Name: _____ Section: _____ CM: _____

CSSE 220—Object-Oriented Software Development

Final Exam – Part 1, February 19, 2018

This exam consists of two parts. Part 1 is to be solved on these pages. You may use the back of a page if you need more room. Please indicate on the front if you do so. Part 2 is to be solved on your computer. You will need network access to download template code and upload your solution for part 2. Please disable IM, email, Lync, and other such communication programs before beginning the exam.

Resources for Part 1: One 8 1/2" by 11" double-sided sheet of notes plus the UML Cheatsheet and the OO Principles pages linked from the course schedule. Closed book, closed computer, closed electronic devices.

Resources for Part 2: Open book, open notes, and open computer. Limited network access. You may use the network only to access your own files and Moodle sites, the course web pages, the textbook's site, and Oracle's Java website (including API documentation).

Any communication with anyone other than an exam proctor during the exam may result in a failing grade for the course.

Part 1 is included in this document. You should read over all of the questions before beginning work, but...

You must turn in part 1 before accessing the resources for part 2.

Problem	Poss. Pts.	Earned
1	6	_____
2	8	_____
3	6	_____
4	6	_____
5	9	_____
Paper Part Subtotal	**35**	_____
Computer Part Subtotal	**65**	_____
Total	100	_____

Part 1—Paper Part

1. (6 points) Consider the following initial array configuration. In each question, assume we want to sort the array in ascending order (1 is less than 2, 2 is less than 3, etc.).

| 9 | 5 | 7 | 10 | 18 | 1 | 12 | 8 | 16 | 4 |

a. (3 points) Suppose the **insertion** sort algorithm is applied to the **initial array above**. Show the state of the array immediately following **each of the first three** iterations of the outer loop. Please **clearly mark (as in 0th)** the sorted part of the array after each iteration.

0st iteration:

| 9 ‖ 5 | 7 | 10 | 18 | 1 | 12 | 8 | 16 | 4 |

1st iteration:

| 5 | 9 | 7 | 10 | 18 | 1 | 12 | 8 | 16 | 4 |

2nd iteration:

| 5 | 7 | 9 | 10 | 18 | 1 | 12 | 8 | 16 | 4 |

3rd iteration:

| 5 | 7 | 9 | 10 | 18 | 1 | 12 | 8 | 16 | 4 |

1 pt each item

−1 if they don't include dividing marker (but not more than −1 total)

b. (1 point) Suppose the **merge** sort algorithm from class is applied to the **initial array above**. Show the state of the two sub-arrays immediately before the final merge.

| 5 | 7 | 9 | 10 | 18 | | 1 | 4 | 8 | 12 | 16 |

exact no partial credit

c. (2 points) Suppose the **selection** sort algorithm from class is applied to the **initial array above**. In the boxes below, show the state of the array immediately following **each of the first two** iterations of the outer loop. Please **clearly mark** the sorted part and the unsorted part of the array after each iteration. (Initially the sorted part of the array is empty.)

1st iteration:

| 9 | 5 | 7 | 10 | 18 | 1 | 12 | 8 | 16 | 4 |

2nd iteration:

| 1 | 5 | 7 | 10 | 18 | 9 | 12 | 8 | 16 | 4 |

3rd iteration:

| 1 | 4 | 7 | 10 | 18 | 9 | 12 | 8 | 16 | 5 |

award full credit even if lists treated as linked lists with shifts. Must put smallest at front.

(Do not take off points for missing marker if they already lost above)

2

2. (8 points) Answer the questions below on the sorting, searching and data structures that we discussed in class. Assuming that you are given an ArrayList of **unsorted** CSSE 220 student ids (integer), what is the runtime (Big-O) for each algorithms below, write the Big-O runtime in the **worst** case. In each case "n" refers to the length of the array.

You want to add a new student to the start of the arraylist: $O(n)$

One of the student's id got messed up, so you need to replace that id with the correct one (assume that you are given the arraylist, incorrect id, and correct id): $O(n)$

You need to sort the arraylist in ascending order; it currently has student ids in no particular order:

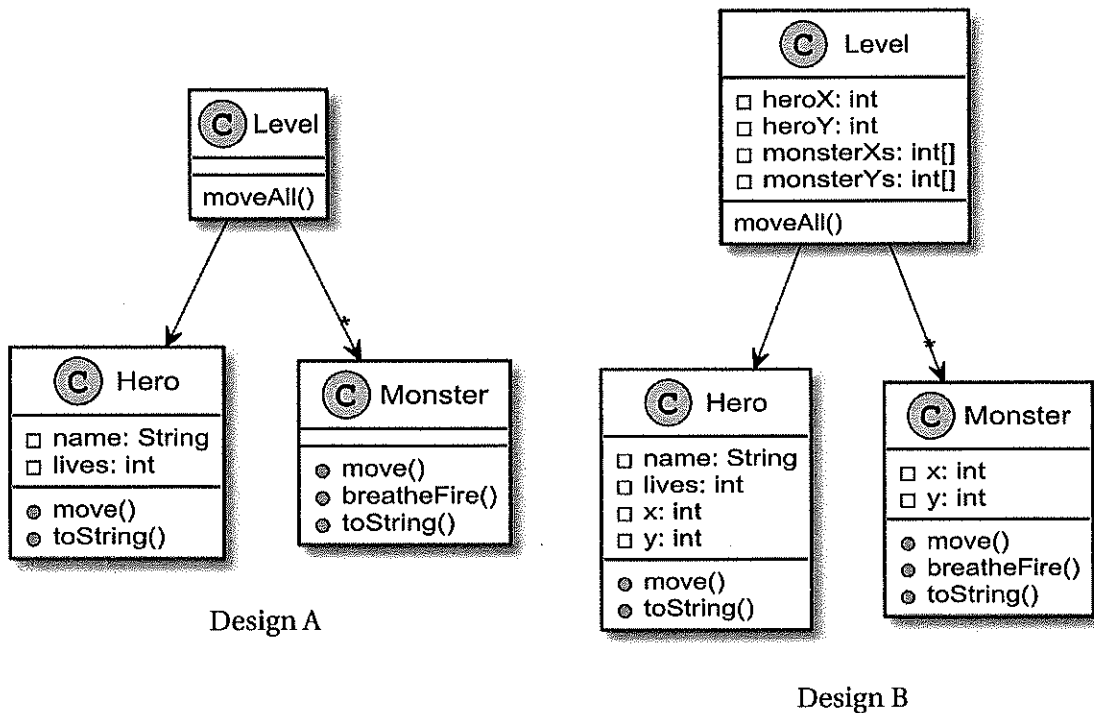Insertion sort: $O(n^2)$, Merge sort: $O(n \cdot \log(n))$

Now that you have the sorted arraylist, we want to find a student with the given id:

Brute force array search using a for loop: $O(n)$, Binary search: $O(\log(n))$

Now we have a transfer student who is just enrolled in CSSE 220 and was appended to the arraylist. She has the highest id by chance (so the arraylist is still sorted), but you are still asked to sort the arraylist again:

Insertion sort: $O(n)$, Merge sort: $O(n \cdot \log(n))$

if students just write n, mark $O(n)$ but do not take away any points.

3

Design A

Level
moveAll()

Hero
- name: String
- lives: int
- move()
- toString()

Monster
- move()
- breatheFire()
- toString()

Design A

Design B

Level
- heroX: int
- heroY: int
- monsterXs: int[]
- monsterYs: int[]
moveAll()

Hero
- name: String
- lives: int
- x: int
- y: int
- move()
- toString()

Monster
- x: int
- y: int
- move()
- breatheFire()
- toString()

Design B

3. (6 points)

In a particular video game, there are pieces called heroes and monsters. Heroes and monsters move differently from each other. The move is a complex determination which requires specific data that is updated over the monster and hero's lifetime. The game must support giving each piece the opportunity to move.

a. (3 points) Identify the problem in Design A that Design B is attempting to fix. Please identify one problem and give at least one sentence (and no more than 3) explaining the details of the issue you have identified. *Answer*

> 1.a. Design does not function. The position of Heros and monsters is not stored.

b. (3 points) Identify the main problem in Design B. Please give at least one sentence (and no more than 3) explaining the details of the issue you have identified. (If you see more than one problem, feel free to identify and describe more than one but we will be looking for one main problem.) *Answer*

> 1.c. Data should not be duplicated. Level holds information that is in Hero and Monster. Hero and Monster correctly store their own position, Level should not.

Rubrik
If they correctly identify the problem 3/3 even if they give other problems as well. If they give additional <u>incorrect</u> *items you may take (−1).*

4. (6 points) Give the Big–O runtime for each of the code snippets below. In each case, n refers to the size of the input list. Answers are worth 2 points each.

a.
```
public static void function1(int[] array) {
    for(int i = 0; i < 100; i++) {
        for(int j = 0; j < array.length; j++) {
            array[j] += i;
        }
    }
}
```

Answer:

$O(n)$

b.
```
public static void function2(int[] array) {
    for(int i = 0; i < array.length; i++) {
        for(int j = 0; j < i; j++) {
            if(array[i] != array[j]) {
                array[j] = array[i];
            }
        }
    }
}
```

Answer:

$O(n^2)$

c.
```
public static void function3(int[] array) {
    for(int i = 1; i <= array.length; i++) {
        for(int j = array.length; j >= 1; j = j / 2) {
            if(array[i-1] <= array[j-1]) {
                array[j-1] = array[i-1];
            }
        }
    }
}
```

Answer:

$O(n \cdot \log(n))$

rubric   do not take off if they
only write n, n^2, $n\log n$, but
you can write in $O(n)$ ...

5

5. (9 points) Use the code below to answer the associated questions.

```
interface Calc {
    double add(double a, double b);
    double sub(double a, double b);
}

abstract class ACalc implements Calc {
    public double add(double a, double b) {
        return a + b;
    }
}

class WeirdCalc implements Calc {
    public double add(double a, double b) {
        return a - b + (a + b) - a + b;
    }

    public double sub(double a, double b) {
        return a - b;
    }

    public double mul(double a, double b) {
        return a * b;
    }
}
```

```
class SimpleCalc extends ACalc {
    public double sub(double a, double b) {
        return a - b;
    }
}

class CoolCalc implements Calc {
    Calc first, second;

    public CoolCalc(Calc first, Calc second) {
        this.first = first;
        this.second = second;
    }

    public double add(double a, double b) {
        return first.add(a, b);
    }

    public double sub(double a, double b) {
        return second.sub(a, b);
    }

    public double mul(double a, double b) {
        return a * b;
    }
}
```
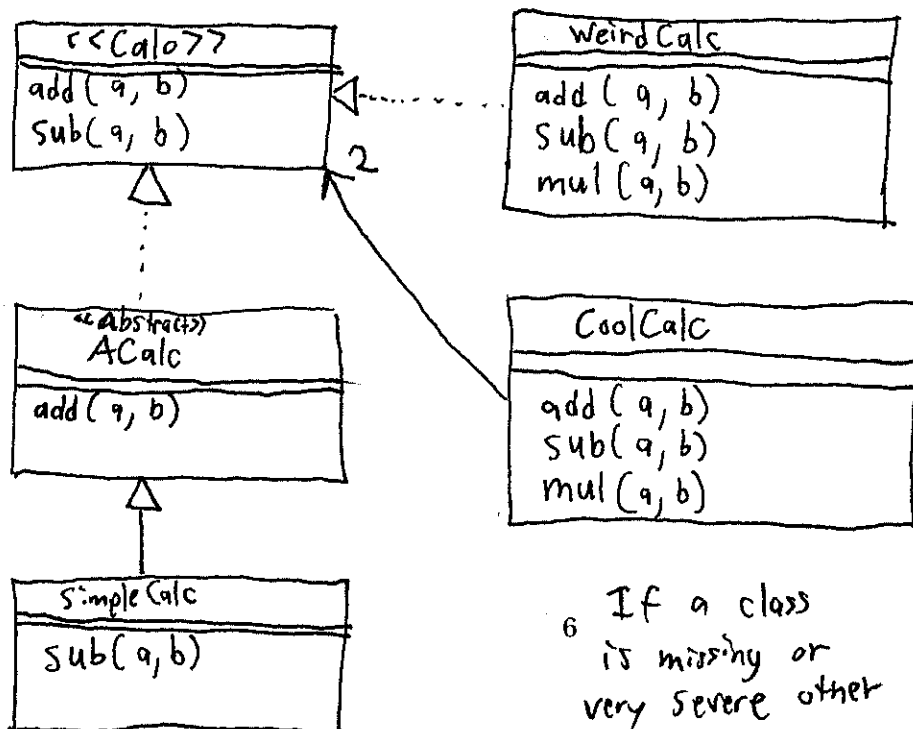
a. (3 points) Draw a UML class diagram that shows the relationships among the above classes.

b. (6 points) Using the declarations from the previous classes: Consider each of the code snippets independently (that is, errors and declarations in one do not affect the others). For each, write the output. If the code snippet results in an error, indicate the type of error (either Compile-time or Runtime error). If there is no output, write "No output".

i
```
Calc c = new SimpleCalc();
System.out.println(c.add(10, 10));
```
_20_

ii
```
ACalc c1 = new WeirdCalc();
System.out.println(c1.sub(20, 10));
```
Compile error

iii
```
CoolCalc c2 = new CoolCalc(
    new SimpleCalc(),
    new WeirdCalc());
System.out.println(c2.add(20, 10));
```
_30_

iv
```
Calc c3 = new SimpleCalc();
System.out.println(
    ((WeirdCalc)c3).mul(10, 10));
```
run time error

v
```
Calc c4 = new ACalc();
System.out.println(c4.add(20, 10));
```
Compile error

vi
```
Calc c5 = new WeirdCalc();
System.out.println(
    ( (WeirdCalc)c5).mul(10, 10));
```
_100_

rubric   1 pt each   no partial credit