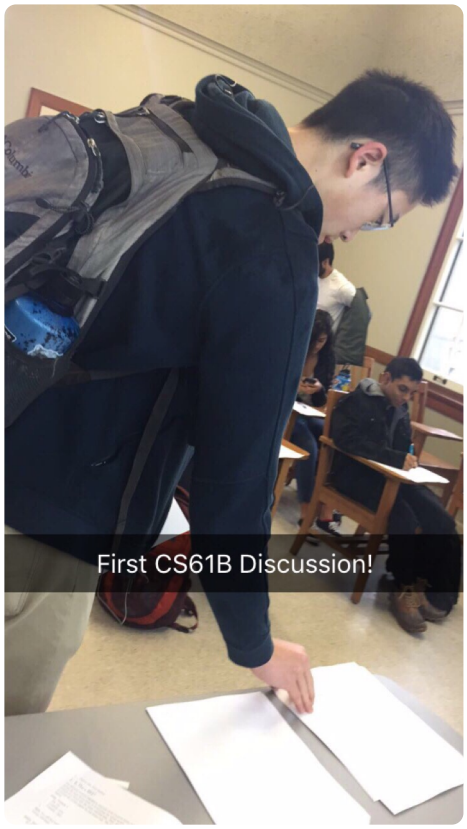
Shubham Gupta

Disc#114

Tuesdays 10-11 am

Wheeler 224



CS 61B Spring 2019 | Data Structures Josh Hug

Discussion 1: Introduction to Java

1 Our First Java Program

Below is our first Java program of the semester. Next to each line, write out what you think the code will do when run. *This exercise is adapted from Head First Java.*

```
int size = 27;
String name = "Fido";
Dog myDog = new Dog(name, size);
int x = size - 5;
if (x < 15) {
    myDog.bark(8);
}

while (x > 3) {
    x -= 1;
    myDog.play();
}

int[] numList = {2, 4, 6, 8};
System.out.print("Hello ");
System.out.println("Dog: " + name);

System.out.println(numList[1]);
if (numList[3] == 8) {
    System.out.println("potato");
}
```

type var

size 27

name "Fido"

myDog

Dog: name → "Fido", size 27

x 22

x += 1

x = x + 1

x = x - 1

19 times!

array

numList → 0 1 2 3: 2 4 6 8

Hello Dog: Fido

4

potato

2 Mystery

This is a function (a.k.a. method). It takes an array of integers and an integer as arguments, and returns an integer.

```
public static int mystery(int[] inputArray, int k) {
    int x = inputArray[k];
    int answer = k;
    int index = k + 1;
    while (index < inputArray.length) {
        if (inputArray[index] < x) {
            x = inputArray[index];
            answer = index;
        }
        index = index + 1;
    }
    return answer;
}
```

Describe in English what `mystery` returns if `inputArray = [3, 0, 4, 6, 3]` and `k = 2`.

2 Mystery

This is a function (a.k.a. method). It takes an array of integers and an integer as arguments, and returns an integer.

access modifier ret type name of method args

```
public static int mystery(int[] inputArray, int k) {
    int x = inputArray[k];
    int answer = k;
    int index = k + 1;
    while (index < inputArray.length) {
        if (inputArray[index] < x) {
            x = inputArray[index];
            answer = index;
        }
        index = index + 1;
    }
    return answer;
}
```

5 6 < 4 3 < 4 4

inputArray → [3, 0, 4, 6, 3] (0 1 2 3 4)

k → 2

x → ~~4~~ 3

answer → ~~2~~ 4

index → ~~3~~ ~~4~~ 5

Describe in English what `mystery` returns if `inputArray = [3, 0, 4, 6, 3]` and `k = 2`.

- array.length = len(array)
- returns 4
- X updates inputArray[index]
  - → x keeps track of the smallest elem found so far!
- answer: index of the variable

∴ method is returning the index of the smallest variable after k.

1) all variables have a specified type

~~integer~~ ← int age = 5;

~~string~~ ← age = "Hello";

int, String, double, Char, float, bool...

2) whitespace / indentation does not matter

Python

while ... :

1 →

Java

while ( ) {

}

3) End every complete with a semicolon

System.out.println("Hello World");

↳ print

Extra: This is another function. It takes an array of integers and returns nothing.

```
public static void mystery2(int[] inputArray) {
    int index = 0;
    while (index < inputArray.length) {
        int targetIndex = mystery(inputArray, index);
        int temp = inputArray[targetIndex];
        inputArray[targetIndex] = inputArray[index];
        inputArray[index] = temp;
        index = index + 1;
    }
}
```

Describe what `mystery2` does if `inputArray = [3, 0, 4, 6, 3]`.

3 Writing Your First Program

Implement `fib` which takes in an integer `n` and returns the nth Fibonacci number.

The Fibonacci sequence is $0, 1, 1, 2, 3, 5, 8, 13, 21, \ldots$.

```
public static int fib(int n) {



}
```

Extra: Implement `fib` in 5 lines or fewer. Your answer must be efficient.

```
public static int fib2(int n, int k, int f0, int f1) {



}
```

3 Writing Your First Program

Implement `fib` which takes in an integer `n` and returns the *n*th Fibonacci number.

The Fibonacci sequence is $0, 1, 1, 2, 3, 5, 8, 13, 21, \ldots$.

```
public static int fib(int n) {
```

Iteration

```
int counter = 0;
int fo = 0;
int f1 = 1;

while (counter < n) {
    int temp = f1;
    f1 = f1 + f0;
    counter++;
}
```

int f1, f0 = 2, 3;

counter += 1;
counter = counter + 1
counter++;

0, 1, 1, 2, 3, 5, 8

4

Recursion

```
fib(int n) {
    if (n <= 1) {
        return n;
    } else {
        return fib(n-1) + fib(n-2);
    }
}
```

n = 4

fib(3) + fib(2)

n=3 n=2

fib(1) + fib(2) fib(0) + fib(1)

1 fib(0) fib(1) 0 1

0 1

= 4