

# Computer Programming in Java

## Bush School *CPJava Fall 2020*

### *Block D*

Welcome to CPJava!  
Start in Remote Schooling format  
Follow in Hybrid format in Oct



*Chandru Narayan*  
*Bicyclist Astronomer Engineer*  
*Teaching Math and CS*





# Introductions!

1. State your name - how would you like to be addressed?
2. Your personal pronoun?
3. What are your hobbies?
4. Say something interesting or peculiar about yourself
5. Do you have any exposure to programming?
6. What made you choose this course?
7. What are your expectations from this course?



# A unique first course in programming with some advanced topics!

1. Visual and Project-based learning
2. Computing and the Web
3. Programming using Java
4. Simulate Natural Systems
5. Machine Learning
6. Develop Games
7. Learn Math & Physics Concepts
8. AP Exam preparation (optional)



# Course Requirements

1. The CPJava course does not require you to have prior programming experience
2. Prerequisites include Algebra 1
3. For students wanting to take the AP CSA exam there will some extra assignments that will need to complete



# 21st Century Expectations for Learning

- Advanced Java Programming
- Object Oriented including inheritance
- How to make and maintain your own website
- Some basic HTML & CSS
- How to use GitHub
- Searching and sorting
- Recursion, Fractals
- Robust code design and style
- APCS A topics
- Code Vectors, Newton's Laws, Machine Learning etc
- + Fun stuff like Asteroids, Super Mario Games and Computer Art that they don't teach you in college!



- The complete 1-year coursework is online!
- We'll start at the top and work our way to the bottom through this course

You are here! Click here to access.

Bookmark this  
You will need it every day!



**/ CPJava Course**  
Bush School Computer Programming in Java Course

[View on GitHub](#)

---

[Bush School CPJava Fall Semester 2020](#)



[Click here for live code and click bubbles inside resulting tab or window](#)

**CPJava - Computer Programming in Java Course**

This course is designed to introduce computer programming in the Java language. Learning to use a computer language is a necessary skill for all students regardless of discipline. In this course we will teach the fundamentals of computer programming from the stand point of simulation, automation, and problem solving of real-world systems and natural processes. At the same time, the design and implementation of computer programs is taught from the context of fundamental aspects of computer science, including the development and analysis of algorithms, the development and use of fundamental data structures, the study of standard algorithms and typical applications, and the use of logic and formal methods.

In addition, the year-long course will cover many of the topics necessary for preparation to the AP Computer Science A (APCSA) examination in Spring of the following year. This is an introductory course in computer programming using Java. As such, no specific programming prerequisites are needed to take this course. However, additional preparation may be needed to fully prepare a student for the AP CSA exam with no prior knowledge of computer programming.

# Course Schedule

LESSON UNITS	APPROXIMATE DURATION	TOPICS
<b>FALL 2020 SEMESTER</b>	<b>13-Weeks</b>	<b>Sep to Dec 2020</b>
<b>Unit 1</b>	3-Weeks	Introduction to Java, Tools walkthrough, Classroom processes, Environment setup, Java Primitive types and Operators
<b>Unit 2</b>	2-Weeks	Objects, Methods, String, Integer, Double, Math
<b>Unit 3</b>	2-Weeks	Control flow, Booleans, If statements, Object traversals, String, Integer, Double, Math
<b>Unit 4</b>	3-Weeks	Iteration, While loops, For loops, Nested Loops, Loop Analysis
<b>Unit 5</b>	3-Weeks	Anatomy of a class, Constructor, Accessor, Mutator Methods, this keyword, Social Impact of CS
<b>SPRING 2021 SEMESTER</b>	<b>14-Weeks</b>	<b>Jan to May 2021</b>
<b>Unit 6</b>	3-Weeks	Arrays, Array Lists, Array Traversal, Final Exam or Project
<b>Unit 7</b>	3-Weeks	Searching Sorting Algorithms, Ethics of Data Collection, Privacy
<b>Unit 8</b>	2-Weeks	2D Arrays, Traversal
<b>Unit 9</b>	3-Weeks	Projects, Inheritance, Encapsulation, Hierarchy, Polymorphism
<b>Unit 10</b>	2-Weeks	Recursion, Recursive Search. Sort
<b>Unit 11</b>	1-Weeks	Final Exam or Project. Peer Sharing Final Project Presentation

A complete Syllabus is available at the [CPJava Course Website](#)



# Grading Policy

Grades are cumulative for the full-year

## Fall Semester

40% Projects

40% Classwork / Assignments

20% Student Portfolio

## Spring Semester

TBD



# Grading Policy

Points are assigned for:

Completing Exercises  
Frequency of Code Updates  
Submitting Program Assignments  
Quizzes and Tests  
Paired Programming  
Professionalism & Integrity

AP CSA exam is on May 6th 2021  
Talk to me if you are going to take it



# How to succeed

Simply write lots and lots of Java code

Publish all of your work to the web (Github)

You cannot learn programming by reading or watching!

Make lots of mistakes - truly the best way to learn to code!

Working cooperatively with your peers - Pair Programming!

Don't be afraid to try new things!



# Professionalism

- How you conduct yourself during your work
- Includes (but not limited to)
  - Phone and Zoom etiquette
  - Reliability and accountability
  - Ethics
  - Working cooperatively with your peers



# Office Hours and Class Assistance

- Class Hours
  - Tues 9:00 - 10:10 [Zoom Sync sessions](#)
  - Tues 1:20 - 2:30 [Zoom Sync sessions](#)
- Wed Async
  - Not time specific - work individually
  - [Wed Zoom Scheduler upon request](#)
- 1-1 Conference and Office Hours
  - M-W-T-F 2:30 - 3:30 or Ad-hoc
  - [Ad-Hoc Zoom Scheduler upon request](#)
- Class Assistance
  - Mark Parrish is the TA for this class!
  - Contact us via [Slack](#) (preferred) or eMail



# Tools/Resources for CPJava

Zoom - Live and Recorded Class Sessions

CPJava website - Lesson Plans and Projects

Bush Portal - Course Syllabus, Schedule and Grades

CPJava Google Classroom - Classroom code nnqro5m

Applications for Laptop

- Processing - Java Editor / Compiler

- Slack, Trinket, repl.it, Github

- See Instructions in Google Classroom

Runestone BUSHSCHOOL\_CPJAVA Online course

- Online Textbook, Lessons, Exercises, Detailed Reference



# System Requirements

An Internet connection

A robust wireless connection is preferred

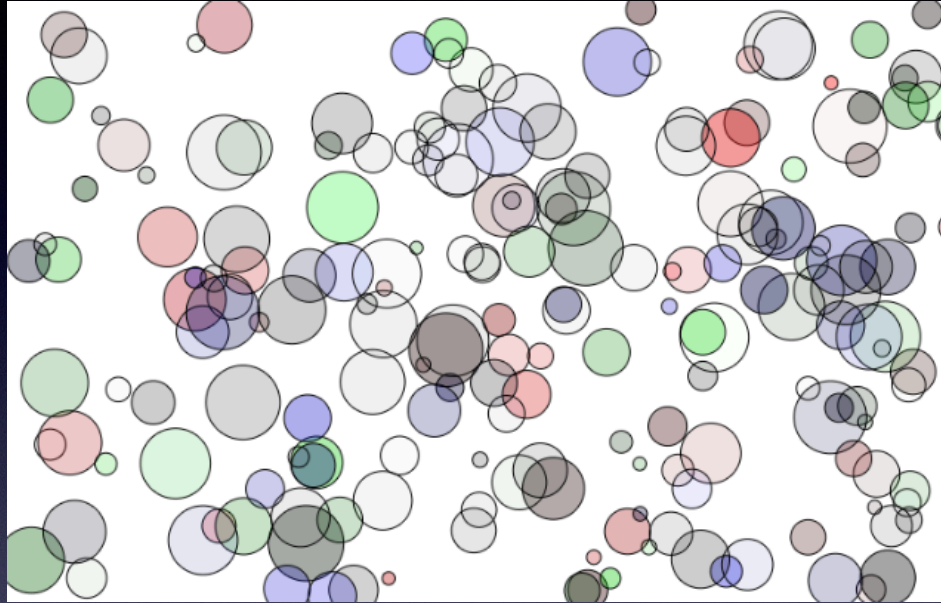
A laptop Computer - Mac or Windows  
(Chromebook may not be adequate)

Sufficient disk space, a functioning laptop battery

Functioning Camera and Microphone - especially  
required for Paired Programming



# Let's run our first Java Program!



Bubbles - click to run the program

You can sort by color (key 's') and freeze each bubble by clicking on it

Can you freeze all the bubbles?

You will develop visual code like this in Java and much more!



# Let's write our First Java Program!

Go to the Google Classroom (GCL)

- Complete Pre-Course pre-survey
- Install Processing - Java Editor / Compiler
- Write first Java code!





# Mood Meter Check-IN

Go to  
[www.menti.com](https://www.menti.com)  
code: 36 47 31 7

**Type in a single word up to 3 times that expresses your mood this morning!**

[Live Results](#)



# Where we left off

Access Intro Exercises in [Google Classroom](#)!

For the first 2 weeks you are working on:

1. Pre-course survey (Complete!)
  - [schedule 1-1 with me from Portal](#)
2. [Learn about Google Classroom](#) (Complete!)
3. [Intro Videos to Watch](#) (6 minutes total)
4. [Setup and Regn -Part 1](#) (Complete!)
5. [First Hello and Sticks Java Program](#) (Complete!)



# What are we doing Today?

1. A very brief History of Computer Science
2. [Java Basics - Four-Fours](#) (Start now in class - complete Friday)
  - Setup Part 2
    - Install Github, Repl.IT, Runestone csawesome, Codingbat
  - Complete Runestone assignment
  - Start Four-Fours assignment



**So we begin ....**

# **Overview of Computer Science**



# What is computer science?

- Computer Science
  - The study of theoretical foundations of information and computation and their implementation and application in computer systems. -- Wikipedia
  - Many subfields
    - Graphics, Computer Vision
    - Artificial Intelligence
    - Scientific Computing
    - Robotics
    - Databases, Data Mining
    - Computational Linguistics, Natural Language Processing ...
- Computer Engineering
  - Overlap with CS and EE; emphasizes hardware



# What is programming?

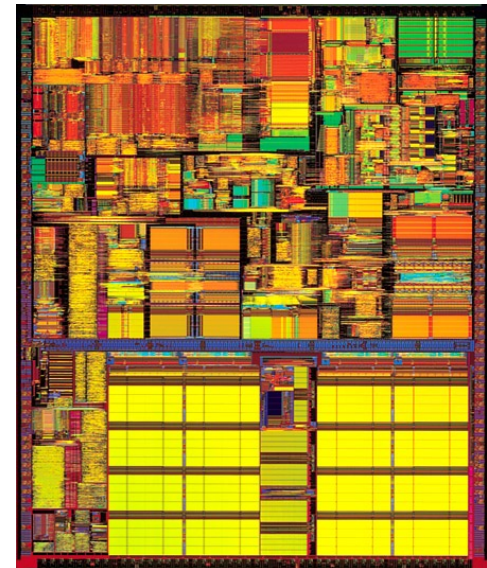
- **program:** A set of instructions to be carried out by a computer.
- **program execution:** The act of carrying out the instructions contained in a program.
- **programming language:** A systematic set of rules used to describe computations in a format that is editable by humans.





# Processors

- Computers are fast
  - Pentium 4 chip from 2001 can perform approximately 1,700,000,000 (1.7 GHz) computations per second
  - made up of 42,000,000 transistors (a switch that is on or off)
- Computers are dumb
  - They can only carry out a very limited set of instructions
    - on the order of 100 or so depending on the computer's processor
    - machine language instructions, aka instruction set architecture (ISA)
    - Add, Branch, Jump, Get Data, Get Instruction, Store



# Neumann's Idea

- John von Neumann - co-author of paper in 1946 with Arthur W. Burks and Hermann H. Goldstine,
  - "Preliminary Discussion of the Logical Design of an Electronic Computing Instrument"
- One of the key points
  - program commands and data stored as sequences of bits in the computer's memory

- A program:

```
1110001100000000
0101011011100000
0110100001000000
0000100000001000
0001011011000100
0001001001100001
0110100001000000
```





# Low Level Programming

- Programming with Strings of bits (1s or 0s) is not the easiest thing in the world.
- Assembly language
  - mnemonics for machine language instructions

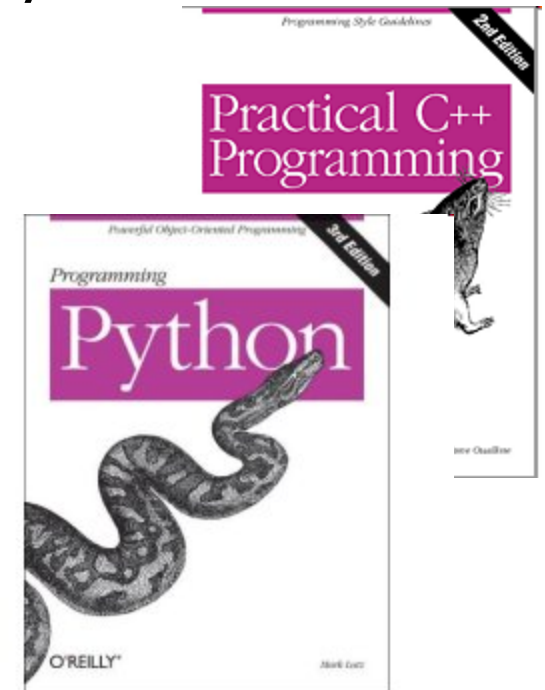
```
                .ORIG          x3001
                LD              R1, x3100
                AND             R3, R3 #0
                LD              R4, R1
                BRn             x3008
ADD             R3, R3, R4
                ADD             R1, R1, #1
LD              R4, R1
                BRnzp          x3003
```



# High Level Programming

- Assembly language, still not so easy, and lots of commands to accomplish things
- High Level Computer Languages provide the ability to accomplish a lot with fewer commands than machine or assembly language in a way that is hopefully easier to understand:

```
int sum = 0;
int count = 1;
while(count <= 100) {
    sum += count;
    count += 1;
}
```



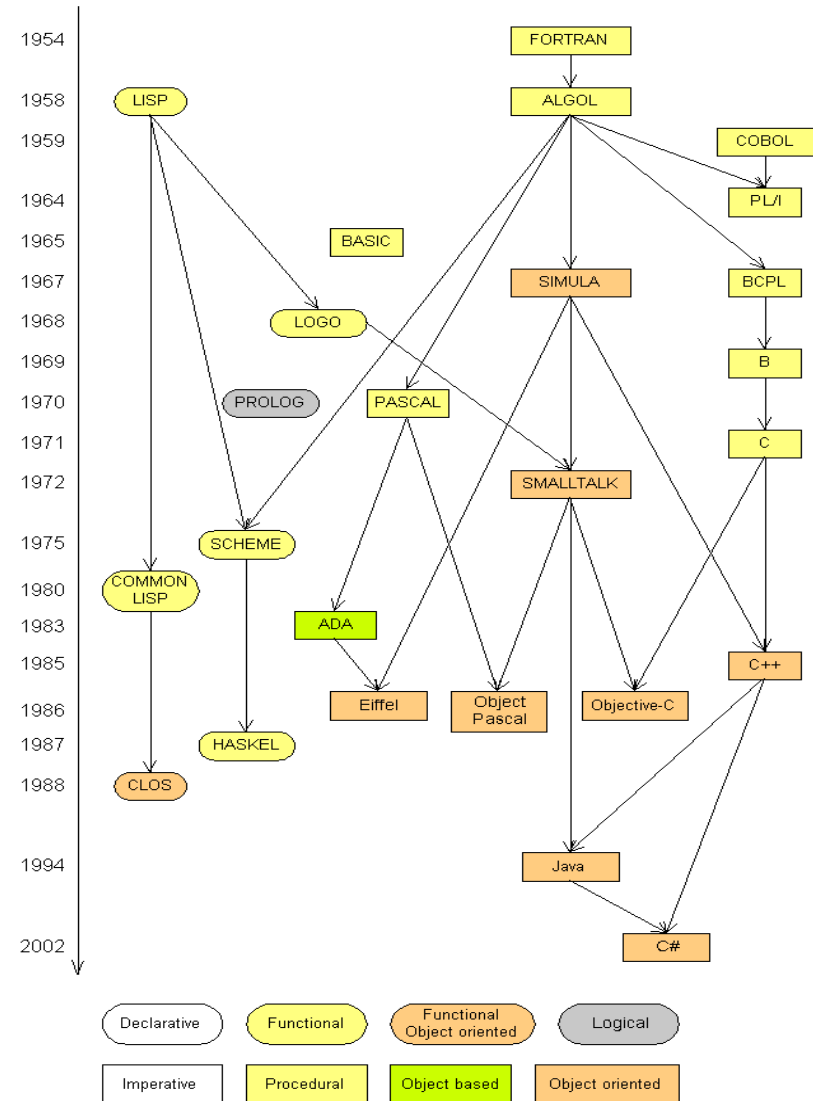


# Programming Languages

- There are hundreds of high level computer languages. Java, C++, C, Basic, Fortran, Cobol, Lisp, Perl, Prolog, Eiffel, Python
- The capabilities of the languages vary widely, but they all need a way to do
  - declarative statements
  - conditional statements
  - iterative or repetitive statements
- A **compiler** is a program that converts commands in high level languages to machine language instructions

# Programming languages

- Some influential ones:
  - FORTRAN
    - science / engineering
  - COBOL
    - business data
  - LISP
    - logic and AI
  - BASIC
    - a simple language





# Some modern languages

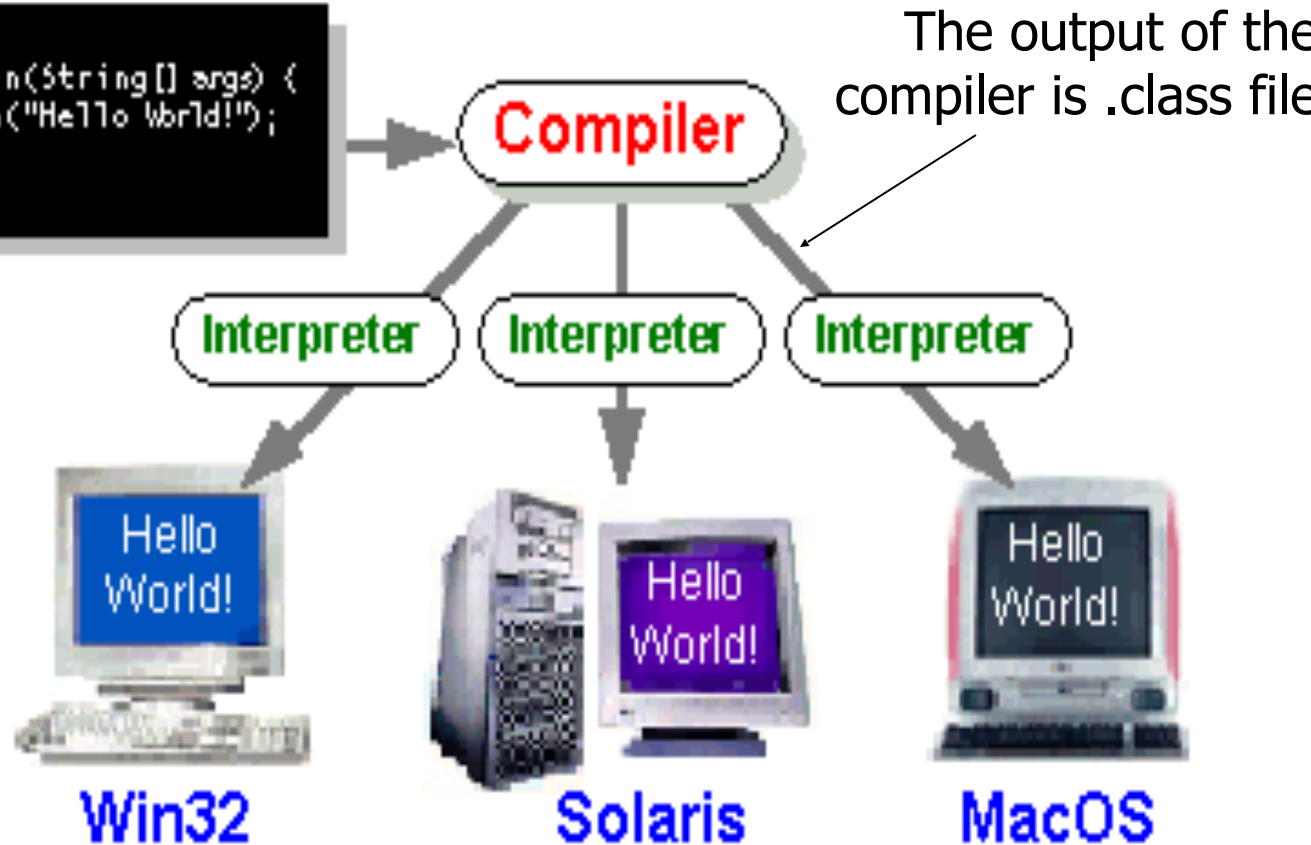
- procedural languages: programs are a series of commands
  - **Pascal** (1970): designed for education
  - **C** (1972): low-level operating systems and device drivers
- functional programming: functions map inputs to outputs
  - **Lisp** (1958) / **Scheme** (1975), **ML** (1973), **Haskell** (1990)
- object-oriented languages: programs use interacting "objects"
  - **Smalltalk** (1980): first major object-oriented language
  - **C++** (1985): "object-oriented" improvements to C
    - successful in industry; used to build major OSes such as Windows
  - **Java** (1995): designed for embedded systems, web apps/servers
    - Runs on many platforms (Windows, Mac, Linux, cell phones...)
    - The language taught in this textbook

# A Picture is Worth...

## Java Program

```
class HelloWorldApp {  
    public static void main(String[] args) {  
        System.out.println("Hello World!");  
    }  
}
```

HelloWorldApp.java



The Interpreter's are sometimes referred to as the Java Virtual Machines



# Compile/run a program

1. Write it.

- **code** or **source code**: The set of instructions in a program.

2. Compile it.

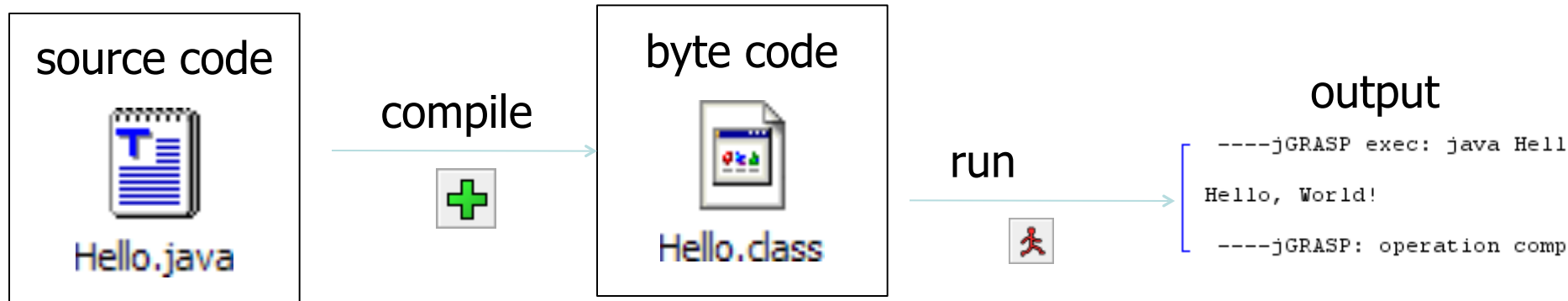
- **compile**: Translate a program from one language to another.
- **byte code**: The Java compiler converts your code into a format named byte code that runs on many computer types.
- **interpreter**: A converts one instruction or line of code from one language to another and then executes that instruction

When java programs are run the bytecode produced by the compiler is fed to an interpreter that converts it to machine code for a particular CPU.

# Compile/run a program

## 3. Run (execute) it.

- **output:** The messages printed to the user by a program.





# References

- 1) [CPJava Website](#)
- 2) [CPJava Google Classroom](#)
- 3) [CPJava repl.it Classroom](#)
- 4) [Runestone CSAwesome BUSHSCHOOL\\_CPJAVA Course](#)
- 5) Building Java Programs: A Back to Basics Approach by Stuart Reges and Marty Stepp

# What's next for today?

- Split into 2 Teams (Breakout Rooms)
- Java Basics - Four-Fours ([Access from GC linked here](#))
  1. Setup Part 2 - Mark and I will support the Setup
    - A. Follow instructions carefully. It will be tedious to undo!
    - B. Install Github, Repl.IT, Runestone csawesome, Codingbat
  2. Complete Runestone assignment
  3. Start Four-Fours assignment
- Debrief as a class starting at 9:55