CS 61B Discussion 1 Spring 2017

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.

```java
int size = 27; //Declares a variable of type int and assigns it the value
    27. In Java, all variables must be declared before they are used
String name = "Fido"; //Declares a variable of type String and assigns it
    the variable "Fido"
Dog myDog = new Dog(name, size); //Declares and initializes a new
    variable of type Dog. Calls the Dog constructor to create a new
    object of type Dog
int x = size - 5; //Declares a new variable of type int and assigns it
    the value 22
if (x < 15) { //If x is less than 15, calls the bark method on an
    instance of the Dog class. Since x is 22, myDog.bark is not called
     myDog.bark(8);
}

while (x > 3) { //Checks if x is greater than 3 and if so calls myDog's
    play method. Subtracts 1, and as long as x is bigger than 3, goes
    back to the beginning of the loop. Play happens a total of 19 times.
     x -= 1;
     myDog.play();
}

int[] numList = {2, 4, 6, 8}; //Declares an array of ints and initializes
    it to {2, 4, 6, 8}
System.out.print("Hello "); //Prints the String "Hello " to the standard
    output
System.out.println("Dog: " + name); //Prints the String "Dog: Fido" to
    the standard output and then terminates the line

System.out.println(numList[1]); //Prints the String "4" to the standard
    output and then terminates the line. In Java, arrays are indexed from
    0
if (numList[3] == 8) { //numList[3] is equal to 8
     System.out.println("potato"); //Prints the String "potato" to the
         standard output and then terminates the line
}
```

Acknowledgement: This exercise is adapted from page 5 of our textbook Head First Java.

2 Mystery

```java
/** 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;
}

/** Extra for experts. This is another function. It takes an
  * array of integers and returns nothing at all. */
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;
    }
}
```

- What does `mystery` return if `inputArray` is the array 3, 0, 4, 6, 3, and `k` is 2?
  `It returns 4.`

- Describe, in English, what `mystery` returns.

  It returns the index of the smallest element that occurs at or after index `k` in the array. If `k` is greater than or equal to the length of the array or less than 0, an `ArrayIndexOutOfBoundsException` will be thrown, though this exception is not something you'd know without running the program.

  The variable `x` keeps track of the smallest element found so far and the variable `answer` keeps track of the index of this element. The variable `index` keeps track of the current position in the array. The while loop steps through the elements of the array starting from index `k` + 1 and if the current element is less than `x`, `x` and `answer` are updated.

- Extra for experts: What does `mystery2` do if `inputArray` is the array 3, 0, 4, 6, 3? Describe, in English, what `mystery2` does to the array.

  If `mystery2` is called on the array 3, 0, 4, 6, 3 then after the method runs, the array will be 0, 3, 3, 4, 6. Given any array, the method `mystery2` sorts the elements of the array in increasing order.

At the beginning of each iteration of the while loop, the first `index` elements of the array are in sorted order. Then the method `mystery` is called to find the index of the smallest element of the array occurring at or after `index`. The element at the index returned by `mystery` is then swapped with the element at position `index` so that the first `index` + 1 elements of the array are in sorted order.

This algorithm is called *selection sort*. We will talk about it more later on in the course.

3 Writing Your First Program

```
/** fib(n) returns the nth Fibonacci number, for n≥0.
  * The Fibonacci sequence is 0, 1, 1, 2, 3, 5, 8, 13, 21, ... */
public static int fib(int n) {
    if (n <= 1) {
        return n;
    } else {
        return fib(n - 1) + fib(n - 2);
    }
}
```

This solution above is recursive. We can also write it iteratively:

```
public static int fib(int n) {
    int f0 = 0;
    int f1 = 1;
    while (n > 0) {
        int temp = f1;
        f1 = f0 + f1;
        f0 = temp;
        n -= 1;
    }
    return f0;
}
```

Extra for experts: Complete `fib2` in 5 lines or less. Your answer must be efficient.

```
/** fib2(n, 0, 0, 1) returns the nth Fibonacci number, for n≥0. */
public static int fib2(int n, int k, int f0, int f1) {
    if (n == k) {
        return f0;
    } else {
        return fib2(n, k + 1, f1, f0 + f1);
    }
}
```

This is the tail-recursive solution. We can write this more succintly using the ternary operator:

```
/** fib2(n, 0, 0, 1) returns the nth Fibonacci number, for n≥0. */
public static int fib2(int n, int k, int f0, int f1) {
    return n == k ? f0 : fib2(n, k + 1, f1, f0 + f1);
}
```