Introduction to Java

Discussion 02

Announcements

1. Lab 1, Lab 2, and HW 0 due Friday 01/28 (all of these CANNOT be dropped)
2. HW 1 released Tuesday at noon, due next Tuesday 02/01
3. OH starts this week entirely online
4. Please complete the Pre-Semester Survey!

All About Your TA!

Hey, I'm Aniruth! I'm a second year EECS and Business major.

Discussion: Wednesday 9 AM

Lab: Thursday 9 AM

Office Hour: Monday 10 AM

Review

Anatomy of a Function

```
/** Print all primes up to and including LIMIT. */
private static void printPrimes(int limit) {
    for (int p = 2; p < = limit; p += 1) {
        if (isPrime(p)) {
            System.out.print(p + " ");
        }
    }
    System.out.println();
}
```

Comments

Keywords for the basic elements of the language (we will cover more later)

Type declarations - Java is statically typed so we have to tell the computer what type of value every variable holds and what every function returns

Variable and Function Names that allow us to refer to our stored values

Don't forget the brackets and semicolons!

Structure of a Class

```
public class CS61BStudent { // Class Declaration
    public int idNumber; // Instance Variables
    public int grade;
    public static String professor = “Hilfinger”; // Class (Static) Variables
    public CS61BStudent (int id) { // Constructor
        this.idNumber = id;
        this.grade = 100;
    }
    public void watchLecture() { // Instance Method
        ...
    }
    public static void updateGrades() { // Class (Static) Method
        ...
    }
}
```

Instantiating Classes

```
public class CS61BLauncher {
    public static void main(String[] args) {
        CS61BStudent studentOne; // Declare class
        studentOne = new CS61BStudent(32259); // Instantiate and assign class
        CS61BStudent studentTwo = new CS61BStudent(19234); // Both at once

        studentOne.watchLecture(); // Instance methods are called on instance

        CS61BStudent.updateGrades(); // Static methods can be
                                     // called by class OR instance
    }
}
```

Static vs. Instance

Static variables and functions belong to the whole class.
Example: Every 61B Student shares the same professor, and if the professor were to change it would change for everyone.

Instance variables and functions belong to each individual instance.
Example: Each 61B Student has their own ID number, and changing a student's ID number doesn't change anything for any other student.

CS 61B Spring 2022

Introduction to Java

Discussion 2: January 24, 2022

1 Old Town Code

Next to each line, write out in words what you think the code will do when it is run. Assume the `Singer` class exists and that the code below compiles. You can assume that the `sing` function in Singer returns a `String` and prints nothing.

```
1  int x = 7;
2  String chorus = "Thank u, next";
3  Singer queen = new Singer("Ariana");
4
5  while (x > 0) {
6      x -= 1;
7      queen.sing(chorus);
8  }
9
10 String[] phrases = {"love", "patience", "pain", "what does the fox say?"};
11
12 for (int i = 0; i < 3; i += 1) {
13     System.out.println("One taught me " + phrases[i]);
14 }
15
16 System.out.println(phrases[phrases.length - 1]);
```

Hint: For reference, here is an equivalent Python program.

```
1  x = 7
2  chorus = "Thank u, next"
3  queen = Singer("Ariana")
4
5  while (x > 0):
6      x -= 1
7      queen.sing(chorus)
8
9
10 phrases = ["love", "patience", "pain", "what does the fox say?"]
11
12 for i in range(3):
13     print("One taught me " + phrases[i])
14
15
16 print(phrases[len(phrases) - 1])
```

2 A Mystery

Below is a function (or method) called `mystery1`. It takes in two arguments and returns an integer, `answer`. The first argument it takes in is an array of integers called `inputArray`, and the second argument it takes in is an integer, `k`.

Extra: check if it works

```
1  public static int mystery1(int[] inputArray, int k) {
2      int x = inputArray[k];
3      int answer = k;
4      int index = k + 1;
5      while (index < inputArray.length) {
6          if (inputArray[index] < x) {
7              x = inputArray[index];
8              answer = index;
9          }
10         index = index + 1;
11     }
12     return answer;
13 }
```

start index

loop to the right

"new" smallest has been found

represents the smallest value

update the index to return

returns an index

4, 6, 3

Write the return value of `mystery1` if `inputArray` is the array {3, 0, 4, 6, 3} and `k` is 2. Then, explain in English what the mehtod `mystery1` does.

4

Extra Below is another function called `mystery2`. It takes an array of integers called `inputArray` as an argument and returns nothing.

```
1  public static void mystery2(int[] inputArray) {
2      int index = 0;
3      while (index < inputArray.length) {
4          int targetIndex = mystery1(inputArray, index);
5          int temp = inputArray[targetIndex];
6          inputArray[targetIndex] = inputArray[index];
7          inputArray[index] = temp;
8          index = index + 1;
9      }
10 }
```

iterating through array

find index of smallest value

swapping smallest value to the front

Describe what `mystery2` will do and return if `inputArray` is the array {3, 0, 4, 6, 3}. Then, explain in English what the method `mystery2` does.

Selection sort

30463

03463

03463

03346

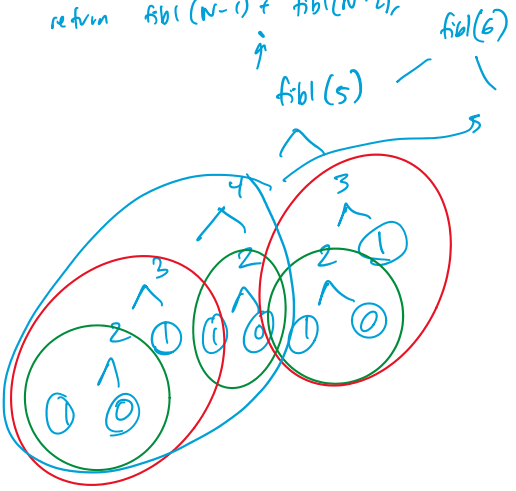
3 Fibonacci

Implement `fib1` recursively. `fib1` takes in an integer `N` and returns an integer representing the Nth Fibonacci number. The Fibonacci sequence is 0, 1, 1, 2, 3, 5, 8, 13, 21, ..., where 0 is the 0th Fibonacci number. As a reminder, the Nth Fibonacci number is calculated as follows:

```
fib(N) = fib(N - 1) + fib(N - 2)
```

Base Case(s)
Recursive Leap

```
public static int fib1(int N) {
    if (N < 2) {
        return N;
    }
    return fib1(N-1) + fib1(N-2);
}
```



Extra Implement `fib2` in 5 lines or fewer that avoids redundant computation. `fib2` takes in an integer `N` and helper arguments `k`, `f0`, and `f1` and returns an integer representing the Nth Fibonacci number. To compute the Nth fibonacci number, you should call `fib2(N, 0, 0, 1)`. If you're stuck, try implementing `fib1` iteratively and then see how you can transform your iterative approach to implement `fib2`.

```
public static int fib2(int N, int k, int f0, int f1) {
    if (k == N) {
        return f0;
    } else {
        return fib2(N, k+1, f1, f0+f1);
    }
}
```

build up

k is current iteration
f0 is current fib
f1 is next fib

Better than tree recursion!

Can be done w/o k by decrementing N

fib2(5, 0, 0, 1)
fib2(5, 1, 1, 1)
fib2(5, 2, 1, 2)
fib2(5, 3, 2, 3)
fib2(5, 4, 3, 5)
fib2(5, 5, 5, 8)

Tail Recursion: Computation first, then recurse
Tree Recursion: Recurse first, computation last