

Name: \_\_\_\_\_

# CSSE 220—Object-Oriented Software Development

## Final Exam, Fall 2013

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 using your computer. You will need network access to download template code and upload your solution for part 2. Please disable IM, email, and other such communication programs before beginning the exam.

*Resources for Part 1:* One sheet of notes, closed book, closed computer.

*Resources for Part 2:* Open book, notes, and computer. Limited network access. You may use the network only to access your own files, the course ANGEL site and web pages, the textbook's site, Sun's Java website, and Logan Library's Safari Tech Books Online. Any communication with anyone other than the instructor or a TA during the exam may result in a failing grade for the course.

Parts 1 and 2 are 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	8	_____
2	8	_____
3	6	_____
4	10	_____
5	9	_____
6	10	_____
<b>Paper Part Subtotal</b>	<b>51</b>	_____
C1. LinkedList	10	_____
C2. Recursion	12	_____
C3. Multithreading	8	_____
C4. Subclasses	19	_____
<b>Computer Part Subtotal</b>	<b>49</b>	_____
Total	100	_____

## Part 1—Paper Part

1. (8 points) Consider the following initial array configuration. In each question, assume we want to sort the array in **ascending** order (that is, from smallest to largest).

4	5	1	7	3	2	8	6
---	---	---	---	---	---	---	---

- a. (3 points) Suppose the **insertion** sort algorithm from class is applied to the **initial array above**. Show the state of the array immediately following the first **three** executions of the outer loop.

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

- b. (3 points) Suppose the **selection** sort algorithm from class is applied to the **initial array above**. Show the state of the array immediately following the first **three** executions of the outer loop.

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

- c. (2 points) 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.

--	--	--	--

--	--	--	--

2. (8 points) For each of the scenarios below, indicate which of the *data structures* we studied—**array list, linked list, stack, queue, hash set, tree set, hash map, or tree map**—would be best. Justify your answer.

- a. Modeling my personal todo list, where every item has a priority and I always want it to remain sorted (also, all items are unique)

Data Structure: \_\_\_\_\_

Justification:

- b. Maintaining a list of files to process, where newly added files need to wait for all the files currently in the list to finish processing before they can be started

Data Structure: \_\_\_\_\_

Justification:

- c. Keeping track of student birthdays, where I want to easily be able to look up a birthday given a student name

Data Structure: \_\_\_\_\_

Justification:

- d. An ordered list of elements that I rarely add or remove elements to, but often need to access by index (e.g. the 5th list element, the 234th element, etc.)

Data Structure: \_\_\_\_\_

Justification:

3. (6 points) Predict the output of each of the following code blocks:

a.

```
Stack<Integer> s = new Stack<Integer>();
s.push(1);
s.push(2);
System.out.print(s.pop());
System.out.print(s.pop());
s.push(3);
System.out.print(s.pop());
s.push(4);
s.push(5);
System.out.print(s.pop());
System.out.print(s.pop());
```

Answer: \_\_\_\_\_

b.

```
Queue<Integer> s = new Queue<Integer>();
// actually Java's queue uses slightly different
// methods but these are the ones from class
// just in case you forgotten enqueue = offer
// dequeue = poll
s.enqueue(10);
s.enqueue(9);
System.out.print(s.dequeue());
System.out.print(s.dequeue());
s.enqueue(8);
System.out.print(s.dequeue());
s.enqueue(7);
s.enqueue(6);
System.out.print(s.dequeue());
System.out.print(s.dequeue());
```

Answer: \_\_\_\_\_

c.

```
HashMap<Character,String> map =
    new HashMap<Character, String>();

map.put('x', "Ninja");
map.put('y', "Pirate");
map.put('z', "Robot");

System.out.print(map.get('z'));
System.out.print(map.get('x'));
```

Answer: \_\_\_\_\_

4. (10 points) Write the Big-O of each of the following code blocks:

a.

```
public void fun1(int n) {  
    if(n % 7 == 0)  
        return;  
    for(int i = 0; i < n; i++) {  
        for(int j = 0; j < n; j++) {  
            System.out.println("fun1");  
        }  
    }  
}
```

Answer: \_\_\_\_\_

b.

```
public void fun2(int n) {  
    for(int i = 0; i < n; i++) {  
        for(int j = 0; j < n-i; j++) {  
            System.out.println("fun2");  
        }  
    }  
}
```

Answer: \_\_\_\_\_

c.

```
// n is the number of elements in the list  
public int fun3(LinkedList<Integer> list) {  
    int n = list.size();  
    //returns the middle element  
    return list.get(n/2);  
}
```

Answer: \_\_\_\_\_

d.

```
// n is the number of elements in the array  
public int fun4(int[] list) {  
    for(int i = 0; i < list.length; i++) {  
        doABinarySearch(list, 44);  
    }  
}
```

Answer: \_\_\_\_\_

e.

```
public void fun5(int n) {
    int data = n;
    while(data > 3) {
        data = data/2;
        System.out.println("Foo!");
    }
}
```

Answer: \_\_\_\_\_

5. (9 points) What does the following code print?

a.

```
fun6("abc");

// elsewhere...
public void fun6(String input) {
    if(input.isEmpty())
        return;
    char c = input.charAt(0);
    System.out.print(c);
    fun6(input.substring(1));
    System.out.print(c);
}
```

Answer:

b.

```
int[] data = {1,2,6,7,8,9};
int[] result = fun7(data, 0);
for(int i = 0; i < result.length; i++) {
    System.out.print(result[i] + ", ");
}

//elsewhere...
public int[] fun7(int[] data, int i) {
    int a = data[i];
    if(a == 6) return data;
    if(i > data.length/2) return data;

    data[i] = data[data.length - 1 - i];
    data[data.length - 1 - i] = a;
    return fun7(data, i+1);
}
```

Answer:

c.

```
System.out.println(fun8(28));

//elsewhere...
public int fun8(int input) {
    int min = input;
    if(input > 9 && input % 3 == 0) {
        min = Math.min(min, fun8(input / 3));
    }

    if(input > 10) {
        min = Math.min(min, fun8(input - 7));
    }
    return min;
}
```

Answer:

6. (10 points) Consider the following related declarations:

```
class A {  
    public void number()  
    {  
        System.out.println(1);  
    }  
  
    public void letter()  
    {  
        System.out.println("a");  
    }  
}  
  
class B extends A {  
    public void letter()  
    {  
        System.out.println("b");  
    }  
  
    public void letter2()  
    {  
        System.out.println("B");  
    }  
}
```

```
class C extends A { }  
  
abstract class D extends C {  
    abstract public void l3();  
}  
  
class E extends D {  
    public void l3()  
    {  
        System.out.println("E");  
    }  
}
```



Suppose we declare and initialize these variables:

```
A a = new A();
A a2 = new B();
B b = new B();
C c = new C();
```

For each line of code below, **circle** what would be output. If the line would work but not output anything, circle nothing. If a line would give an error, then circle the type of error. Consider each line separately. That is, if a line would give an error, then assume that line doesn't affect any others.

Code	Output Choices (circle one in each line)
a2.letter();	a   b   B   1   E <i>nothing</i> <i>runtime error</i> <i>compile error</i>
a2.letter2();	a   b   B   1   E <i>nothing</i> <i>runtime error</i> <i>compile error</i>
C c1 = new B();	a   b   B   1   E <i>nothing</i> <i>runtime error</i> <i>compile error</i>
C c2 = new A();	a   b   B   1   E <i>nothing</i> <i>runtime error</i> <i>compile error</i>
C c3 = (C) new A();	a   b   B   1   E <i>nothing</i> <i>runtime error</i> <i>compile error</i>
b.number();	a   b   B   1   E <i>nothing</i> <i>runtime error</i> <i>compile error</i>
a = b;	a   b   B   1   E <i>nothing</i> <i>runtime error</i> <i>compile error</i>
c.letter2();	a   b   B   1   E <i>nothing</i> <i>runtime error</i> <i>compile error</i>
D d = new D(); d.l3();	a   b   B   1   E <i>nothing</i> <i>runtime error</i> <i>compile error</i>
D d2 = new E(); d2.l3();	a   b   B   1   E <i>nothing</i> <i>runtime error</i> <i>compile error</i>

## Part 2—Computer Part

*Resources for Part 2:* Open book, notes, and computer. Limited network access. You may use the network only to access your own files, the course ANGEL site and web pages, the textbook's site, Sun's Java website, and Logan Library's Safari Tech Books Online. Any communication with anyone other than the instructor or a TA during the exam may result in a failing grade for the course.

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

**Instructions.** You must actually get these problems working on your computer. Almost all of the credit for this problem will be for code that actually works. Comments are not required. If you do not get a method to work, comments may help me to understand enough so I can give you (a small amount of) partial credit.

After you have handed in part 1, **begin part 2 by checking out the project named *FinalExam* from your course SVN repository.** (Ask for help immediately if you are unable to do this.)

When you have finished the problems, and more frequently if you wish, you should **submit your code by committing it to your SVN repository.** We will check commit logs, so you must be careful not to commit anything after the end of the exam. For grading, we will ensure that the included JUnit tests have not been changed.

---

### Problem Descriptions

C1. (10 points) *Linked List Questions.* Implement the functions `addSecond(String newData)` and `uppercase()` in the simple linked list class `StringLinkedList.java`. Note that tests are provided.

C2. (12 points) *Recursion Questions.* Implement the three recursion questions in `Recursion.java`. Your solutions must be recursive to get the majority of the credit. Note that tests are provided.

C3. (8 points) *Multithreading Question*. Oftentimes when designing webpages it's a good idea to ensure that they hold up when many visitors try to visit at once. This can be a little tricky though, because you usually only have one computer and you can only click "reload" so fast. One solution is to write a little program to hit the webpage quickly.

If you run the program in WebpageLoader.java you'll see that it hits the same webpage 10 times, reporting on its speed each time. But there's a problem – it waits for the webpage to load completely before it tries again. This is not a good simulation — instead we want to load 10 different requests as quickly as possible to simulate heavier load.

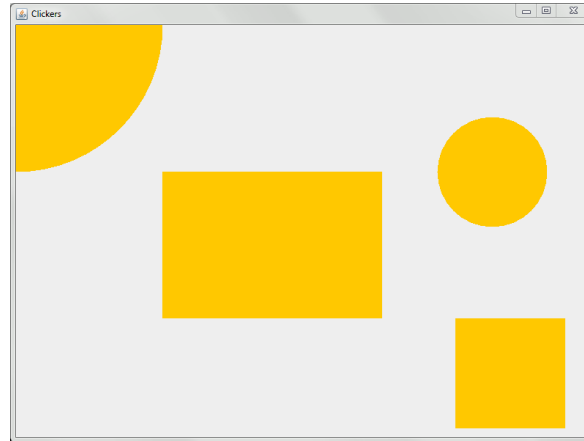
Write the function `doParallelTest()` in webpage loader to use Threads to load 10 webpages at the same time. If you do it correctly, your console ought to look something like this:

```
Doing parallel test...(10 pages at the same time)
```

```
loading webpage  
loading webpage  
loading webpage  
loading webpage  
loading webpage  
loading webpage  
loading webpage  
loading webpage  
loading webpage  
loading webpage  
loaded 84 characters in 0.20800 seconds  
loaded 84 characters in 0.41100 seconds  
loaded 84 characters in 0.41200 seconds  
//etc..
```

C4. (19 points) *Subclassing Question*. In this problem we have objects we call *clickers*. Clickers are objects that know how to do two things: draw themselves and detect if they are clicked on. In this question you'll both implement some existing clicker methods and add some of your own.

- a. (9 points) Take a look at the clicker classes. In particular, focus on ClickerComponent and the Clicker abstract superclass. You can see that ClickerComponent already has code for drawing clickers in its paintComponent method. Go ahead and write the code in RectangleClicker and CircleClicker so that they draw correctly. If you do it properly, you should see a window like this:



- b. (10 points) Now our clicker objects need to handle mouse clicks. The code to determine if a particular mouseclick occurred within the confines of a particular Clicker should be within the Clicker objects themselves. The mouseClicked method of ClickerComponent will detect the click, and then call some method you create (passing parameters so that the clicker can determine if it was clicked on or not).

Modify the Clicker superclass so that Clickers can do this. Modify RectangleClicker and CircleClicker to implement the methods you added correctly (e.g. a circle clicker should only register a click if you click on the actual circle). Then update the mouseClicked method of ClickerComponent to call your new methods when a mouse click is detected in the window. When a clicker detects that it has been clicked on, it should print a message that looks something like "You clicked on a Circle clicker!".