COS Crash Course: Week 2

conditionals and loops

name game

Yacoub Kahkajian *he/him*

Graduating in 2026

A.B. Computer Science

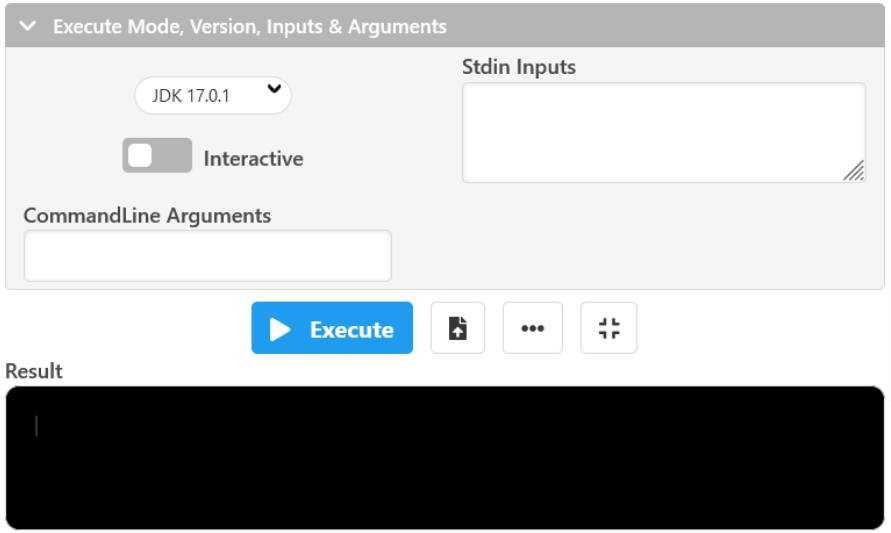
yacoub.xyz

jdoodle.com/online-java-compiler

Feel free to play around with it while I yap.

```java
public class MyClass {
    public static void main(String args[]) {
      int x=10;
      int y=25;
      int z=x+y;

      System.out.println("Sum of x+y = " + z);
    }
}
```

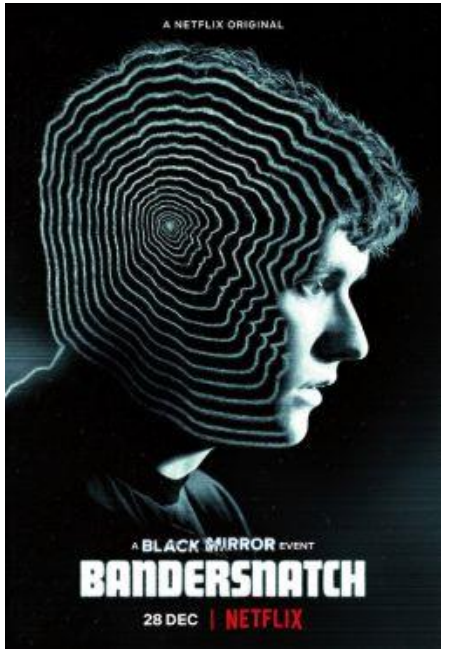


lecture

conditionals
loops
debugging

conditional love

- In the past lesson, we wrote programs that only had a single set of instructions. They get an input, then run a static series of commands.
- Now, we're getting spicy with conditionals: lines of code that only run if a certain condition is met.
- Your previous choices can affect the final outcome of the program! Choose your own adventure!



if

If statements use boolean expressions (remember the boolean math from last time?) and execute the code in brackets if the expression is true.

```
int x = Integer.parseInt(args[0]);
int y = 4;
if (x > y) {
  y = y + 1;
}
System.out.println(y);
```

Command line arguments

“2”

```
> 4
```

Command line arguments

“7”

```
> 5
```

else

Else statements run if the expression if the if statement above it ended up being false.

```
int x = Integer.parseInt(args[0]);
if (x < 0) {
  System.out.print("Negative");
}
else System.out.print("Positive");
```

Command line arguments

"-2"

> Negative

Command line arguments

"16"

> Positive

```
int x = Integer.parseInt(args[0]);
if (x < 0) {
  System.out.print(“Negative”);
}
else System.out.print(“Positive”);
```

what would the program print if we removed the else and input a negative?

else if

If you want to have a whole series of connected if statements, use else if, which gives you more options for potential conditions and outputs.

```
int x = String.charAt(0);
if (x == ‘a’) System.out.print(“apple”);
else if (x == ‘b’) System.out.print(“banana”);
else if (x == ‘c’) System.out.print(“cherry”);
else System.out.print(“Invalid char!”);
```

If we used an if statement instead of an else if, this program would have printed a warning for all inputs except for ‘c’.

Try tracing the code to figure out why!

lecture

conditionals
loops
debugging

running in circles

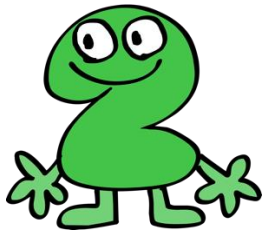
- By using loops, we can repeat certain parts of our code until a particular condition is met.
- This is great for a lot of things, including...
  - Looking at every character in a string.
  - Repeating calculations for a set number of times.
  - Searching through arrays (more on those next week!)
  - And much, much more!

while loop

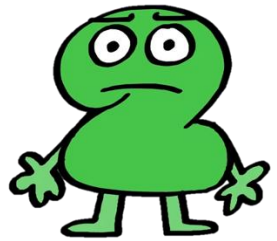
While loops run code for an indefinite number of times until a certain boolean expression returns true.

```
int n = Integer.parseInt(args[0]);
int i = 1;
while (i <= n) {
    i = i * 2;
    System.out.println(i);
}
```

What does this code do?



Is there a case where this code would print out an incorrect number of powers?



for loop

- For loops run code for a set number of times. This number can be set by another variable or hard-corded.

```
for (int i = 0; i < 4; i++)
```

- The initialization statement creates a new variable with a value that tracks how many times the code has already looped.
- The boolean expression tells the code under what condition it should break out of the loop.
- The increment statement tells the initialized variable to increase by a certain amount after every loop.

```
int n = Integer.parseInt(args[0]);
int i = 1;
while (i <= n) {
    i = i * 2;
    System.out.println(i);
}
```

how would you rewrite this with a for loop?

nesting

- Putting a loop within a loop or a conditional within a conditional.
- Nesting conditionals is… okay? Overuse makes code hard to trace.
- Nesting loops is okay for solving problems where you need to check all combinations of values using brute force, but you'll learn in DSA how you can usually find more efficient ways to solve problems.

```
String x = args[0];
for (int i = 0; i < x.length(); i++) {
  for (int j = i + 1; j < x.length(); j++) {
    if (x.charAt(i) == x.charAt(j)) {
      return false;
    }
  }
}
return true;
```

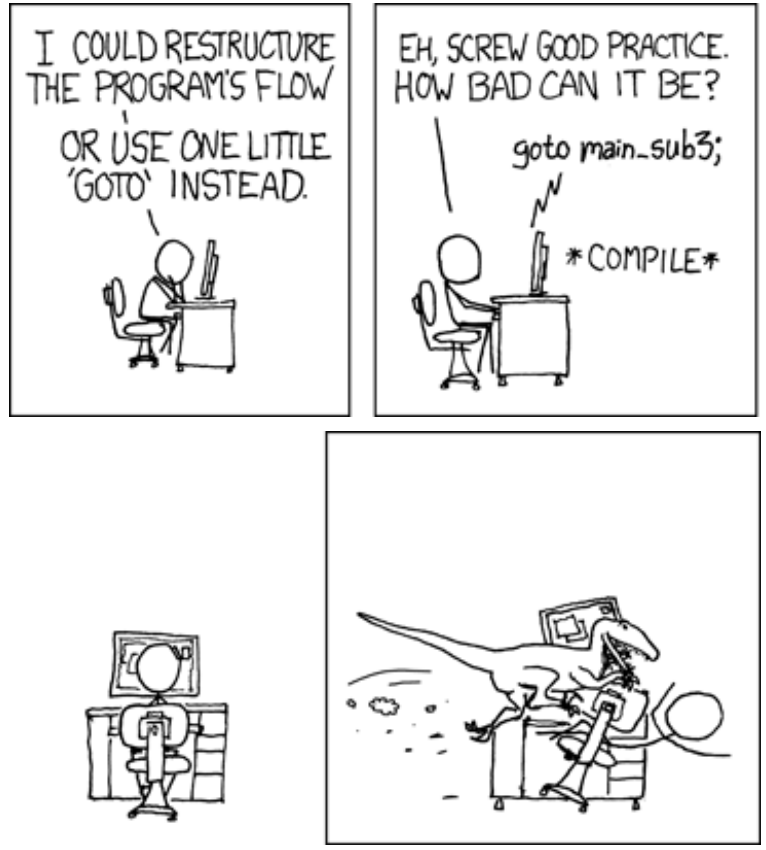
what does this code do? trace through it!

lecture

conditionals
loops
debugging

uh oh

- As you may expect, debugging programs that use loops and conditionals is much more complex since there's more things that can go wrong.
- It used to be *waaaay* worse, though. Goto was the original flow control method of choice for programmers, which executed a specific line number.
- Compared to that, loops and conditionals provide much more structure.



https://xkcd.com/292/

certified debugging tips*

- **Read terminal errors**: It may sound like a lot of technical goobly-gook but it provides immensely helpful information about syntax errors and the lines where they occur.
- **Trace through the program**: As we did already. Think like the computer, going through the code line-by-line and seeing where your code may have tripped up.
- **Liberally use print statements**: These are amazingly helpful for keeping track of values at different points within loops or after conditional statements.

exercises

fizzbuzz

An absolute classic! Apparently, there's a British game called fizzbuzz where two kids count while replacing numbers divisible by 3 "fizz," numbers divisible by 5 with "buzz," and numbers divisible by both with "fizzbuzz."

Sounds super boring! But super fun to code!

fibonacci

Slightly less of a classic but still pretty prevalent.

Given an integer, n, write a program that generates n Fibonacci numbers (0, 1, 1, 2, 3, 5, 8, 13, 21...).

fibonacci

Slightly less of a classic but still pretty prevalent.

Given an integer, n, write a program that generates n Fibonacci numbers (0, 1, 1, 2, 3, 5, 8, 13, 21...).