# CSE142 Sample Final
Summer 2018

Name ________________________________ Student ID (e.g. 1234567) ____________

Section (e.g., AA) ___________ TA __________________________________________

| # | Problem Area | Points | Score |
|---|---|---|---|
| 1 | Reference Mystery | 5 | _____ |
| 2 | Array Simulation | 10 | _____ |
| 3 | Inheritance | 6 | _____ |
| 4 | File Processing | 9 | _____ |
| 5 | File Processing | 10 | _____ |
| 6 | Array Programming | 10 | _____ |
| 7 | Classes | 10 | _____ |
| 8 | Critters | 15 | _____ |
| 9 | Array Programming | 15 | _____ |
| 10 | Programming | 10 | _____ |
| | Total | 100 | _____ |

This is a closed-book/closed-note exam. Space is provided for your answers. You can also request scratch paper from a TA. You are not allowed to access any of your own papers during the exam. In general the exam is not graded on style and you do not need to include comments, although you are expected to utilize good object-oriented design and respect all special requirements of the Critter class on problem #8.

You are limited to the constructs included in chapters 1 through 10 of the textbook. In particular, you are not allowed to use break statements or to have a return from a void method. You are not allowed to use methods from the Arrays class or other methods that aren't included on the cheat sheet. You are not required to include any import statements; you may assume standard classes are imported. You are allowed to abbreviate "System.out.print" and "System.out.println" as "S.o.p" and "S.o.pln" respectively, but you should otherwise NOT use any abbreviations on the exam.

You are NOT to use any electronic devices while taking the test, including calculators or smart watches. Anyone caught using an electronic device will receive a 10-point penalty.

Do not begin work on this exam until instructed to do so. Any student who makes any modifications to their exam (writing OR erasing) before the exam begins or after time is called will receive a 10-point penalty. If you finish the exam early, please hand your exam to a TA and exit quietly.

Initial here to indicate you have read and agree to these rules rules. If you do not initial, your exam may not receive credit.

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

```
    public class Location {
        int lat;
        int lng;

        public Location(int initialLat, int initialLng) {
            lat = initialLat;
            lng = initialLng;
        }
    }

    public class ReferenceMystery {
        public static void main(String[] args) {
            int n = 4;
            Location location = new Location(16, 23);
            System.out.println(n + " " + location.lat + " " + location.lng);

            mystery(n, location);
            System.out.println(n + " " + location.lat + " " + location.lng);

            n++;
            location.lat = 42;
            mystery(n, location);
            System.out.println(n + " " + location.lat + " " + location.lng);
        }

        public static void mystery(int n, Location location) {
            n = n - 2;
            location.lng += 100;
            System.out.println(n + " " + location.lat + " " + location.lng);
        }
    }
```

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 = 0; i < arr.length; i++) {
            arr[i] = i * arr[i];
        }
    }
```

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
    --------------                      ----------------------------

    []                                  ____________________________

    [7]                                 ____________________________

    [3, 2]                              ____________________________

    [5, 4, 3]                           ____________________________

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

3. Inheritance, 6 points.

```
public class Denny extends John {
    public void method1() {
        System.out.print("denny 1   ");
    }

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

public class Cass {
    public void method1() {
        System.out.print("cass 1   ");
    }

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

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

public class Michelle extends John {
    public void method1() {
        System.out.print("michelle 1 ");
    }
}

public class John extends Cass {
    public void method2() {
        System.out.print("john 2   ");
    }

    public String toString() {
        return "john";
    }
}
```

Given the classes defined, show the exact output produced by the client code below:

```
// client code:
Cass[] elements = {new Cass(), new Denny(), new John(), new Michelle()};
for (int i = 0; i < elements.length; i++) {
    elements[i].method1();
    System.out.println();
    elements[i].method2();
    System.out.println();
    System.out.println(elements[i]);
    System.out.println();
}
```

4. File Processing, 9 points. You may have lost style points this quarter for having lines over 100 characters long. Many organizations enforce that no line of code they produce should be over 80 characters long. Write a method called reportLongLines that takes as parameters a Scanner input and an int maxLength, and reports the lines in the input that are over maxLength characters long, along with the text that goes past the character whose index is maxLength. The method should return the number of lines that were over the length limit. For example, if a Scanner were constructed over the following input file, **in bold:**

```
                     (line length guide not part of input file)
   1   5   10   15   20   25   30   35   40   45   50   55   60   65   70   75
   v   v    v    v    v    v    v    v    v    v    v    v    v    v    v    v
   ---------------------------------------------------------------------------
   This line is 31 characters long
   Meanwhile, this line contains a whopping 54 characters
   13 characters
   This long long long long long long long long line has 67 characters
```

and passed to your method along with the int 40, as in:

```
   int longLines = reportLongLines(input, 40);
   // longLines should now be set to 2
```

your method would produce the following output, and return 2, as there are two lines longer than 40 characters in the input file:

```
   Line 2 excess text: ' 54 characters'
   Line 4 excess text: 'long line has 67 characters'
```

If the method were called with the same Scanner and the integer 80, the method would produce no output and return 0, as no lines in the the input are more than 80 characters long.

5. File Processing, 10 points. Write a static method called formatLib that recreates behavior similar to your MadLibs assignment. This method accepts four parameters: console (Scanner), input (Scanner), output (PrintStream), and maxLength (int).

You are to process the input in the Scanner and print the MadLib output to the output PrintStream. Print each non-placeholder without modification, and when you encounter a placeholder, use the console to prompt for a replacement. You may assume that the user types in a single token when prompted for a replacement.

However, you must also reformat the text as you are printing it: the number of characters printed on each line should not exceed maxLength, and each word should be followed by exactly one space. Note that spaces, including the trailing space on the last word of each line, count towards the maxLength. If any individual word is longer than maxLength, simply print it on its own line.

A placeholder is a single token, surrounded by angle brackets: <proper-noun> . The only modification that you are to perform on the placeholder for the prompt is to remove the angle brackets.

Recall that System.out is a PrintStream object: you can call methods like .print() and .println() on a PrintStream, just as you would System.out.

For example, assume that the following text is in the file "input.txt":

```
<male-name> has      announced that          his
<adjective>
clothing store    in   the  heart of   downtown  <CITY>  is having
a/an <Adjective>   sale       of   all   merchandise!
```

The following code would produce the console interaction and file output below:

```
Scanner console = new Scanner(System.in);
Scanner input = new Scanner("input.txt");
PrintStream output = new PrintStream(new File("output.txt"))
formatLib(console, input, output, 40);
```

Console interaction:
(sample input **bold** and underlined)

```
male-name needed: Jerry
adjective needed: beautiful
CITY needed: Bellevue
Adjective needed: YUUUUUGE
```

output.txt, with line length guide:
(do not print the line length guide)

```
    5   10   15   20   25   30   35   40
    v    v    v    v    v    v    v    v
----------------------------------------

Jerry has announced that his beautiful
clothing store in the heart of downtown
Bellevue is having a/an YUUUUUGE sale
of all merchandise!
```

6. Arrays, 10 points. Write a static method named isConsecutive that accepts an array of ints as a parameter and returns true if the array of integers contains a increasing sequence of consecutive integers and returns false otherwise. Consecutive integers are integers that come one after the other, as in 5, 6, 7, 8, 9, etc.

For example, if a variable called arr stores the following values:

```
    [16, 17, 18, 19]
```

a call on isConsecutive(arr) should return true.

If instead arr stored the following sequence of values:

```
    [16, 17, 18, 19, 20, 19]
```

a call on isConsecutive(arr) should return false.

An array of fewer than two elements is considered to be consecutive, and you may assume that the given array is not null.

7. Classes, 10 points. Suppose that you are provided with a pre-written class ClockTime as described below. Assume that the fields, constructor, and methods shown are implemented. You may refer to them or use them in solving this problem.

```
// A ClockTime object represents an hour:minute time during
// the day or night, such as 10:45 AM or 6:27 PM.
public class ClockTime {
    private int hour;
    private int minute;
    private String amPm;


    // Constructs a new time for the given hour/minute
    public ClockTime(int h, int m, String ap) { /* implementation omitted */ }

    // returns the field values
    public int getHour() { /* implementation omitted */ }
    public int getMinute(){ /* implementation omitted */ }
    public String getAmPm(){ /* implementation omitted */ }

    // returns String for time; for example: "6:27 PM"
    public String toString(){ /* implementation omitted */ }

    // advances this ClockTime by the given # of minutes
    public void advance(int m) { /* implementation omitted */ }


    // your method will go here
}
```

Write an instance method named isWorkTime that will be placed inside the ClockTime class to become a part of each ClockTime object's behavior. The isWorkTime method returns true if the ClockTime object represents a time during the normal "work day" from 9:00 AM to 5:00 PM, inclusive. Any times outside that range would cause the method to return a result of false.

For example, if the following object is declared in client code:

```
ClockTime t1 = new ClockTime(3, 27, "PM");
```

The following call to your method would return true:

```
System.out.println(t1.isWorkTime());    // true
```

Here are some other objects. Their results when used with your method are shown at right in comments:

```
ClockTime t2 = new ClockTime(12, 45, "AM"); //false
ClockTime t3 = new ClockTime( 6, 02, "AM"); //false
ClockTime t4 = new ClockTime( 8, 59, "AM"); //false
ClockTime t5 = new ClockTime( 9, 00, "AM"); //true
ClockTime t6 = new ClockTime(11, 38, "AM"); //true
ClockTime t7 = new ClockTime(12, 53, "PM"); //true
ClockTime t8 = new ClockTime( 3, 15, "PM"); //true
ClockTime t9 = new ClockTime( 4, 59, "PM"); //true
ClockTime ta = new ClockTime( 5, 00, "PM"); //true
ClockTime tb = new ClockTime( 5, 01, "PM"); //false
ClockTime tc = new ClockTime( 8, 30, "PM"); //false
ClockTime td = new ClockTime(11, 59, "PM"); //false
```

Your method should not modify the state of the ClockTime object. Assume that the state of the ClockTime object is valid at the start of the call and that the amPm field stores either "AM" or "PM".

8. Critters, 15 points. Write a class Minnow that extends Critter from HW8, along with its movement and eating behavior. All other aspects of Minnow use the defaults. Add fields, constructors, etc. as necessary to your class.

Minnow objects initially move in a S/E/S/E/... pattern. However, when a Minnow encounters food (when its eat method is called), it should do all of the following:

* Do not eat the food.
* Start the movement cycle over. In other words, the next move after encountering food should always be South.
* Lengthen and reverse the horizontal portion of the movement cycle pattern. The Minnow should reverse its horizontal direction and increase its horizontal movement distance by 1 for subsequent cycles. For example, if the Minnow had been moving S/E/S/E, it will now move S/W/W/S/W/W. If it hits a second piece of food, it will move S/E/E/E/S/E/E/E, and a third, S/W/W/W/W/S/W/W/W/W, and so on.

The following is an example timeline of a particular Minnow object's movement.

* S, E, S, E (hits food)
* S, W, W, S, W, W, S (hits food)
* S, E, E, E, S, E, E, E, S, E (hits food)
* S (hits food)
* S, E, E, E, E, E, S, E, E, E, E, E, ...

9. Arrays, 15 points. Write a static method named delta that accepts an array of integers as a parameter and returns a new array formed by inserting between each pair of values the difference between those values. For example, given this array:

```
    int[] numbers = {3, 8, 15};
```

the call of delta(numbers) should return the following array (new elements are bolded):

{3, **5**, 8, **7**, 15}

In this example, 5 is inserted between 3 and 8 because (8 - 3) is 5, and 7 is inserted between 8 and 15 because (15 - 8) is 7. The difference should always be computed as the second value minus the first, so you can get negative values. For example, given the following array:

```
    int[] numbers2 = {3, 8, 2, 5, 1, 9};
```

The call of delta(numbers2) should return the following array (new elements are bolded):

{3, **5**, 8, **-6**, 2, **3**, 5, **-4**, 1, **8**, 9}

You may assume that the array passed is not null. If the array contains fewer than two elements, the resulting array should contain the same elements as the original array.

You are allowed to create one array during the execution of this method; you may not use any other auxiliary data structures (more arrays, ArrayLists, etc.).

10. Programming, 10 points. Write a static method called printReversed that takes a String as a parameter and that prints a line of output with each word in the String reversed. For this problem, we will define words as nonempty sequences of characters separated by one or more spaces. For example, if we make the following calls:

```
    printReversed("four score and seven years ago");
    printReversed("our fathers brought forth on this continent");
```

We should get the following output:

```
    ruof erocs dna neves sraey oga
    ruo srehtaf thguorb htrof no siht tnenitnoc
```

Notice that the words appear in the same order as in the Strings that were passed as parameters, but each individual word is reversed ("four" has become "ruof", "score" has become "erocs", and so on). The String you are passed might have leading or trailing spaces, which should be printed exactly as they appear in the String. For example, if we make the following call:

```
    printReversed("   merry-go-round      is     one   word  ");
```

the following output should be produced:

```
      dnuor-og-yrrem      si     eno   drow
```

This output has 3 spaces at the beginning of the line and 2 spaces at the end of the line, just as in the String. You are allowed to call the toCharArray method on the String to convert the entire String into a char[] if you prefer to use array notation rather than String notation in solving the problem.