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)**

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

**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
- Social media friendship graph
- Printer queue
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

```java
/**
 * Our first CS230 program.
 * It prints out Wellesley's motto.
 * @author Orit Shaer
 */

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");
    }
}
```

A Java "method" is similar to a Python "function"

Curly braces, rather than indentation, indicate the body of classes, methods, loops, and conditionals

A public class must be in a Java file with the same name

Multi-line JavaDoc comment

Single line comment

String denoted by double quotes

Statements end with semicolons
The native language of a computer is a **low-level language**. E.g.,

```c
# 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.,

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

To get from high to low a **translator** is needed.
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");
    }

}
/* 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 ministrarë");
        System.out.println("sed ministrare");
    }
}

Non ministrari
sed ministrare

Results

Object code

Source code

Compiler

Object code
First compiler was created by Admiral Grace Murray Hopper in 1952. (3 mins) https://www.youtube.com/watch?v=E3PjvadIIXE
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

Results (JVM = Java Virtual Machine)

Interpreter

Java byte codes (.class)
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

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

Variables declared and initialized in separate statements

```java
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
  ```java
  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,

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

<table>
<thead>
<tr>
<th>Line number</th>
<th>Line 1</th>
<th>Line 2</th>
<th>Line 3</th>
<th>Line 4</th>
<th>Line 5</th>
<th>Line 6</th>
<th>Line 7</th>
</tr>
</thead>
<tbody>
<tr>
<td>Memory model of the current state of the program</td>
<td>x (int)</td>
<td>x (int)</td>
<td>x (int)</td>
<td>x (int)</td>
<td>x (int)</td>
<td>x (int)</td>
<td>x (int)</td>
</tr>
<tr>
<td></td>
<td>y (int)</td>
<td>y (int)</td>
<td>y (int)</td>
<td>y (int)</td>
<td>y (int)</td>
<td>y (int)</td>
<td>y (int)</td>
</tr>
<tr>
<td></td>
<td>z (int)</td>
<td>z (int)</td>
<td>z (int)</td>
<td>z (int)</td>
<td>z (int)</td>
<td>z (int)</td>
<td>z (int)</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>7</td>
<td>7</td>
<td>7</td>
</tr>
<tr>
<td>Other actions</td>
<td>Print value of z to screen</td>
<td>Print value of z to screen</td>
<td>Print value of z to screen</td>
<td>Print value of z to screen</td>
<td>Print value of z to screen</td>
<td>Print value of z to screen</td>
<td>Print value of z to screen</td>
</tr>
</tbody>
</table>
Operator Precedence

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

\[ a + b + c + d + e \]
\[ a + b \times c - d / e \]
\[ a / (b + c) - d \% e \]
\[ a / (b \times (c + (d - e))) \]
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