# Practice Final Self Assessment
Summer 2020

1. Reference Mystery, 5 points. The following program produces 3 lines of output. Write the output in the box below, exactly as it would appear on the console.

```java
import java.util.*;
public class ReferenceMystery {
   public static void main(String[] args) {
      int x = 4;
      int y = 8;
      int[] data = {5, 10, 15};
      x = mystery1(y, data);
      System.out.println(y + " " + Arrays.toString(data));
      mystery2(x, y);
      System.out.println(x + " " + y);
   }

   public static int mystery1(int n, int[] numbers) {
      n = n + 100;
      numbers[2]--;
      System.out.println(n + " " + Arrays.toString(numbers));
      return numbers[1] * 2;
   }

   public static void mystery2(int x, int y) {
      x++;
      y = y * 3;
   }
}
```

2. Array Simulation, 10 points. You are to simulate the execution of a method that manipulates an array of integers. Consider the following method:

```
      public static void mystery(int[] arr) {
         for (int i = 1; i < arr.length - 1; i++) {
            if (arr[i] > i + 1) {
               arr[i + 1] = arr[i] - arr[i - 1];
            }
         }
      }
```

In the left-hand column below are specific arrays of integers. You are to indicate in the right-hand column what values would be stored in the array after method mystery executes if the integer array in the left-hand column is passed as a parameter to mystery.

```
            Original Array          Final Array
            --------------          ----------------------------

            [2, 3, 1]               ____________________________

            [2, 6, 3, 5]                  ____________________________

            [3, 3, 7, 9]                  ____________________________

            [2, 4, 5, 6, 8]         ____________________________

            [1, 5, 8, 4, 10, 9]     ____________________________
```

3. Inheritance Mystery, 6 points. Consider the following classes:

Given the classes to the left, write the output produced by the client code below exactly as it would appear on the console.

```
public class Vier extends Drei {
   public void method2() {
      super.method2();
      System.out.print("Vier 2 ");
   }

   public String toString() {
      return "Vier " + super.toString();
   }
}


public class Zwei extends Eins {
   public void method2() {
      System.out.print("Zwei 2 ");
      method1();
   }
}

public class Drei extends Zwei {
   public void method1() {
      System.out.print("Drei 1 ");
   }

   public String toString() {
      return "Drei";
   }
}

public class Eins {
   public String toString() {
      return "Eins";
   }

   public void method1() {
      System.out.print("Eins 1 ");
   }

   public void method2() {
      System.out.print("Eins 2 ");
   }
}

// client code
public static void main(String[] args) {
   Eins[] elements = { new Zwei(), new Eins(), new Vier(), new Drei() };
   for (int i = 0; i < elements.length; i++) {
      System.out.println(elements[i]);
      elements[i].method1();
      System.out.println();
      elements[i].method2();
      System.out.println();
      System.out.println();
   }
}
```

4. ArrayList Debugging, 5 points. Consider a static method called split that takes an ArrayList of integer values as a parameter and that replaces each value in the list with a pair of values, each half the original. If a number in the original list is odd, then the first number in the new pair should be one higher than the second so that the sum equals the original number. For example, if a variable called list stores this sequence of values:

```
      [18, 7, 4, 24, 11]
```

The number 18 is split into the pair (9, 9), the number 7 is split into (4, 3), the number 4 is split into (2, 2), the number 24 is split into (12, 12) and the number 11 is split into (6, 5). Thus, the call:

```
      split(list);
```

should cause list to store the following sequence of values afterwards:

```
      [9, 9, 4, 3, 2, 2, 12, 12, 6, 5]
```

The following is a proposed implementation of split:

```
      public static void split(ArrayList<Integer> list) {
         for (int i = 0; i < list.size(); i++) {
            int n = list.get(i);
            list.add(i, n / 2 + n % 2);
            list.add(i + 1, n / 2);
         }
      }
```

This implementation has one or more bugs. Modify the implementation so that it behaves as described above. Your modified method should retain the same basic approach to the problem as the buggy implementation; you should not write an entirely new implementation. You may assume that all numbers in the list are nonnegative.

You may not construct any extra data structures to solve this problem. You must solve it by manipulating the ArrayList you are passed as a parameter. See the cheat sheet for a list of available ArrayList methods.

5. File Processing, 10 points. Write a static method called formatList that takes a Scanner connected to an input file as a parameter and prints to System.out the input with certain lines indented and with asterisks. The lines to be indented all begin with at least one period. These leading period(s) should not be printed. For each line with leading period(s), you should print the text on that line (not including the period(s)) preceded by four spaces of indentation per period, an asterisk, and a space.

For example, consider the following input file:

```
CSE schedule

.week one
..static methods
..System.out.println()
..expressions

.week two
..for loops
..constants
..parameters
```

Then suppose the text above is stored in a Scanner called input and we make this call:
formatList(input);

The method should print the following output to System.out:

```
CSE schedule

* week one
      * static methods
      * System.out.println()
      * expressions

* week two
      * for loops
      * constants
      * parameters
```

Notice that input lines can be blank lines, and that input lines can contain periods of their own. For example, the periods in "System.out.println()" are not interpreted as indentation because they are not at the beginning of the line. Also note that lines without leading periods (like "CSE schedule") are printed as-is, with no indentation or asterisks.

You may not construct any extra data structures to solve this problem, though you may create as many String or primitive variables as you like.

6. File Processing, 10 points. Write a static method called calculateGrade that takes a Scanner connected to input file as a parameter. The file will contain a series of records representing a student’s performance on various assignments (homeworks and exams). Each record consists of three tokens: a label (either "homework" or "exam" in any casing), the student’s score on that assignment, and the total possible points on the assignment. Your method should calculate and print the student’s homework grade, exam grade, and overall grade. The overall grade is calculated by taking the average of the percentage of points earned on homeworks and the percentage of points earned on exams.

For example, suppose an input file contained the following text:

```
homework 18 20         HOMEWORK               12
 20        Exam 87           100
HoMeWOrk
 23              25 exam 44   75
```

Then suppose the text above is stored in a Scanner called input and we make this call:

```
calculateGrade(input);
```

The method should produce the following output to System.out:

```
Homeworks: 53 / 65 = 81.53846153846153
Exams: 131 / 175 = 74.85714285714286
Overall grade: 78.19780219780219
```

This student earned 53 homework points (18 + 12 + 23) out of a possible 65 points (20 + 20 + 25) and 131 exam points (87 + 44) out of a possible 175 points (100 + 75). Their overall grade is the average of their homework percentage (81.538...) and their exam percentage (74.857...). Notice that none of the percentages are rounded.

You may assume the input file contains only valid records; that is, tokens occur in multiples of three and consist of a String followed by two integers. You may also assume that the only labels will be "homework" or "exam" (in any casing), that all integers in the file will be positive, and that the input file contains at least one homework record and at least one exam record.

7. Arrays, 10 points. Write a static method named sweep that accepts an array of integers as a parameter and performs a single "sweep" over the array from lowest to highest index, comparing adjacent elements. If a pair of adjacent elements is not in increasing order(if the element at the lower index has a greater value than the element at the higher index), your method should swap them.

For example, in an array of 6 elements, your method first examines elements at indexes 0 and 1, then 1 and 2, then 2 and 3, 3 and 4, and finally 4 and 5, swapping if they are out of order. One side effect of this method is that the single element with the largest value will be moved into the final index of the array. (Repeated "sweeping" can be used to sort an array.)

If your method ends up swapping any elements, it should return true to indicate that the array was changed. Otherwise (if the array was already in increasing order before the sweep), it should return false.

The following table shows some calls to sweep and their expected results:

| arr | arr after call to sweep(arr) | Return Value |
|---|---|---|
| {1, 6, 2, 7, 3, 6, 4} | {1, 2, 6, 3, 6, 4, 7} | true |
| {9, 4, 2, 1, 3, 12, 14, 6, 8} | {4, 2, 1, 3, 9, 12, 6, 8, 14} | true |
| {3, 4, 8, 2, 1, 8, 8, 4, 12} | {3, 4, 2, 1, 8, 8, 4, 8, 12} | true |
| {-1, -4, 17, 4, -1} | {-4, -1, 4, -1, 17} | true |
| {2, 3, 5, 7, 11, 13} | {2, 3, 5, 7, 11, 13} | false |
| {42} | {42} | false |

You may assume that the array passed to your method is not null and has a length of at least 1.

8. Critters, 15 points. Write a class called Sponge that extends the Critter class. The instances of the Sponge class infect if an enemy is front of them, hop if there is an empty space in front of them, and otherwise turn left or right. They display themselves as square brackets with one or more dashes inside, as in "[-]" or "[----]". They are always colored yellow. They should always display at least one dash and should initially display one dash. Each time a Sponge infects, it increases the number of dashes in the display by one and each time a Sponge turns it decreases the number of dashes in the display by one unless it has gotten down to a single dash. In deciding which direction to turn, each Sponge should follow a pattern that repeats every three turns. The first turn should be left and the second and third turns should be right. Then it repeats with another left turn followed by two right turns, and so on.

9. Arrays, 15 points. Write a static method named maxes that takes two arrays of integers as parameters and returns a new array that contains the larger element at each index of the parameter arrays. If the two arrays are not the same length, the result array should be the same length as the longer array, and should include elements from the longer array at indexes that do not exist in the shorter array.

For example, suppose the following arrays are declared:

```
int[] arr1 = {1, 2, 3, 4, 5};
int[] arr2 = {3, 1, 3, 6, 3};
int[] arr3 = {-1, -1, -1, -1, -1, -1, -1};
int[] arr4 = {5, 5, 5, 5};
```

The following table shows the result of various calls to maxes:

```
Call                    Array Returned
--------------------------------------
maxes(arr1, arr2);      [3, 2, 3, 6, 5]
maxes(arr1, arr3);      [1, 2, 3, 4, 5, -1, -1]
maxes(arr1, arr4);      [5, 5, 5, 5, 5]
maxes(arr2, arr3);      [3, 1, 3, 6, 3, -1, -1]
maxes(arr2, arr4);      [5, 5, 5, 6, 3]
maxes(arr3, arr4);      [5, 5, 5, 5, -1, -1, -1]
```

Your method must not modify either of the parameters. You may assume that both arrays are not null. You are limited to the methods on the cheat sheet in solving this problem.

10. Programming, 10 points. Write a static method called undouble that takes a string as a parameter and that returns a new string obtained by replacing every pair of repeated adjacent letters with one of that letter. For example, the String "bookkeeper" has three repeated adjacent letters ("oo", "kk", and "ee"), so undouble("bookkeeper") should return the string "bokeper".

The following table shows some calls to undouble and their expected results:

| Call | Return Value |
|---|---|
| undouble("odegaard") | "odegard" |
| undouble("baz") | "baz" |
| undouble("mississippi") | "misisipi" |
| undouble("carry") | "cary" |
| undouble("juggle") | "jugle" |
| undouble("little") | "litle" |
| undouble("oops") | "ops" |
| undouble("foobar") | "fobar" |
| undouble("apple") | "aple" |
| undouble("berry") | "bery" |
| undouble("theses") | "theses" |
| undouble("") | "" |

You may assume that the string is composed entirely of lowercase letters and that no letter appears more than two times in a row. You are limited to the methods on the cheat sheet in solving this problem. In addition, for this problem only, you may NOT use the replace method of the String class.