1 Javaian Rhapsody

Next to each line, write out what you think the code will do when run. (Assume the Singer class exists and that the code below compiles.) Comments are above the line they describe.

```java
/* Declare a variable of type String and assign it the value "no". In Java, all variables must be declared before they are used. */
String disagree = "no";

/* Declare a variable of type int and assign it the value 7. */
int x = 7;

/* Declare a variable of type Singer and initialize it using the Singer constructor with the argument "Queen" */
Singer queen = new Singer("Queen");

/* Checks if x is greater than 0; if so, subtract 1 from x, then call queen’s sing method with argument "no", then go back to the beginning of the loop. queen object sings "no" 7 times. */
while (x > 0) {
    x -= 1;
    queen.sing(disagree);
}

/* Declares a variable of type String array and initializes it to hold three Strings. */
String[] phrases = {"Oh", "mamma mia", "let me go"};

/* Prints "Oh" to standard output */
System.out.print(phrases[0]);

/* Declares variable i and initialize to 0 and checks if it is less than 3. If so, print "mamma mia", then add one to i, then go back to the beginning of the loop and re-check that i < 3. "mamma mia" is printed out 3 times. */
for (int i = 0; i < 3; i += 1) {
    System.out.print(" " + phrases[1]);
}

/* Prints "let me go" to standard output. */
System.out.print(" " + phrases[2]);
```
2 Fibonacci

Implement this function recursively:

```java
/** 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);
    }
}
```

Now implement a function that provides the same results but avoids redundant computations:

```java
public static int fib2(int N) {
    int k = 0;
    int f0 = 0;
    int f1 = 1;
    while (N != k) {
        int temp = f1;
        f1 = f0 + f1;
        f0 = temp;
        k++;
    }
    return f0;
}
```

Now use this modified function header to write a recursive solution that avoids redundant computations and that has a body that is 5 lines or fewer. The inputs are defined as follows: your goal is to compute the $N$th fib number, you are currently computing the $k$th fib number (which is $f_0$), and the $(k+1)$th fib number is $f_1$.

```java
public static int fib3(int N, int k, int f0, int f1) {
    if (N == k) {
        return f0;
    } else {
        return fib3(N, k + 1, f1, f0 + f1);
    }
}
```

To compute the $N^{th}$ fibonacci number using `fib3`, call `fib3(N, 0, 0, 1).`
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;
}

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 at runtime.
  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 return if inputArray is the array {3, 0, 4, 6, 3}? Then, describe, in plain English, what mystery2 does to the array.
  mystery2 doesn’t return anything because its return type is void.
  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.