

# Exam 2 Procedures, Tips and Practice Problems

## General Information

Exam 2 will be a 75-minute, timed, pencil-and-paper written exam. No books, no notes, no electronic devices, no headphones, no collaboration. The problems will primarily involve writing Java code or reading and interpreting Java code.

## Exam Format

The exam covers everything done in lectures *roughly* through Wednesday, April 3 and in labs through Lab 7. The exam will concentrate on topics covered since the first exam. Most problems will require loops, arrays, lists, or recursion. However, since the subject matter is cumulative, *knowledge of everything from the first exam is assumed*. If you had any trouble with Exam 1 you may also want to refer to the review sheet for that exam. The list below is an overview of the main topics. You will not be tested on JUnit or the Eclipse debugger. **The exam does not cover sorting. Coverage of recursion is limited to reading and interpreting recursive methods (writing recursive methods will be on the final exam).**

- Loops
- Arrays
- Two-dimensional arrays
- `ArrayLists`
- Array, list and string algorithms – counting, searching, inserting, deleting, etc.
- Wrapper classes
- Reading text files
- Using `Scanner` to parse text
- Reading and interpreting recursive methods

You do not have to memorize methods from the Java API. You should know `System.out.print` and `System.out.println`. You should know how to use `String`, `Scanner`, `Math`, `Random`, `File`, and `ArrayList<E>`. You should know how to read text files and how to read input from `System.in`. For

specific methods from these classes that might be needed, we'll provide you with the minimal one-sentence descriptions from the API.

## How to Prepare

The most important thing you can do to prepare for an exam like this is to practice solving problems and writing code. You should write your solutions first on paper (since that is the format of the exam). Then check your work by copying what you've written into Eclipse and testing it. We encourage you to post and discuss your sample solutions to practice problems on Piazza. **Please remember that we will NOT grade you on formatting or indentation, so don't waste too much time trying to format your code on the exam.**

You will need to write quickly and accurately, and to recognize correctly written code without the help of Eclipse. You will not be expected to write comments or documentation or define symbolic constants for numbers. (However, including brief comments can sometimes help us interpret what you were trying to do, in case you have errors.)

## More Practice

There are some sample problems below.



Another thing you can do to practice is to make sure that you can understand and reproduce the examples that were done in class. Just take an example we've done, delete the method body, and try to re-do it on your own.

There are more problems to practice on in the book, of course.

Another entertaining way to practice writing Java code is the interactive site

<http://codingbat.com/java>  <http://codingbat.com/java>. There are lots of problems involving arrays and strings and recursion.

This site is fun too:

<https://leetcode.com/tag/array/>  <https://leetcode.com/tag/array/>  
<https://leetcode.com/tag/string/>  <https://leetcode.com/tag/string/>

*(However, note that the above do not include problems involving `ArrayList`, `Scanner`, or reading files, which ARE definitely included on our exam.)*

## Some Practice Problems

*You are encouraged to post your sample solutions on Piazza for discussion.*

1) Some loop and array examples. Write a static method for each that has all needed information as parameters. (You don't have to create an entire class.)

- a) Given an array of doubles, return the average.
- b) Given a sentence, find and return the longest word.
- c) Given a string, return the string with all spaces and non-alphabetic characters replaced by the character '#', e.g. "Hello, world!" becomes "Hello###world#" (You can use the static method `Character.isAlphabetic(char c)` to determine whether a given character is alphabetic.)
- d) Interest is added to the balance of a savings account each month. Write a method that, given an annual interest rate and an initial balance, determines how many months it takes for the balance to double.
- e) Given an `ArrayList` of `Integers`, determine whether they are in increasing order.
- f) Given a string, return the index of the first vowel (or -1 if there are none).
- g) Given a string, determine whether any letter appears two or more times.
- h) Given an array of ints, reverse its contents (the method must modify the given array and returns void).
- i) Given an array of integers, determine whether the array is a permutation of the numbers 0 through  $n - 1$ , where  $n$  is the length of the array. (A permutation means that each number appears exactly once.)
- j) Given a number  $n$ , print out a reverse diagonal line of  $n$  stars:
 

```

* *
*
*
```
- k) Given a 2D array of doubles, return a 1D array whose  $i$ th entry is the average of the  $i$ th column.
- l) Given a 2D array of ints, find the column with the maximum sum.
- m) Given positive integers  $w$  and  $h$  and an `int[]` array `arr` of length  $w * h$ , return a 2d array with  $h$  rows and  $w$  columns that contains the numbers in `arr`, listed left-to-right and top-to-bottom.
- n) Given an integer  $n$ , return the smallest prime number that is larger than  $n$ . (A number is prime if it is greater than 1 and has no divisors other than 1 and itself.)
- o) Given an array of positive integers, "collapse" the array to remove duplicates, and fill in the unused cells at the end with zeros. For example, given the array `[5, 4, 5, 6, 4, 2]`, after this method executes the array should be `[5, 4, 6, 2, 0, 0]`. The method modifies the given array, and returns void.

- p) Given an array of positive integers, return a new array containing the same numbers, in the same order, but without duplicates. For example, given the array [5, 4, 5, 6, 4, 2], the method returns [5, 4, 6, 2].
- q) Given an instance of Random, generate a list of numbers between 0 and 99, inclusive, stopping when the same number has appeared more than once. The method returns a list of all the generated numbers. (The ArrayList contains() method might be useful.)
- r) Given a string, return a new string with the words in the opposite order. (E.g. given "He's dead, Jim", return "Jim dead, He's".)
- s) Given an array of ints, swap the first half with the second half. The method modifies the given array and returns void. If the length is odd, the middle element is not moved. For example, if called on the array [10, 20, 30, 40, 50, 60, 70], after the method executes the array would be [50, 60, 70, 40, 10, 20, 30].

~~2) Write a program that will remove all the // style comments from a Java file. Your program should prompt the user to enter the name of the input file. The output file should have the same name as the input file but should end with the extension ".out" instead of ".java". The output file should be the same as the input file except that all // style comments are removed. (You can assume that the sequence "//" does not occur inside any String literals within the program.)~~

Writing to a file will not be on the exam, you only need to know reading files.

3) Trace the execution of the call **enigma(12,0)** and show all output that is produced.

```
public static void enigma (int x, int y) {
    while (x > 0){
        if (x % 2 == 0){
            y = y + 1;
        } else {
            x = x + 2;
        }
        x = x - y;
        System.out.println(x + ", " + y);
    }
}
```

4) Given the array: `int[] test = {6, 7, 4, 3, 5, 2, 7, 9, 8};` Trace the execution of the call `whatever(test)` and show contents of the array after the method returns.

```
public static void whatever (int[] arr) {
    int i = 0;
    for (int count = 0; count < arr.length; count += 1)
    {
        if (arr[i] % 2 != 0)
        {
            for (int j = i; j < arr.length - 1; j += 1)
            {
                arr[j] = arr[j + 1];
            }
            arr[arr.length - 1] = 0;
        }
    }
}
```

```

    else
    {
        i += 1;
    }
}
}

```

5) Write a static method `getPassword` that will read a user's password from `System.in`. The user has to enter the password twice. The method should iterate the following steps as many times as necessary until the user successfully enters two values that match:

1. prompt the user and read the password
2. prompt the user and read the password again
3. check that the second entry matches the first

(The method has no parameters and should return the entered password as a `String`.)

6) Suppose that a text file contains lines with a name and phone number having the format

name, xxx-xxx-xxxx

a) Create a class `Contact` suitable for storing a name and phone number (the phone number may be stored as a `String`). It should include the constructor and methods:

```

Contact(String givenName, String givenPhoneNumber)
String getName()
String getPhoneNumber()    // returns phone number as String

```

b) Create a class `ContactDirectory` suitable for storing a list of `Contacts`. The `ContactDirectory` should have the methods:

```

// adds the given contact to the directory
void addContact(Contact c)

```

```

// add all contacts from a file of the above form
void addFromFile(String filename) throws FileNotFoundException

```

```

// returns phone number for name, or null if name is not in the list
String lookUp(String name)

```

7)

a) Given the method `mystery` below, determine the output printed by the call `mystery(10)`. (It might be helpful to sketch the call stack as you go.)

```

public static void mystery(int x)
{
    if (x == 1) {
        System.out.println("pooh");
    }
    else if (x % 2 == 0)

```

```

    {
        System.out.println(x);    (**)
        mystery(x / 2);          (*)
    }
    else
    {
        mystery(x - 1);
    }
}

```

b) Suppose we have a method `mystery2` that is the same as `mystery` except that the lines labeled `(*)` and `(**)` are switched. Trace the call `mystery2(10)`.

c) What happens when you call `mystery(-1)` ? Explain.

8) What does the following method do? Rewrite it so it produces the same results but does not use recursion.

```

public static boolean whoKnows(int[] arr, int i, int j)
{
    if (i >= j) {
        return true;
    }
    else
    {
        int mid = (i + j) / 2;
        boolean leftOk = whoKnows(arr, i, mid);
        boolean rightOk = whoKnows(arr, mid + 1, j);
        return leftOk && rightOk && arr[mid] <= arr[mid + 1];
    }
}

```