y (int) 5 z (int)	x (int) 7 y (int) 5 z (int) 12	x (int) 7 y (int) 5 z (int) 12
Other actions							Print value of z to

Operator Precedence

- What is the order of evaluation in the following expressions?

```
a + b + c + d + e
```

```
a + b * c - d / e
```

```
a / (b + c) - d % e
```

```
a / (b * (c + (d - e)))
```

Find the Errors!

```
// This program has at least 5 errors. Can you
// find them all?
public class Errors {

      public static void main(String[] args)
            String temperature = 80.3;
            int n = 100
            n = “Wait, what?”;
            print(“This is fine.”);
}
```

Choose your own adventure…

- Create a group, find/create shared space/document
- Write your own Java program to calculate some value and print it out
- Some ideas (or make your own!):
  - Area of a circle (or other shapes) given its radius (or other necessary dimensions)
  - Volume of a box/sphere/cylinder of some given dimensions
  - Simple interest given amount, rate, time