CS163/164 Final Study Guide

Fall 2016

Contents

1 Disclaimer **3**

2 Abstract Classes & Interfaces **3**
- 2.1 Formatting . . . 3
- 2.2 Common Methods to Override . . . 3
- 2.3 Differences between Abstract Classes & Interfaces . . . 3

3 Recursion **3**
- 3.1 Steps . . . 3
- 3.2 Examples . . . 4
- 3.3 Suggestions, Warnings, Common Errors . . . 4

4 ArrayList and Collections **4**
- 4.1 Creating an ArrayList . . . 4
- 4.2 Common ArrayList Methods . . . 4
- 4.3 Examples . . . 5
- 4.4 Suggestions, Warnings, Common Errors . . . 5

5 Sorting and Complexity **5**

6 File I/O and Exceptions **5**
- 6.1 Creating Fille I/O Objects . . . 5
  - 6.1.1 Common Scanner Methods . . . 6
  - 6.1.2 Common PrintWriter Methods . . . 6
- 6.2 Suggestions, Warnings, Common Errors . . . 6

7 Arrays **7**
- 7.1 Examples . . . 7
- 7.2 Suggestions, Warnings, Common Errors . . . 8

8 Conditionals **8**
- 8.1 if/if/if vs. if/else if/else if . . . 8
- 8.2 Switch . . . 9
- 8.3 Suggestions, Warnings, Common Errors . . . 10

9 Random Topics **10**
- 9.1 Printf . . . 10
- 9.2 Loops . . . 11
  - 9.2.1 While and Do-While . . . 11
  - 9.2.2 For and For-Each . . . 11

10 Practice Written Exam **13**
10.1 Short Answer 13
10.2 Tracing 14

11 Practice Programming Exam **15**

12 Suggestions/Resources for Studying **15**
12.1 Study Suggestions 15
12.1.1 Written Portion 15
12.1.2 Programming Portion 16
12.2 Resources 16

1 Disclaimer

This is a review of this courses' material, but there may be material on the exam not covered in this study guide.

2 Abstract Classes & Interfaces

2.1 Formatting

Using/implementing an abstract class & an Interface

- `public returnType methodName (params if needed)`
- `public class Practice extends AbstractClassName { }`
- `public class Practice implements Interface { }`

2.2 Common Methods to Override

- toString - `public String toString() { }`
- equals - `public boolean equals (Object o) { }`
- compareTo - `public int compareTo (ClassName c) { }`

2.3 Differences between Abstract Classes & Interfaces

- Abstract methods can have code filled in, where an interface must only contain the method headings/stubs only.

3 Recursion

Recursion is the idea that the method calls itself. By doing this you replace the need for loops.

3.1 Steps

1. Find simplest case. When do you need to do NO work? This is your base case.
2. What do you do to the bigger problem so it "fits" into the simpliest case? You want this call (which is your recursive call) to get you closer to your base case.

3.2 Examples

```
public static int fib (int n) {
  // base case: when n is 0 or 1
  if (n == 0 || n == 1) return n;
  // recursive call
  else {
    return fib(n-1) + fib(n-2);
  }
}
```

3.3 Suggestions, Warnings, Common Errors

- When starting, try not to think too much about the code. Make sure you know what the problem is and how you'd solve it in "English" (versus not in code).
- If you get a StackOverFlow error go back and check your recursive call and make sure you get closer to your base case. A StackOverFlow error is very similar to an infinite loop.

4 ArrayList and Collections

4.1 Creating an ArrayList

Syntax: `ArrayList<Object> name = new ArrayList<Object>();`

4.2 Common ArrayList Methods

```
ArrayList<Character> cList = new ArrayList<Character>();
```

- `arrayListName.size()` - used to get the number of elements in the arraylist. `cList.size();`
- `arrayListName.get(index)` - used to get an element at an index (like the square brackets are for array). `cList.get(3);`
- `arrayListName.add(value)` - used to add something at the end of the arraylist. `cList.add('a');`
- `arrayListName.add(index, value)` - used to add something at an index (this pushes every element after that index farther back). `cList.add(1, 'b');`
- `arrayListName.remove(index)` - used to remove at an index. `cList.remove(0);`
- `arrayListName.remove(value)` - used to remove a value (finds the first instance of that value). `cList.remove('b');`

4.3 Examples

4.4 Suggestions, Warnings, Common Errors

- if you're making an arraylist of a primitive type, you must use the wrapper class. For example, to make an int arraylist you need to the following syntax: `ArrayList<Integer> listName = new ArrayList<Integer>();`. To make a String arraylist however you can use this syntax: `ArrayList<String>listName = new ArrayList<String>();`

5 Sorting and Complexity

Tutorials Point has great tutorials and visuals on sorting.

6 File I/O and Exceptions

6.1 Creating Fille I/O Objects

```
// Scanner to read from a keyboard
Scanner name = new Scanner (System.in);

// Scanner to read from a file: way 1
try {
  Scanner name = new Scanner (new File (filename));
} catch (IOException e) {
        System.out.println(e.getMessage());
}
// Scanner to read from a file: way 2
try {
  File f = new File (filename);
  Scanner name = new Scanner (f);
} catch (FileNotFoundException e) {
        System.out.println("Cannot read from " + filename);
}
// PrintWriter
try {
  PrintWriter name = new PrintWriter (new File (filename));
} catch (IOException e) {
        System.out.println("Cannot write to " + filename);
}
// NOTE: Your try-catches may be different (can use Exception e, FileNotFoundExc
//           and you can personalize what is printed in the catch.
```

6.1.1 Common Scanner Methods

NOTE: reading from the keyboard and reading from a file use the same methods.

- `scannerName.next()` - used to read one word
- `scannerName.nextLine()` - used to read one line
- `scannerName.nextInt()` - used to read the next integer
- `scannerName.nextDouble()` - used to read the next double

6.1.2 Common PrintWriter Methods

- `printWriterName.print()` - used to write with no new line character
- `printWriterName.println()` - used to write with new line character
- `printWriterName.printf()` - used to format/personalize what is written
- `printWriterName.close()`

6.2 Suggestions, Warnings, Common Errors

- When reading from a file make sure to have the "new File" part of the declaration. If this isn't included the scanner will read the string itself. For example, if:

```
String filename = "in.txt";
try {
  Scanner reader = new Scanner (filename);
  String firstLine = reader.nextLine();
  System.out.println(firstLine);        // prints in.txt
} catch (Exception e) {
 System.out.println(e.getMessage());
}
```

- Only put the file name in quotes if it is not a variable.

```
public static void readFile (String filename) {
  try {
    Scanner read = new Scanner (new File (filename));
   } catch (Exception e) {
     System.out.println("Cannot read " + filename);
   }
   }
    // OTHER WAY
    public static void readFile () {
      try {
```

```
            Scanner read = new Scanner (new File ("in.txt"));
        } catch (Exception e) {
            System.out.println("Cannot read in.txt");
        }
    }
```

- When using `scannerName.hasNext()` or `scannerName.hasNextLine()` make sure to parse/read in that same form. Meaning when using `scannerName.hasNext()` use `scannerName.next()` and when using `scannerName.hasNextLine()` use `scannerName.nextLine()`.
- If your data doesn't write to the file make sure you closed your PrintWriter!

7 Arrays

Creating Arrays:

```
// 1D
type [] name = new type [size];                    // declaring
type name [] = {val1, val2, val3, ...};          // initializing
// 2D
type name [][] = new type [height][width];  // declaring
type [][] name = { {val1, val2, val3},           // initializing
                   {val4, val5, val6} };
```

NOTES: Your brackets can be between the type and the name or between the name and the equals sign. Spacing doesn't matter when initializing a 2D array.

7.1 Examples

```
double [] d = new double [3];
String [] s = {"Hi", "There"};
int [][] sudoku = new int [9][9];

// getting at one element
System.out.println("first element of s: " + s[0]);

// getting number of elements in an array
// Notice: there are no praenthesis after length
int size = d.length;

// assigning a value in a 1D array
d[1] = 5.5;

// assingin a value in a 2D array
sudoku[0][1] = 3;
```

```
// One way of printing: Arrays.toString()  (use when you want to print [element1
System.out.println(Arrays.toString(s));

// Second way of printing: loops  (use when you need to personalize printing)
// for loop
for (int i = 0; i < s.length; i++)
   System.out.print(s[i]);

System.out.println(); // used for spacing

// for-each loop
for (int [] row : sudoku) {
  for (int col : row)
       System.out.print(col);
   System.out.println(); // used for spacing
}
 /* This prints;
 first element of s: Hi
[Hi, There]
HiThere
030000000
000000000
000000000
000000000
000000000
000000000
000000000
000000000
000000000
  */
```

7.2 Suggestions, Warnings, Common Errors

- When getting the size of an array you don't use parenthesis after the length (becuase length isn't a method).
- Use `Arrays.toString(arrayName)` NOT `arrayName.toString()` to print an array

8 Conditionals

8.1 if/if/if vs. if/else if/else if

If more than one condition is true in an if/if/if statements then it'll evaluate every condition that's true. If more than one condition is true for an if/else if/else if statement only the first condition that is true evaluates.

Example:

```
int num = 0;
// if/if/if statement
if (num == 0)
  System.out.println("ifs: num equals zero");
if (num <= 3)
  System.out.println("ifs: num is less than or equal to three");
if (num > -1)
  System.out.println("ifs: num is greater than -1");
// if/else if/else if statement
if (num == 0)
  System.out.println("if/else ifs: num equals zero");
else if (num <= 3)
  System.out.println("if/else ifs: num is less than or equal to three");
else if (num > -1)
  System.out.println("if/else ifs: num is greater than -1");

 /* What's printed:
ifs: num equals zero
ifs: num is less than or equal to three
ifs: num is greater than -1
if/else ifs: num equals zero
  */
```

8.2 Switch

General Syntax:

```
switch (variable) {
case variableOption0:   (do something) break;
case variableOption1:   (do something) break;
case variableOption2:   (do something) break;
       .
       .
       .
default: (do something) break;        // default is not required
                                      // similar to an else
}
```

Example:

```
char c = 'd';
switch (c) {
case 'a': System.out.println("apple"); break;
case 'b': System.out.println("babble"); break;
case 'c': System.out.println("cobble"); break;
```

```
default: System.out.println("something else"); break;
}
String coolio = "comp sci";
String returnString = "";
switch (coolio) {
case "Snoop Dog":  returnString = "Snoop Dog is pretty coolio yo"; break;
case "Barbie Girl": returnString = "Good song..."; break;
case "comp sci": returnString  = "Dang these switch statements are cool"; break;
default: returnString = "None of the above"; break;
}
```

8.3 Suggestions, Warnings, Common Errors

- When using switch statements, be aware when you want to add breaks and when you do not. Breaks are not required but if you don't have them in your switch statement you can have the "fountain" effect (however, at times this is desired). Example:

```
int num = 3;
switch (num) {
case 0:
case 1:
case 2:
case 3:
case 4:
case 5:
  System.out.println("0 <= number <= 5");
  break;
default:
  System.out.println("number < 0 or number > 5");
}
```

- You can only use characters, byte, shorts, ints, Strings, and enums.

9 Random Topics

9.1 Printf

Syntax: `System.out.printf("formatter", args)`
Formatters:

- %d - integer
- %f - floating point number (ie double)
- %s - String

- %c - character

Examples:

```
// printing an integer
int i = 30;
System.out.printf("%d", 3);      // no new line character
System.out.printf("%d\n", 3);    // prints with new line character
System.out.printf("%04d\n", i); // prints 0030
// printing a double
double d = 3.423490;
double d1 = 1;
System.out.printf("%.2f\n", d1); // prints 1.00
System.out.printf("%.4f\n", d);  // prints 3.4235 (it rounds)
// printing a String
String s = "Pi to";
System.out.printf("%s\n", "Hello");
System.out.printf("%s %d decimals is %.2f\n", s, 2, Math.PI);
// prints Pi to 2 decimals is 3.14
```

9.2 Loops

9.2.1 While and Do-While

Syntax for a while loop:

```
// declare loop variable
while (range) {
  // do something
  // update loop variable
}
```

Syntax for a do-while loop:

```
// declare loop variable
do {
  // do something
  // update loop variable
} while (range);
```

9.2.2 For and For-Each

Syntax for a for loop:

```
for (declare loop variable; range; update) {
  // do something
}
```

Syntax for a for-each loop:

```
for (variable : structure) {
  // do something
}
```

An example of a for and for-each loop (looping through an arraylist and printing each element seperated by an ampersand(&)):

```
ArrayList<Double> dList = new ArrayList<Double>();
// for loop
for (int i = 0; i < dList.size(); i++) {
  double d = dList.get(i);
  System.out.print(d + "&");
}
System.out.println(); // used for spacing
// for-each loop
for (double d : dList) {
  System.out.print(d + "&");
}
```

10 Practice Written Exam

10.1 Short Answer

1. Name the eight primitive types.
2. Declare and initialize an arraylist of type double called `doubleList`
3. Add 3.14, 123.4, 86.3, and -2.1 to `doubleList`
4. Print the contents of `doubleList` in the format: [element1, element2, element3, ...]
5. Print the contents of `doubleList` in the format: element1:element2:element3:...
6. Print the third element of `doubleList`
7. Add 98.5 so it's the second element of `doubleList`
8. Remove the last element of `doubleList`
9. Remove the value 86.3 from `doubleList`
10. Declare and initialize an arraylist of type String called `lines`
11. Create a Scanner (with a try-catch) that reads from a file named `input.txt`
12. Use the Scanner from the previous question to read each line and add it to the arraylist called `lines`. Assume try-catch is surrounding your answer.
13. Print the number of lines in the file
14. Create a PrintWriter (with a try-catch) that writes to the filename stored in the variable `outfile`
15. Use your PrintWriter from the previous question and write the arraylist `lines` to the file (every element on a new line). Assume try-catch is surrounding your answer.
16. Given the following array, after two iterations of bubble sort what does this array look like? `int [] array = {-5, 2, 4, 1, 9};`
17. Given the following array, after two iterations of selection sort what does this array look like? `int [] array = {-5, 2, 4, 1, 9};`

10.2 Tracing

```
import java.util.ArrayList;
public class Tracing {
  public ArrayList<Integer> list = new ArrayList<Integer>();

  public int mystery0 (){
    int count = 0;
    int [] array = new int [list.size()];
    for (int i = 0; i < list.size(); i++) {
      for (int j = 0; j < list.size(); j++){
        if (list.get(i) == list.get(j))
          count++;
      }
      array[i] = count;
    }
    int m = array[0];
    int i = 0;
    for (int j = 0; j < array.length; j++)
      if (array[j] > m) {
        m = array[j];
        i = j;
      }
    count = m;
    return list.get(i);
  }
  public void mystery1 () {
    for (int i = 0; i < list.size(); i++)
      list.set(i, list.get(i)*2);
  }
  public int mystery2 () {
    int ret = list.get(0);
    for (int i : list){
      ret = i;
    }
    return ret;
  }

  public static void main (String [] args){
    Tracing t = new Tracing();
    t.list.add(1); t.list.add(1, 4); t.list.add(0,5);
    t.list.add(2, 3); t.list.add(1); t.list.add(1, 1);
    // Question 1
    System.out.println(t.list.size());
    // Question 2
    System.out.println(t.list);
```

```
    // Question 3
    System.out.println(t.mystery0());
    t.mystery1();
    //Question 4
    System.out.println(t.list);
    // Question 5
    System.out.println(t.mystery2());
  }
}
```

11 Practice Programming Exam

Instructions: Create a project (any name) and class named `QPractice` with a main method. Download the interface and make `QPractice` implement it. The test code is below. Remember to test as you go!

1. Declare and initialize an private non-static class arraylist of Strings named `words`.
2. Complete the method named `read` by reading each word and add it to the arraylist `words`.
3. Complete the method named `find` which returns the index of the first word that matches the parameter, if the word is not in the arraylist return -1. This method needs to be case-insensitive.
4. Complete the method named `mostCommon` which finds the most common (case-insensitive) word in the arraylist `words`. Then return that word in uppercase.
5. Complete the `write` method that prints to the file the most common word and the index of the word "fire" (each on a new line).

Test Code:

```
QPractice q = new QPractice();
q.read("../in.txt");
q.write("../out.txt");
```

12 Suggestions/Resources for Studying

12.1 Study Suggestions

12.1.1 Written Portion

- Practice writing code in Eclipse and then before running it write down what will print out or what each variable's value is
- Go back to past exams and fix mistakes

12.1.2 Programming Portion

- Re-do recitations and assignments without help
- Do practice problems on CodingBat and Hackerrank

12.2 Resources

- TutorialsPoint has good tutorials and visuals for sorting.
- CodingBat and Hackerrank have lots of practice programming questions