WeTeach_Java

A Five-Day Java Workshop with John Owen

The University of Texas at Austin
Center for STEM Education
College of Education

WeTeach_CS

WeTeach_Java

A Five-Day Java Workshop

July 23-27, 2018
8:00 a.m. – 4:30 p.m.
Breakfast and lunch will be served

The University of Texas at Austin
Center for STEM Education
Sanchez Building, Room 324
Austin, TX 78712

Presenter: **John Owen**
CS PD Specialist
johnbowen@utexas.edu
@johnowenteachcs

AGENDA

Times	Monday Intro/Section A	Tuesday Section B	Wednesday Section C	Thursday Section D	Friday Section E
8:00 a.m.	Registration and Breakfast	Breakfast	Breakfast	Breakfast	Breakfast
8:30 a.m.	Welcome Introductions Overview Module Zero • Java Basics	Module Three A • Keyboard Input • Documentation	Module Five A • Math class methods	Module Six B • Counting and Accumulating Loops	Module Seven A,B • Arrays • Array methods
10:00 a.m.	*Break*	*Break*	*Break*	*Break*	*Break*
10:15 a.m.	Module One A,B,C,D • Output • print, println • printf • Calendar class	Module Three B,C • File Input • GUI Methods	Module Five B • String class methods	Module Six C • File Processing with loops	Module Eight A • Primitives, Objects, Scope Module Eight B • Passing Parameters
12:00 p.m.	*Lunch*	*Lunch*	*Lunch*	*Lunch*	*Lunch*
12:45 p.m.	Module Two A,B • Data • Operations	Module Four A • If and if else	Module Five C • Custom static methods	Module Six D • Using split with loops and arrays	Module Eight C • OOP Essentials
2:00 p.m.	*Break*	*Break*	*Break*	*Break*	*Break*
2:15 p.m.	Module Two C,D • Math Operations • Number Bases	Module Four B • Switch statement	Module Six A • Output Pattern Loops	• Work on unfinished labs	• Work on unfinished labs
4:30 p.m.	Adjourn	Adjourn	Adjourn	Adjourn	Adjourn

WeTeach_CS is headquartered at The University of Texas at Austin, and supported by state, federal, and corporate funders and partners. Proceeds may be used in general support of the College of Education.

WeTeach_Java

A Five-Day Java Workshop

Table of Contents

WeTeach_CS For HS

Curriculum Overview

Objective: Provide a comprehensive curriculum for a typical first year of high school computer science, covering approximately 160 days of instruction, originally developed to satisfy the Texas Computer Science 1 standards, but easily adaptable to any other set of standards.

Curriculum Outline: Four areas of focus provide the foundation for this curriculum package, with an estimated time duration provided for each:

- Scratch – an introduction to computational thinking and programming using a fun and easy block-based programming environment, developed by the Lifelong Kindergarten Group at the MIT Media Lab (https://scratch.mit.edu/) – 15 days
- Jeroo – an introduction to scripted and Object Oriented Programming using a graphic display, originally developed at Northwest Missouri State University (http://home.cc.gatech.edu/dorn/jeroo) – 15 days
- CS Concepts and Digital Citizenship – an offline module focused on various CS concepts and citizenship issues, including lessons on ethics, cybersecurity, and basic safety issues related to computing and the internet. – 10 days
- Java – A foundational study of this general purpose and industry standard language, exploring a very limited subset of this vast programming platform, enough to provide a solid basis for students to gain confidence in its use, in which further study in subsequent years will enable students to prepare for the CS AP test. – 120 days

Curriculum Platform and Elements: All materials for the curriculum will be available to subscribing teachers in an online format using Canvas, from which teachers will be able to facilitate a full-year first-year computer science course consisting of various learning elements, including:

- Videos
- Lessons
- Practice Exercises
- Labs
- Quizzes
- Tests
- Lab Tests
- Projects
- Year-long support through online Discussion Forum and Periodic Live Office Hours

My CS Teaching Philosophy

by John Owen

After twenty-plus years of teaching computer science, watching my students encounter their previously undiscovered talents in computational thinking, I have come to believe in some core principles of CS teaching. These are now imbedded and entrenched in my approach to helping other teachers in their quest to help their students embark and progress on this exciting journey of learning, and I feel strongly in these precepts.

- Every student can learn the basics of computational thinking.
- Many students can successfully and confidently complete a full year of computer science, equivalent to a typical CS1 course, given the right learning tools, curriculum, course content and sequence.
- All students bound for higher learning SHOULD complete at least one year of computer science before high school graduation. This should be a mandatory requirement, anywhere in the world.
- Computer Science is NOT easy and should not be used as a "dumping ground" for counselors who can't find another class in which to put a student. However, if that is the case, the teacher should consider it an opportunity to help student reach their learning potential and possibly find a "diamond in the rough" to develop and refine.
- Until CS is mandatory for all students, especially in new situations and in smaller schools where students have choices and are involved in many activities that take their time and energy, the CS teacher must find a **careful balance** between making it a fun course and expecting rigorous achievement, so that significant learning takes place, but not so much pressure is applied to cause students to give up and opt out of the class.
- One way to achieve this balance is to provide for a **liberal grading scheme**, where high expectations are set, but where effort is also rewarded, and a safety net is provided when failure occurs, which will be often. There is **no perfect grading system**, and the teacher must sometimes make a final subjective decision as to what mark a student has truly achieved.
    - A safety net example I have used in the past is to guarantee a final grade of 85%, regardless of the real outcome, as long as the student does his/her best and does not miss any work.
    - Another example is with labs and lab tests, where the student is given a problem to solve in a certain amount of time and is allowed multiple attempts to officially test their solution. When the student thinks they are ready, but the first testing attempt fails, they are given another opportunity, with a minor deduction to their final grade. This process repeats itself until the student is successful, with helpful hints provided along the way.
    - The offer of helpful hints can also be made when the student attempts a lab test, especially if they reach an impasse, and then they are given a choice to either take the helpful hint with a minor deduction or refuse the help and keep trying on their own. This way they have some control over their final result.
    - Another example is in written assessments, where students are encouraged to not use notes at first, but if they decide they need that crutch, are allowed to do so, with a deduction to the final grade. Again, this is their choice to make, which again gives them some control over their final grade.
- **Individuals fail. Collaboration fails.** All worthwhile efforts, whether by individuals, or in collaborative situations, involve failure. **This is a natural part of learning, and is not to be feared, avoided or belittled! To FAIL** is simply to take a **First Attempt At Learning** or at any other worthwhile effort (and sometimes a second, third and fourth, e.g. the invention of the lightbulb, Thomas Edison, 1000 failures, etc.). These multiple steps taking risks and resulting in failure will eventually lead to success given the right encouragement and additional opportunities. Any teacher must understand that students will fail sometimes, and to encourage them to keep giving best efforts until success is achieved.

WeTeach_Java
Workshop Outline

Section A

- **Module Zero – Java Basics**
  - Intro to Java
  - Java JDK and JCreator IDE Demo
  - jGRASP IDE Demo
  - Online IDE – repl.it Demo
- **Module One – Java Output and Formatting**
  - **Lesson 1A – Simple Output Basics –** output with println, error messages, debugging strategies
    - Lab 1A – Five Sentences
  - **Lesson 1B – More Output Techniques –** basic class structure, print vs println, tabs and escape sequences, data literals
    - Lab 1B1 – One Line, One Output
    - Lab 1B2 – One Line, Three Outputs
    - Lab 1B3 – Three Lines, One Output
    - Lab 1B4 – Tabs
    - Lab 1B5 – Return Address
    - Lab 1B6 – ASCII Face
  - **Lesson 1C – Using printf, Calendar –** more printf techniques, intro to Calendar class basics
    - Lab 1C1 – Three variables
    - Lab 1C2 – Name, Date, Time
    - Lab 1C3 – Three Decimal Formats
      - Calendar Class Reference page
  - **Lesson 1D – More printf, Calendar -** advanced printf techniques, more Calendar class output formats
    - Lab 1D1 – Calendar
    - Lab 1D2 – Receipt
- **Module Two – Data Types and Operations**
  - **Lesson 2A – Data –** Data Types, variables, constants, storage limits
    - Lab 2A1 – Constants
    - Lab 2A2 – Receipt Revisited
    - Lab 2A3 – MIN/MAX
  - **Lesson 2B – Operations –** Operators, characters, mixing data types, order of precedence
    - Lab 2B1 – Integer Equation
    - Lab 2B2 – ASCII Initials
    - Lab 2B3 – Decimal Equation
  - **Lesson 2C – Math Operations –** Shortcuts, casting, div, mod, Math class functions intro
    - Lab 2C1 – Math Ops
    - Lab 2C2 – Circle Area
    - Lab 2C3 – Square Area
    - Lab 2C4 – Birthday Magic
    - Lab 2C5 – Distance Formula
    - Lab 2C6 – NAAC
  - **Lesson 2D – Number Base Concepts –** offline – counting, twelve conversions
    - Counting in decimal, octal, hexadecimal and binary
    - Twelve conversion processes

Workshop Outline (cont.)

Section B

- **Module Three – Input from Keyboard and Files**
  - **Lesson 3A – Keyboard Input and Documentation** – **Using the Scanner class to create interactive programs, internal documentation techniques**
    - Lab 3A1 – Name
    - Lab 3A2 – Sum of Integers
    - Lab 3A3 – Circle Stuff
    - Lab 3A3.5 – Input Quirk – "when **nextLine** follows **next**, **nextInt**, or **nextDouble**, put it twice"
    - Lab 3A4 – Receipt, Again
    - Lab 3A5 – Seconds and Minutes
    - Lab 3A6 – Places, Please
    - Lab 3A7 – Change, Please
    - Lab 3A8 – Lyrical Spherical
  - **Lesson 3B – File Input** – **Using the Scanner class to harvest data from text files**
    Repeat all 3A labs using file input
    - Lab 3B1 – Name
    - Lab 3B 2 – Sum of Integers
    - Lab 3B 3 – Circle Stuff
    - Lab 3B3.5 – Input Quirk – "when **nextLine** follows **next**, **nextInt**, or **nextDouble**, put it twice"
    - Lab 3B4 – Receipt, Again
    - Lab 3B5 – Seconds and Minutes
    - Lab 3B6 – Places, Please
    - Lab 3B7 – Change, Please
    - Lab 3B8 – Lyrical Spherical
  - **Lesson 3C – GUI Methods with JFrames**
    - Video 1 with included lab – basics of JFrames, JPanels, and JLabels, interactive dialogues
    - Video 2 with included lab – basics of drawing with AWT (Abstract Windowing Toolkit) tools
- **Module Four – Conditionals**
  - **Lesson 4A – Using if and if else**
    - Lab 4A1 – Vote, Drive, Drink
    - Lab 4A2 – Div and Mod
    - Lab 4A3 – Phone Call 1
    - Lab 4A4 – Grade Message
    - Lab 4A5 – Line Slope
  - **Lesson 4B – Using switch statements**
    - Lab 4B1 – Phone Call 2
    - Lab 4B2 – Standard to Metric
    - Lab 4B3 – Hex Char
    - Lab 4B4 – Hurricane
    - Lab 4B5 – Salutations

Workshop Outline (cont.)

Section C

- **Module Five – Methods**
  - **Lesson 5A – Math class methods –** **sqrt, pow, abs, ceil, floor, sin, cos, tan, PI, E, and more**
    - Lab 5A1 – Absolute Distance
    - Lab 5A2 – Area, Volume, Side
    - Lab 5A3 – Roofing
    - Lab 5A4 – Johnny D
    - Lab 5A5 – Plane Distance
    - Lab 5A6 – Law of Cosines
    - Lab 5A7 – Chief SOHCAHTOA
  - **Lesson 5B – String class methods -** **length, charAt, indexOf, equals, substring, compareTo, and more**
    - Lab 5B1 – Return Address
    - Lab 5B2 – Upper/Lower
    - Lab 5B3 – Equal Strings
    - Lab 5B4 – Char Position
    - Lab 5B5 – Substrings
  - **Lesson 5C – Defining Custom Methods, JavaDoc API Utilities**
    - Lab 5C1 – Upper/Lower, version 2
    - Lab 5C2 – diffWords
    - Lab 5C3 – sameLength
    - Lab 5C4 – areaCircle
    - Lab 5C5 – surfAreaSphere
    - Lab 5C6 – delta
    - Lab 5C7 – slope

    From Codingbat.com
    - Lab 5C8 – helloName
    - Lab 5C9 – sleepIn
    - Lab 5C10 – cigarParty
- **Module Six – Loops**
  - **Lesson 6A – Output Pattern Loops –** **Fifteen output patterns using loops**
    - Lab 6A1 – row5Stars – using **while** loop
    - Lab 6A2 – row5Stars2 – using **do while** loop
    - Lab 6A3 – row5Stars3 – using **for** loop
    - Lab 6A4 – rowNStars – using **while** loop
    - Lab 6A5 – rowNStars2 – using **do while** loop
    - Lab 6A6 – rowNStars3 – using **for** loop
    - Lab 6A7 – colNStars – using **while** loop
    - Lab 6A8 – colNValues – using **do while** loop
    - Lab 6A9 – MtoNValues – using **for** loop
    - Lab 6A10 – MtoNEvens – using **while** loop
    - Lab 6A11 – MtoNOdds – using **do while** loop
    - Lab 6A12 – diagonalNStars
    - Lab 6A13 – reverseDiagonalNStars
    - Lab 6A14 – VofNStars
    - Lab 6A15 – diamondNStars

Workshop Outline (cont.)

Section D

- **Lesson 6B – Accumulation Loops –** solving practical problems using counting and accumulation loops
  - Lab 6B1 – sigmaN
  - Lab 6B2 – prodN
  - Lab 6B3 – countMultZfromXtoY
  - Lab 6B4 – populationGrowth
  - Lab 6B5 – manhattanCostX
  - Lab 6B6 – isPrime
  - Lab 6B7 – Euclid's algorithm
- **Lesson 6C – File Processing Loops –** a continuation of Lesson 3B for file input techniques
  - Lab 6C1 – Col to Row Doubles
  - Lab 6C2 – oddEven
  - Lab 6C3 – String length
  - Lab 6C4 – StringLength2
  - Lab 6C5 – Character Type
  - Lab 6C6 – Integer pairs
  - Lab 6C7 – Social Security
  - Lab 6C8 – Area/Circumference
- **Lesson 6D – Using String.split with loops –** harvesting single line multi-data text files using "split"
  - Lab 6D1 – AlphaOmega – input and process list of words using split
  - Lab 6D2 – dblArr – input and process horizontal list of decimal values
  - Lab 6D3 – avgArr – input values, calculate and output average

Workshop Outline (cont.)

Section E

- **Module Seven - Arrays and *Arrays* and *System* Class Methods**
  - **Lesson 7A – Arrays - A more formal discussion of arrays, first introduced in Lesson 6D**
    - Lab 7A1 – Student Names
    - Lab 7A2 – Student Data
    - Lab 7A3 – Class Average
    - Lab 7A4 – "split" review 1
    - Lab 7A5 – "split" review 2
    - Lab 7A6 – toCharArray
    - Lab 7A7 – Array of 10 Integers
    - Lab 7A8 – Array of Doubles
    - Lab 7A9 – Subjects
  - **Lesson 7B – *Arrays* and *System* Methods – using array manipulation methods from System and Arrays Library Classes**
    - Lab7B1 - Array from four ints
    - Lab7B2 - Combined arrays
- **Module Eight – Primitives, Objects, Scope, Parameters and OOP Essentials**
  - **Lesson 8A – Comparing and contrasting primitives and objects, demonstration of and discussion on scope**
  - **Lesson 8B – Discussion of actual, formal, value and reference parameters**
    - Lab 8B1 – Passing Primitives
    - Lab 8B2 – Passing Strings
    - Lab 8B3 – Passing Arrays
  - **Lesson 8C – Essentials of OOP**

WeTeach_Java

Notes

WeTeach_Java

Notes

WeTeach_Java

Notes