Algorithmic Design I

Forest Agostinelli
University of South Carolina

Outline

- Logistics
- Background
- JVM
- Hello World

Faculty and Staff

- Instructor
  - Forest Agostinelli
    - B.S. in Electrical and Computer Engineering from Ohio State
    - M.S. in Computer Science from the University of Michigan
    - Ph.D. in Computer Science from the University of California, Irvine
  - Area of research
    - Explainable artificial intelligence
    - Applications of deep learning and reinforcement learning to the sciences
- Teaching Assistant
  - TBD

Class Time and Website

- Lecture
  - Mon/Wed 1:10pm-2pm, Swearingen Eng Center, 2A27
- Lab Q&A
  - H01 – Tue/Thur 8:00am-9:40am, Carolina Coliseum, 3012
  - H02 – Mon/Wed 8:00am-9:40am, Carolina Coliseum, 3012
  - Starts next week
- Will create a course website with lecture slides, assignments, and other course material
- Office hours
  - After class
- While attendance is not mandatory, you will most likely benefit from attending

Piazza

- For all questions regarding assignments and course concepts, please use Piazza
  - You can also post anonymously, if you wish
- To ensure everyone turns in their own work, do not post code or solutions (or partially completed solutions) to homework assignments on Piazza.

Textbooks

- Java: An Introduction to Problem Solving and Programming
- All relevant information should be contained in the lecture slides, but you can use this book to supplement your learning

Labs and Homeworks

- There will be 9 labs and 8-9 homeworks
- Submissions will be through Dropbox
  - All deadlines are strict! No late submissions will be accepted!
- **Labs**
  - Smaller coding assignments accompanied by written reports designed to measure conceptual knowledge
- **Homeworks**
  - Longer coding assignments designed to measure practical ability
- A large percentage of grade is determined by the code's functionality
  - Meaning that it must run in order to get credit!

Supplementary Lectures

- Dr. JJ Shepherd at UofSC
  - https://www.youtube.com/channel/UCDPP8U6GG0O1FiGdDFCnFyQ
- Note, this is to supplement your learning. Anything related to course logistics, grading, etc. will be given in this class

Academic Integrity

- All work turned in must be your own. Plagiarism of any kind, including from online sources, is strictly prohibited. All potential Honor Code violations will be reported to the Office of Academic Integrity. Honor Code violations of any kind (including plagiarism) on the homework assignments will result in a zero on that assignment. Furthermore, students who have plagiarized a homework assignment will not be able to drop their lowest grade. Honor Code violations of any kind (including plagiarism) on the midterm or final will result in failure of the course. You can familiarize yourself with the Honor Code here: http://www.sc.edu/policies/ppm/staf625.pdf.
  - If your homework is copied from online sources or another student, you will receive a zero.

Academic Integrity Examples

- Discussing concepts with fellow students
  - Okay
- Copying answers of fellow students
  - Not okay
- Searching the internet for conceptual material
  - Okay
- Copying code from the internet
  - Not okay
- Copying answers to the written homework from the internet
  - Not okay
- If there are *any* questions, please ask!

ChatGPT

- I view ChatGPT as a calculator
  - If you can be a great programmer with ChatGPT, then that is fine by me
- However, we are currently learning the process behind the calculator
- Therefore, I expect all work turned in to be your own

Grading

- Homeworks – 60%
- Laboratory Assignments – 10%
- Laboratory Reports – 10%
- Final – 20%

- Grades
  - A [90 – 100]
  - B+ [85 – 90)
  - B [80 – 85)
  - C+ [75 – 80)
  - C [70 – 75)
  - D+ [65 – 70)
  - D [60 – 65)
  - F [0 – 60)

Honors

- Connections to artificial intelligence (AI) algorithms will be made throughout the class
- I like to think much of AI as meta-algorithms
  - That is, algorithms that write algorithms

Special Thanks

- JJ Shepherd
- Neema Kanapala

Outline

- Logistics
- Background
- JVM
- Hello World

Algorithmic Design

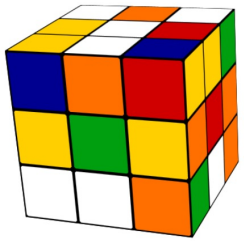
- What is an algorithm?
- Why do we care about algorithms?

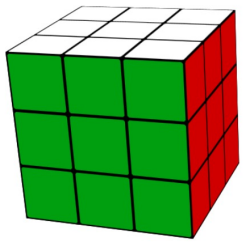
Algorithmic Design

- What is an algorithm?
  - A sequence of instructions
- Why do we care about algorithms?
  - We can solve many problems with a sequence of instructions

Evaluating Algorithms

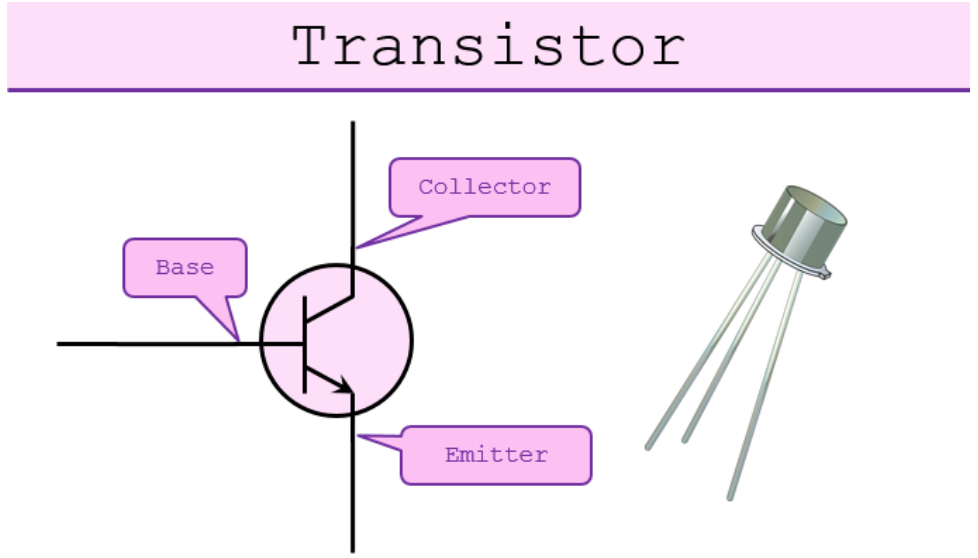
- Imagine developing an algorithm for solving the Rubik's cube.
- What are aspects of the algorithm one may consider when evaluating it?





Implementing Algorithms: Transistors

- We can create digital (binary 0/1) circuits using transistors
- Many possibilities for computation and for storing data in memory

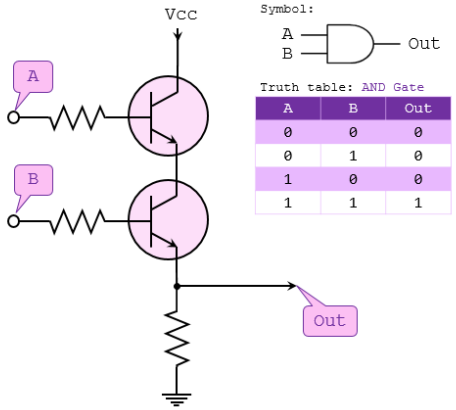


https://www.101computing.net/creating-logic-gates-using-transistors/

Implementing Algorithms: Logic Gates

- From transistors, we can implement basic logical operations

AND Gate

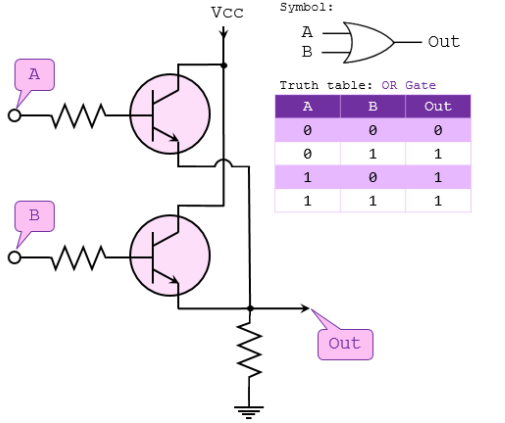


Symbol:

Truth table: AND Gate

A	B	Out
0	0	0
0	1	0
1	0	0
1	1	1

OR Gate

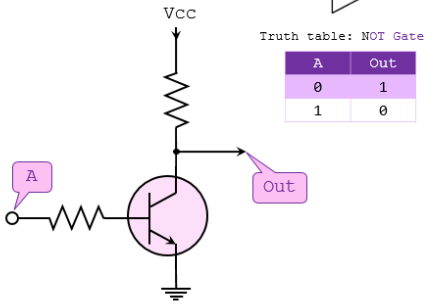


Symbol:

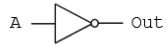
Truth table: OR Gate

A	B	Out
0	0	0
0	1	1
1	0	1
1	1	1

NOT Gate



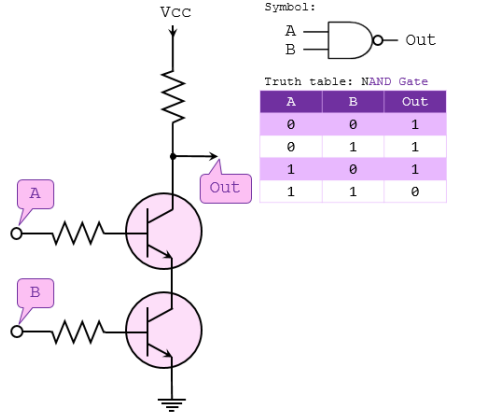
Symbol:



Truth table: NOT Gate

A	Out
0	1
1	0

NAND Gate



Symbol:

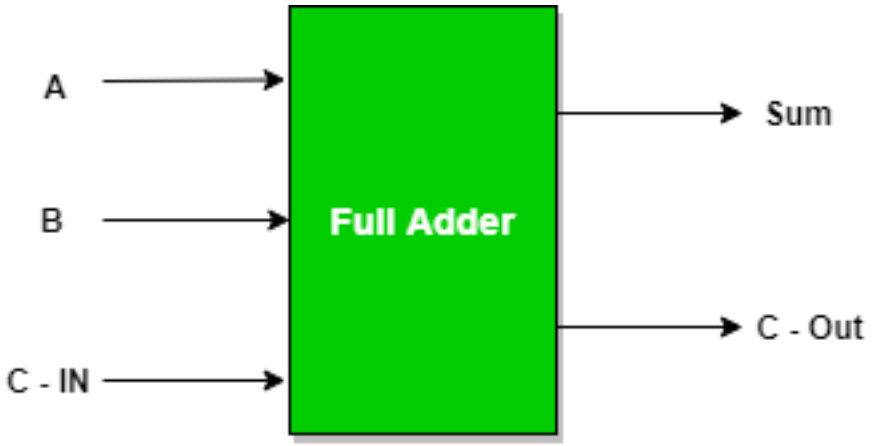
Truth table: NAND Gate

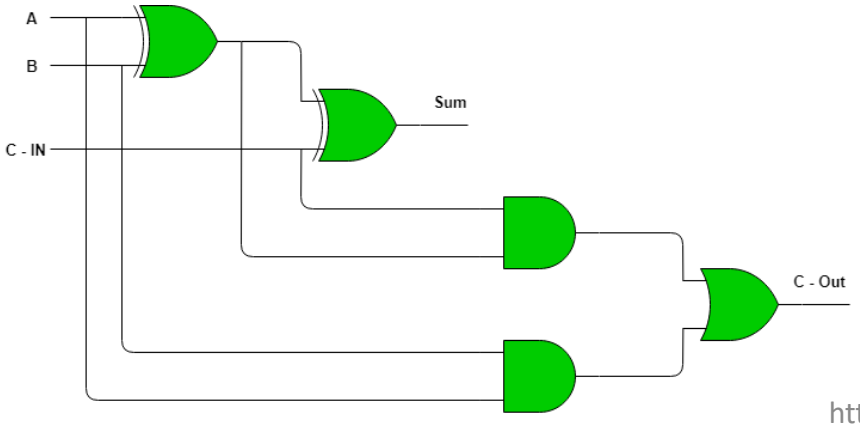
A	B	Out
0	0	1
0	1	1
1	0	1
1	1	0

https://www.101computing.net/creating-logic-gates-using-transistors/

Implementing Algorithms: Computation

- From logic gates, we can do computation
  - From simple adders to CPUs, which have billions of transistors



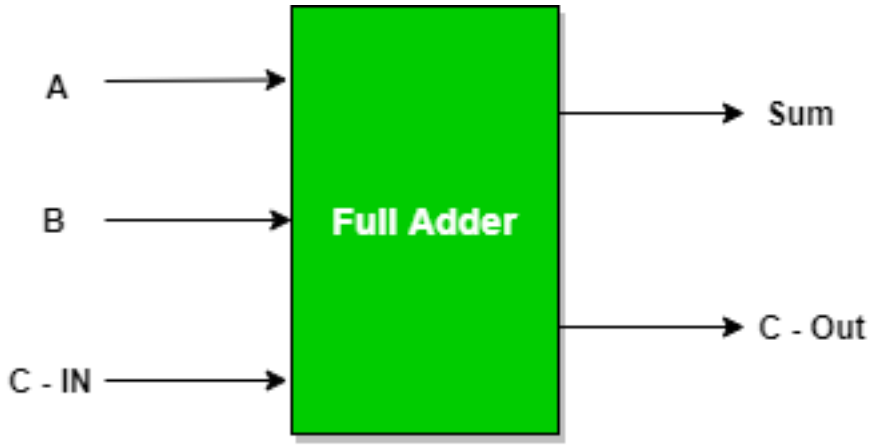


Inputs			Outputs	
A	B	C – IN	Sum	C - Out
0	0	0	0	0
0	0	1	1	0
0	1	0	1	0
0	1	1	0	1
1	0	0	1	0
1	0	1	0	1
1	1	0	0	1
1	1	1	1	1

https://www.geeksforgeeks.org/full-adder-in-digital-logic/

Implementing Algorithms: Machine Code

- Given a CPU, we would like to be able to tell it what computations to do without having to manually set voltages on the hardware
- We can instead tell the machine what to do using some software where we can communicate a series of 1s and 0s to the hardware



https://www.geeksforgeeks.org/full-adder-in-digital-logic/

Implementing Algorithms: Assembly

- Writing 1s and 0s is tedious
- Higher level languages, such as assembly, are able to abstract this process using commands such as “add” and “move”
- This is then “assembled” into machine code

Assembly vs. machine code

Machine code bytes	Assembly language statements
	`foo:`
`B8 22 11 00 FF`	`movl $0xFF001122, %eax`
`01 CA`	`addl %ecx, %edx`
`31 F6`	`xorl %esi, %esi`
`53`	`pushl %ebx`
`8B 5C 24 04`	`movl 4(%esp), %ebx`
`8D 34 48`	`leal (%eax,%ecx,2), %esi`
`39 C3`	`cmpl %eax, %ebx`
`72 EB`	`jnae foo`
`C3`	`retl`

Instruction stream

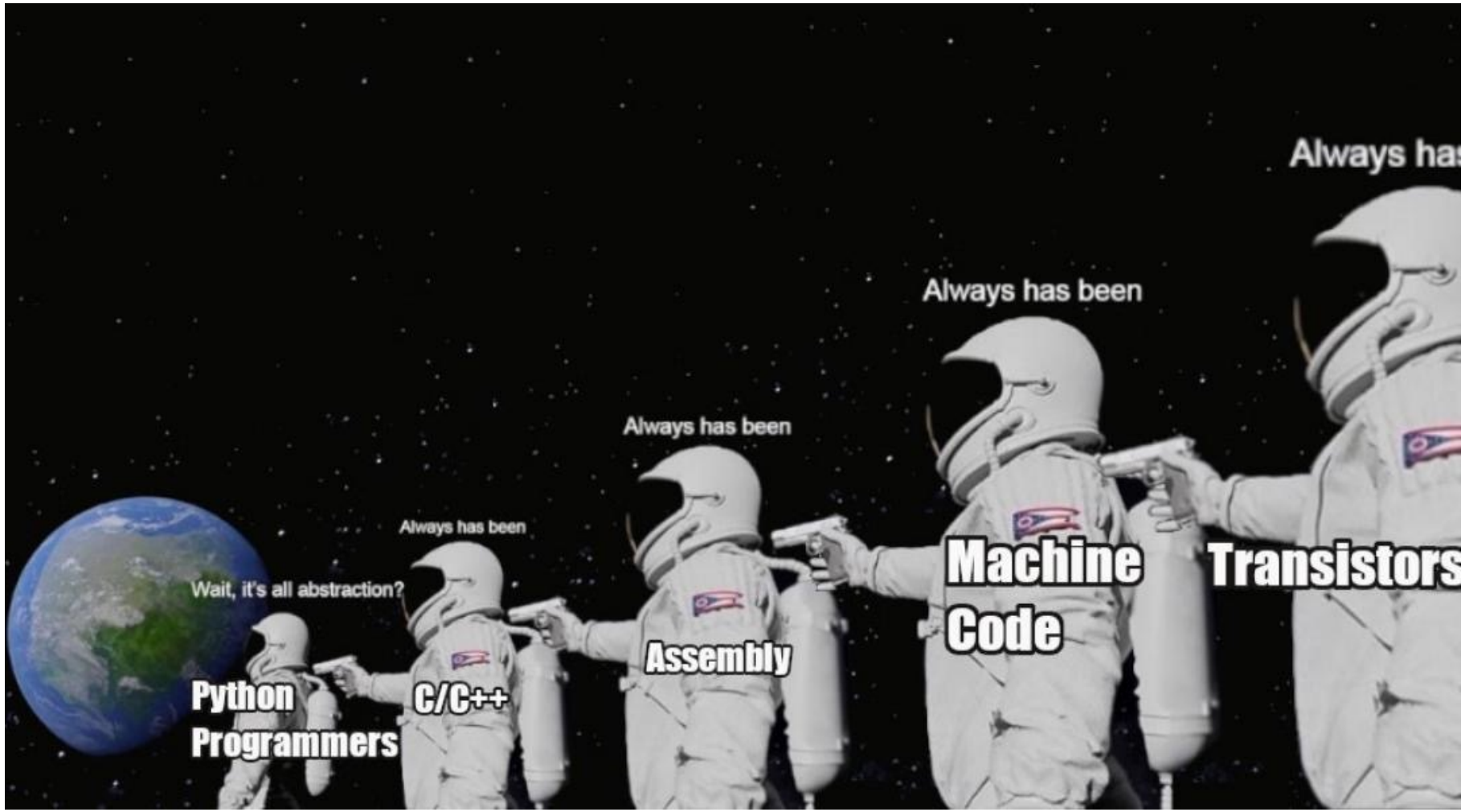
```
B8 22 11 00 FF 01 CA 31 F6 53 8B 5C 24
04 8D 34 48 39 C3 72 EB C3
```

https://www.nayuki.io/res/a-fundamental-introduction-to-x86-assembly-programming/machine-code.svg

Implementing Algorithms: Higher-level Languages

- Writing assembly code is tedious
- We can use higher-level programming languages, such as C++, Java, Python, etc.
- These high-level languages take care of many important functions in a few lines of code, a single line of code, or even without any explicit code at all
  - Memory allocation
  - Control flow
  - Arithmetic
  - Memory cleanup
  - Etc.

Implementing Algorithms



Found on Reddit

Outline

- Logistics
- Background
- JVM
- Hello World

Terminology: Compiled vs Interpreted Languages

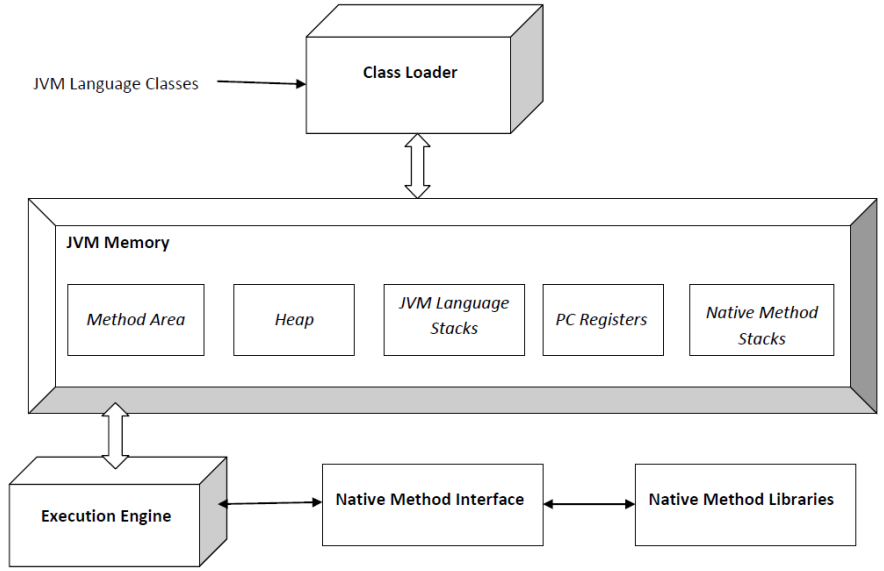
- **Compiled languages**
  - Convert code to assembly or machine code through a process called compilation
  - Examples: C++, Java
- **Interpreted Languages**
  - Call precompiled code based on the code written in the high-level language
  - Examples: Python, Perl

Hardware and Code Execution

- When running a program, the machine code that it eventually produces must be specific to the hardware on which it is running
- Furthermore, the hardware determines what is possible in that language
- Therefore, code may need to be re-compiled or even re-written based on the computer architecture

Java Virtual Machine (JVM)

- To address this issue, Java is compiled to Java bytecode and then runs its code within its own virtual machine
- Therefore, compiled code and its behavior can be consistent across architectures given an architecture-specific translation to machine code



https://en.wikipedia.org/wiki/Java_virtual_machine

Outline

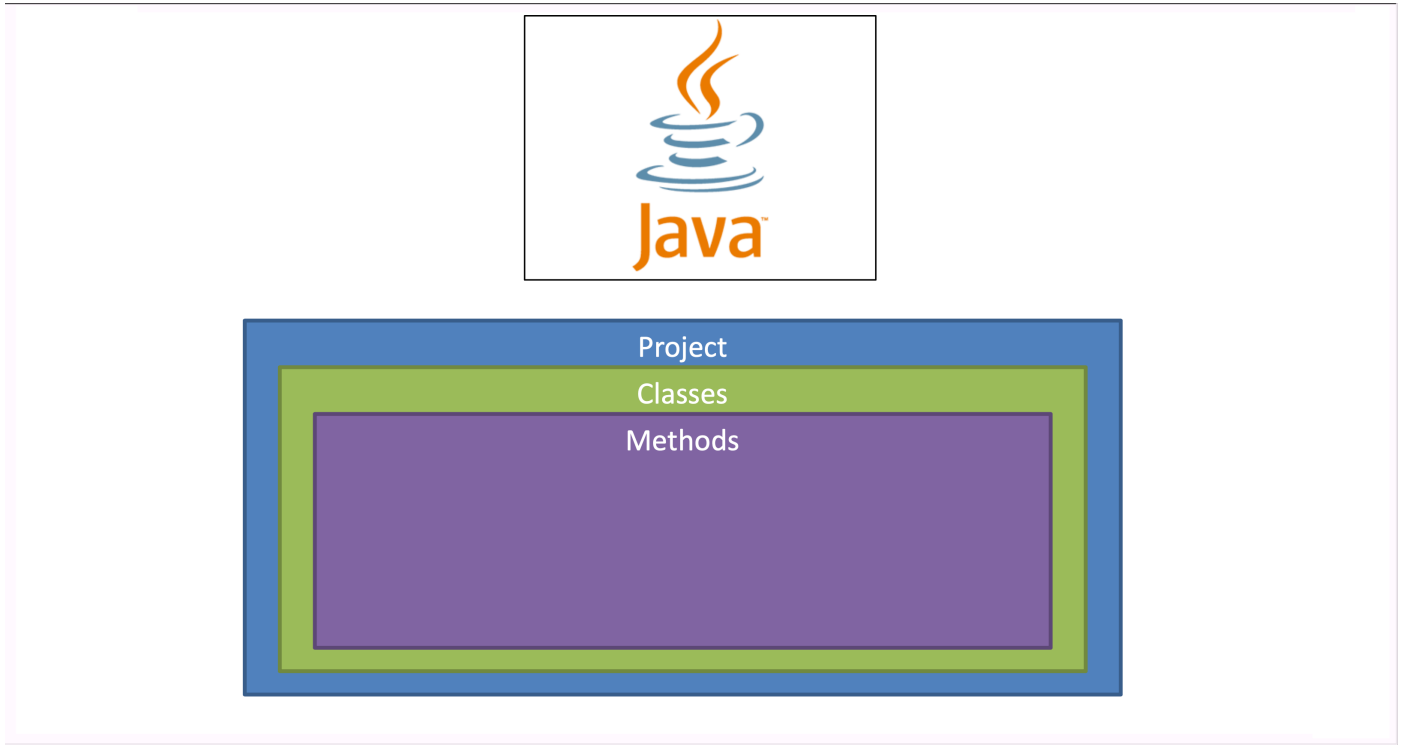
- Logistics
- Algorithms
- Java Virtual Machine (JVM)
- Hello World

Integrated Development Environment (IDE)

- IDEs are convenient software applications that assist in software development
  - Catch syntax errors
  - Organize code
  - Convenient compilation and execution
- While you can really use anything, this class will use Eclipse
- Installing the Java Eclipse IDE
  - https://www.eclipse.org/downloads/packages/installer

Java Organization

- A project is composed of classes
- A class is composed of methods
- Source code are files with the .JAVA extension
- The filename must match the name of the class
- Source code is compiled to the intermediate bytecode which is then run on the java virtual machine

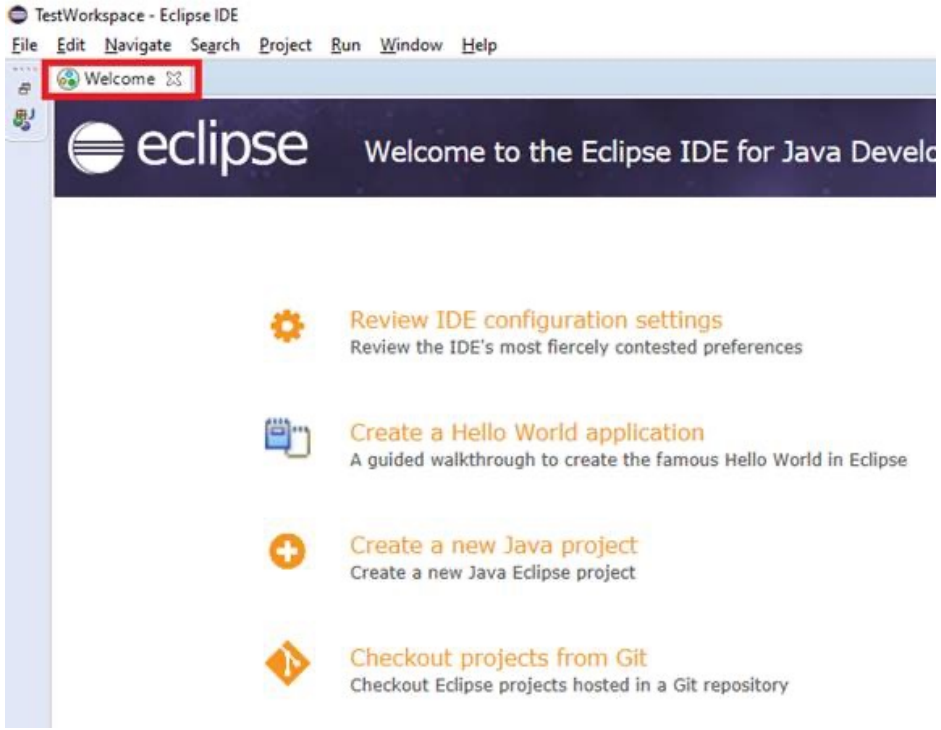


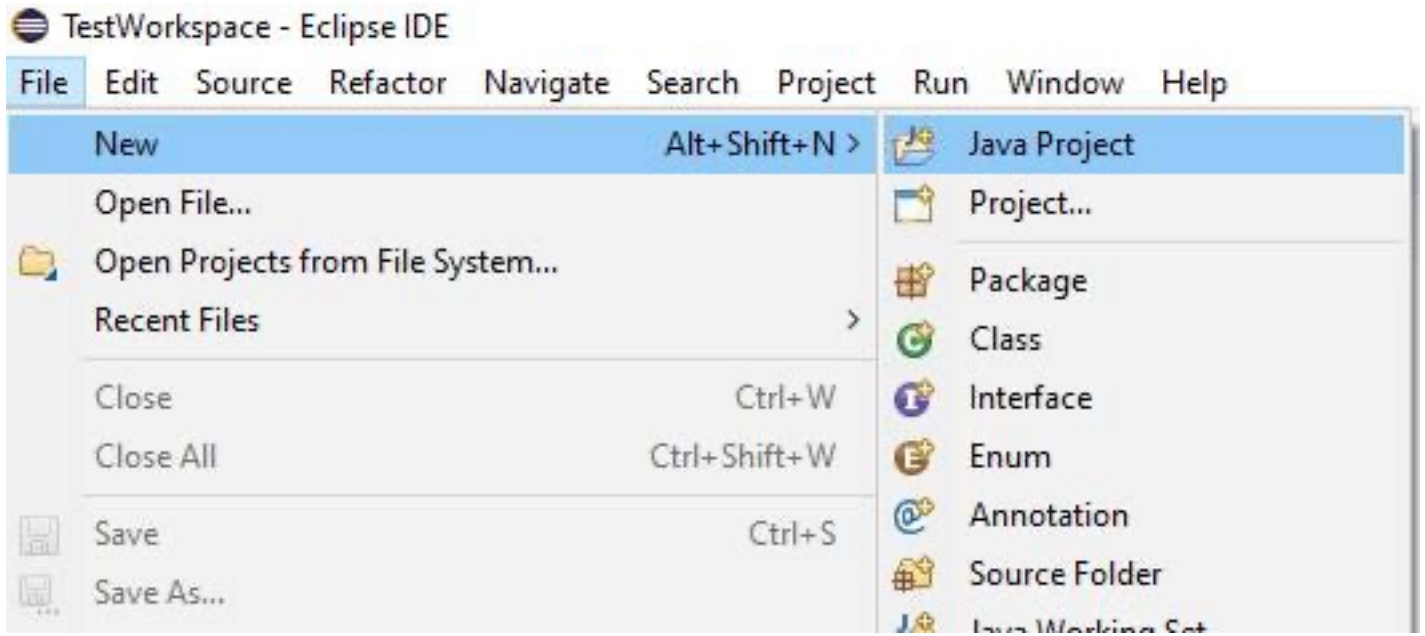
Terminology: General

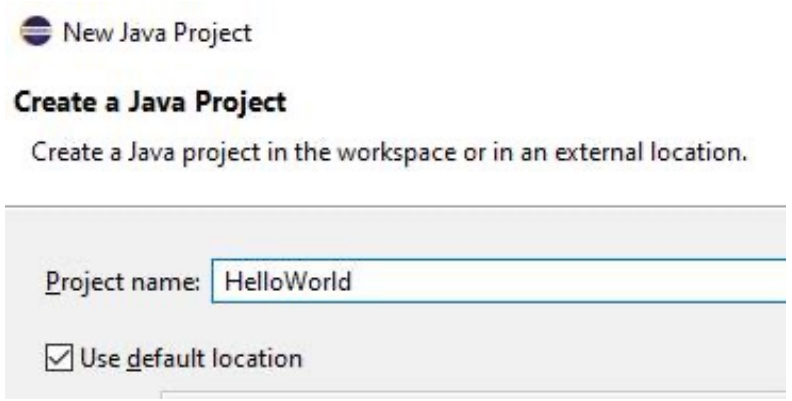
- Running (Executing) – Is when the computer is following the instructions in a program
- Statement – An instruction to the computer. Most end with a semicolon ";"
  - int i;
  - double j;
- Syntax – The grammar rules for a programming language
- Comments – Code ignored by the compiler that is generally used to explain the code.
  - Single line comments use the "//"
  - Multiline comments use "/* */"

- Arguments – Information found inside of parenthesis "()" that provide information for methods or other statements
  - if(<<ARGUMENT>>)
  - System.out.println(<<ARGUMENT>>)
- Bug – an error in a program
- Debugging – the process of removing errors
- There are 3 major classes of errors
  - Syntax
  - Runtime
  - Logic

Creating Your First Project

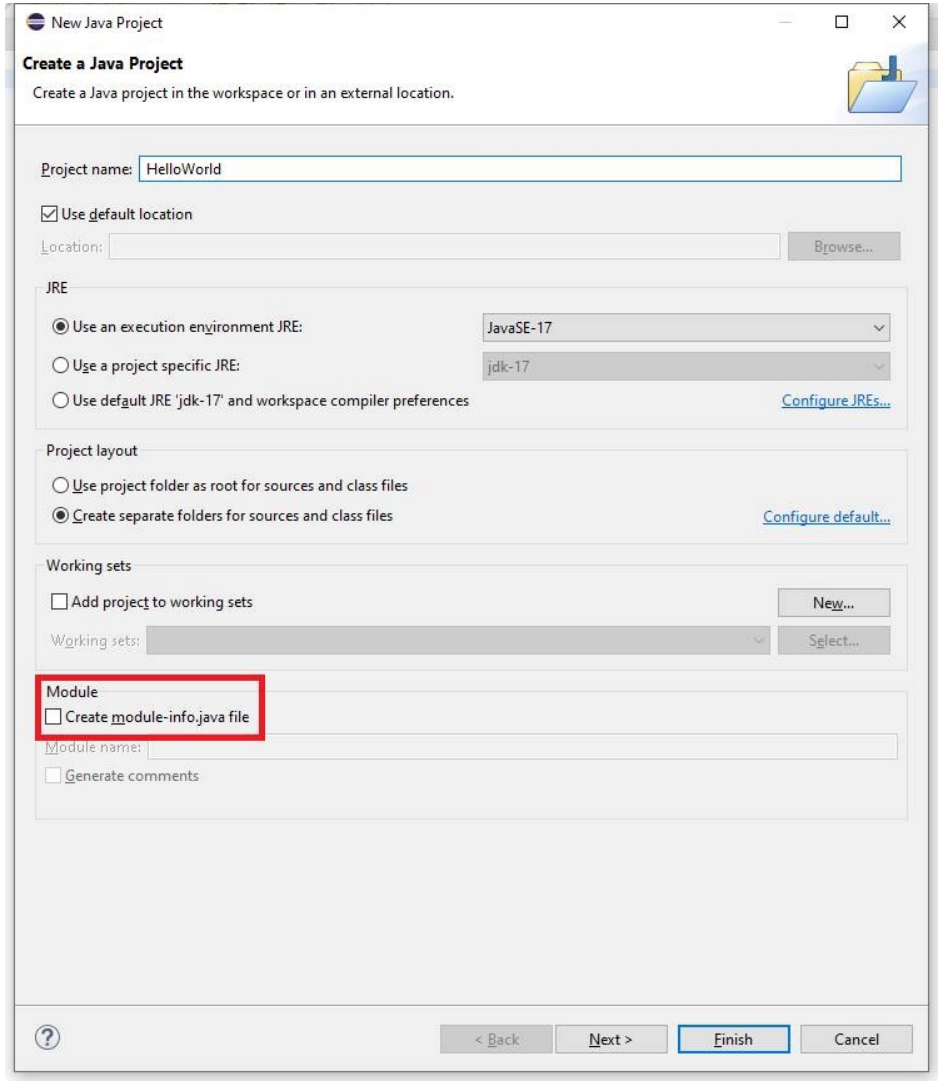


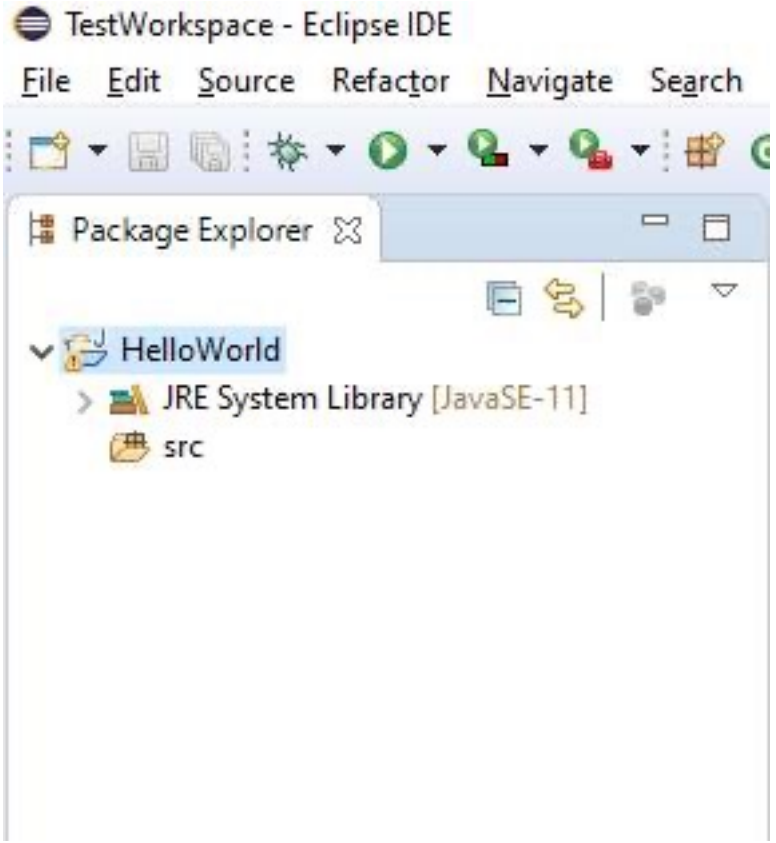




Creating Your First Project

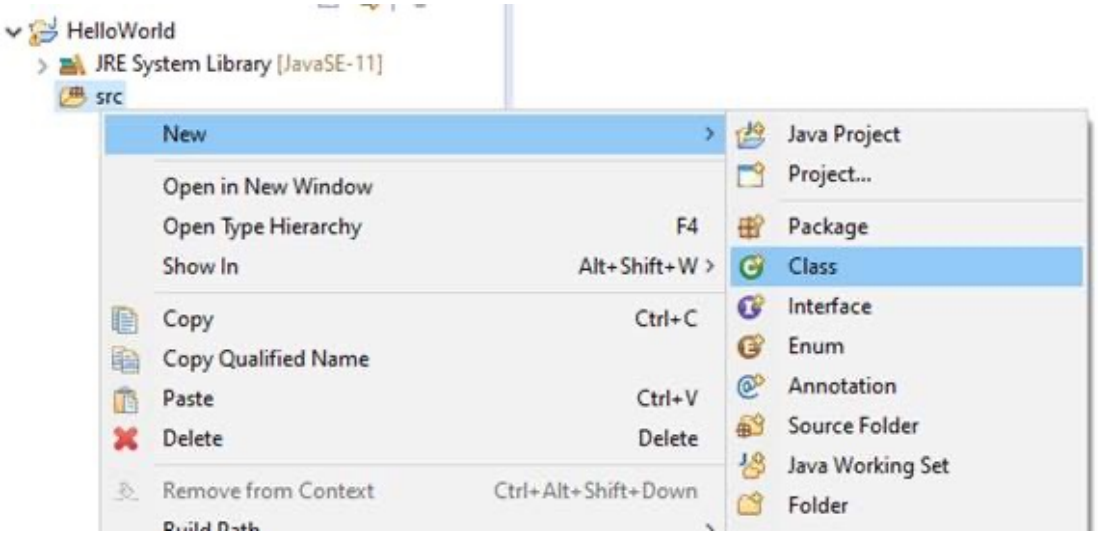
- Uncheck “Create module-info.java” if checked

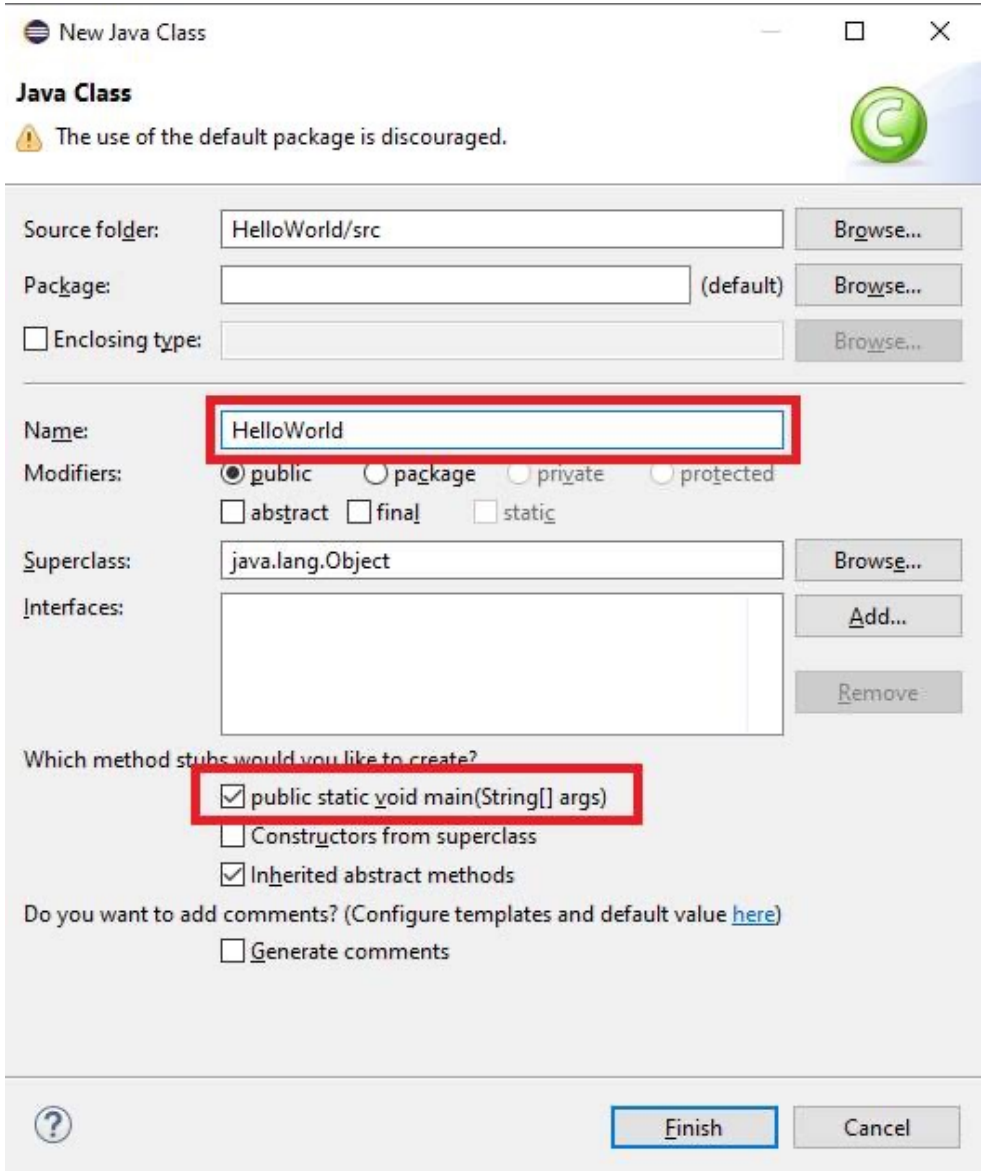




Creating Your First Class

- src - source



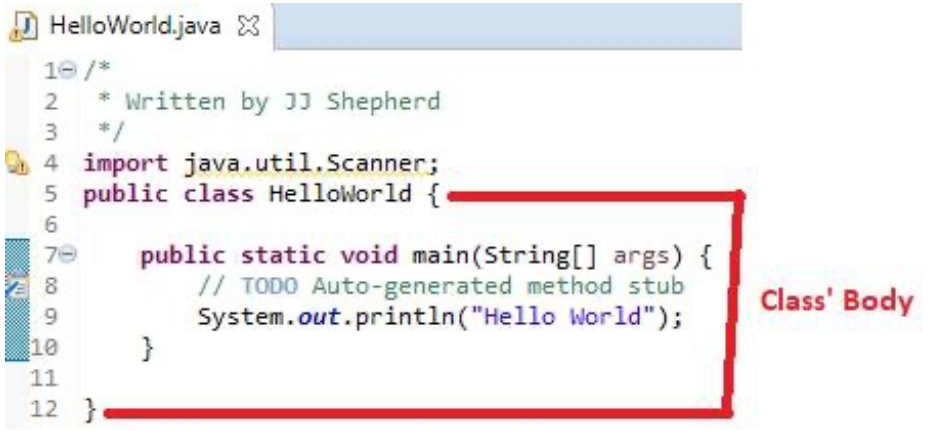


Creating Your First Method

- HelloWorld is the class
- main is a method in the class
  - This is the method that gets executed

HelloWorld.java

```
1  /*
2   * Written by JJ Shepherd
3   */
4  public class HelloWorld {
5
6      public static void main(String[] args) {
7          // TODO Auto-generated method stub
8
9      }
10
11 }
12
```



Creating Your First Method

```
public static void main(String[] args) {
    // TODO Auto-generated method stub
    System.out.println("Hello World");
    System.out.println("What is your name?");
    Scanner keyboard = new Scanner(System.in);
    String name = keyboard.nextLine();
    System.out.println("Greetings! "+name);

    System.out.println("How many cats do you have?");
    int numberOfCats = keyboard.nextInt();
    System.out.println("How does one live with "+numberOfCats+" cats?");
}
```


```
Hello World
What is your name?
JJ
Greetings! JJ
How many cats do you have?
3
How does one live with 3 cats?
```

Terminology: Syntax Errors

- Syntax Errors– Grammatical mistakes in a program
  - Very common at first
  - These prevent a program from compiling and running
  - Common
    - Missing a Semicolon at the end of a statement
    - Using the wrong, misspelled, repeated, or incorrectly capitalized identifier
    - Mismatched curly braces “{}”, parenthesis “()”, single quotes “‘’”, double quotes ““”, etc.

```
int i //Missing a semicolon
double j = 0.0;
J = 1.0;//Wrong identifier
System.out.println(i;//Missing parenthesis
```

Terminology: Runtime Errors

- Runtime Errors – Errors detected when the program is running but not during compilation
  - The code will compile but crashes at some point
  - When this happens the computer detects the error and terminates the program
  - Common
    - Divide by 0
    - Calling a method from a NULL object
    - An index going outside the bounds of an Array

```
double j = 1.0 / 0.0;//Divide by 0
Scanner keyboard;//This has not been constructed so it is NULL
keyboard.hasNextLine();//Calling method from NULL object
int[] a = {5,4,3,2,1};//An array
a[5] = 2;//Index 5 is out of bounds
```

Terminology: Logic Errors

- Logic Errors – despite the program compiling and running, it produces incorrect results
  - Arguably the hardest to fix
  - Common:
    - Order of operations error
    - Round off mistakes
    - Incorrectly using types or methods

```
double f = 72.0;//72 degrees fahrenheit
double c = f - 32.0 * 5.0/9.0;//Order of operations error
double c2 = (f-32.0)*(5/9);// 5/9 = 0 so it will always be 0
double c3 = (f-32.0)*(5.0/9.0);//Correct
```

Getting Involved in Research

- Lots of exciting research going on at USC
  - https://www.sc.edu/study/colleges_schools/engineering_and_computing/departments/computer_science_and_engineering/our_people/index.php
- AI Institute
  - https://aiisc.ai
- My own research: explainability, deep learning, reinforcement learning, and planning