

# CSSE 220—Object-Oriented Software Development

Exam 2, Jan. 25th, 2017

This exam consists of two parts. Part 1 is to be solved on these pages. There is an additional blank page at the end of Part 1 if you need more room. Part 2 is to be solved using your computer, and will be taken on Friday. You will need network access to download template code and upload your solution for part 2.

*Resources for Part 1:* You may use a single sheet of  $8\frac{1}{2} \times 11$  inch paper with notes on both sides. Your computer *must be closed* the entire time you are completing Part 1.

*Resources for Part 2:* This portion is open book, notes, and computer but with limited network access. You may use the network only to access your own files, the course Moodle site and web pages, Piazza (though do not post/respond to questions), the textbook's site, Oracle'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.

When you have finished Part 1, please turn it in and then wait quietly for the programming exam review portion of class to begin.

	<b>Problem</b>	<b>Poss. Pts.</b>	<b>Earned</b>
	1	5	_____
	2	10	_____
	3	9	_____
	4	4	_____
	5	16	_____
	6	12	_____
	<b>Paper Part Subtotal</b>	<b>56</b>	_____
	C1. Recursion problems	21	_____
	C2. Polymorphism problem	6	_____
	C3. GUI problem	17	_____
	<b>Computer Part Subtotal</b>	<b>44</b>	_____
	Total	100	_____

## Part 1—Paper Part

1. (5 points)

The following sentences each describe the design of a different object oriented system. Label each of them with one of the following phrases that best match (*you'll use each phrase exactly once*): High Coupling, Low Coupling, High Cohesion, Low Cohesion, and None (for the description that isn't really describing coupling or cohesion).

\_\_\_\_\_ You create a TelephoneNumber class that provides functions for storing the phone number, formatting it for display, and noting the type of number (home, cell, etc.).

\_\_\_\_\_ The CrazyMonster class of your game is used by only one other class in your system and depends on nothing else.

\_\_\_\_\_ You've been given an existing code base to update and notice that the same code, with minor edits to variable names, exists in 4 different classes.

\_\_\_\_\_ You notice a class named PictureOrText which can handle text or image display, depending on how it is initialized.

\_\_\_\_\_ All classes in your project have fields of type MonsterManager and GameWorld.

2. (10 points) This problem is a design exercise. First, read the problem description below. Then answer the questions.

On a example photo sharing website, users can “like” uploaded photos. Pictures that get a lot of likes are displayed on the homepage. Because this is competitive, it's important to the design of the system that a single user can only "like" a particular picture once - otherwise a motivated user can cheat the system by liking one picture hundreds of times.

All user interaction is handled by a class called PictureMain. Here are three methods that are in the PictureMain class that your design must support:

getLikeCount(pictureid) - returns a number of likes

getAllLikedPictures(userid) - returns a list of pictureids

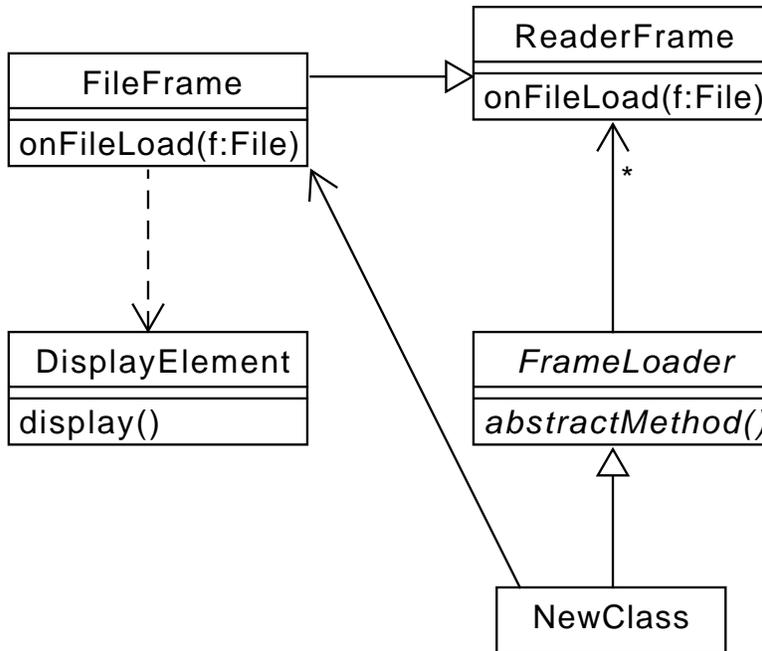
likePicture(userid, pictureid) - returns true/false if it was successful

a. (5 points) Draw a UML class diagram showing how you would design this system. You do not need to include every method or field - just the important ones. Your design should accommodate the features mentioned above and adhere to the design principles we have discussed in class. Do NOT worry if your design contains data classes – this example is a little too small to easily avoid them.

b. (5 points) Describe in a few sentences how the likePicture command is implemented, referencing your UML diagram. Make sure it's clear how the system prevents a user from liking the same picture more than once. Be sure to note what methods are called by other methods, and the fields of your classes that are updated.

Although you do NOT need to describe how getLikeCount and getAllLikedPictures are implemented, be sure your system stores data so that these other methods can work in a reasonable way.

3. (9 points) Use this UML class diagram to answer the subsequent questions.



a. (5 points) Circle true or false for each of the following statements.

- T F FileFrame likely has an ArrayList of ReaderFrame objects as a field
- T F FileFrame inherits from ReaderFrame
- T F ReaderFrame must implement abstractMethod or the code will not compile
- T F FileFrame overrides the method onFileLoad
- T F DisplayElement's superclass is Object

b. (2 points) Which of these could explain the line between FileFrame and DisplayElement?

- (a) FileFrame has a field of type DisplayElement
- (b) FileFrame's onFileLoad method uses a DisplayElement as a local variable
- (c) DisplayElement's display() method uses a FileFrame as a local variable
- (d) FileFrame implements the interface DisplayElement
- (e) None of the above could explain the relationship

c. (2 points) Assuming you wanted to add the non-abstract NewClass as depicted in the diagram, what fields and methods would NewClass have to add?

4. (4 points) Consider the following code.

```
public static int getNumber() throws InterruptedException {
    //code that might throw InterruptedException
    System.out.println("returning 3");
    return 3;
}

public static int getAndIncreaseNumber() throws InterruptedException {
    System.out.println("starting increase");
    int retVal = 0;
    retVal = getNumber();
    System.out.println("returning " + (retVal + 1));
    return retVal + 1;
}

public static void main(String[] args) {
    try {
        int finalVal = getAndIncreaseNumber();
        System.out.println("final value " + finalVal);
    } catch (InterruptedException e) {
        System.out.println("catch 2");
    }
    System.out.println("done.");
}
```

Assuming a `InterruptedException` is thrown in `getNumber()`, what would this code print?

5. (16 points) Consider the following related declarations:

```
public interface Int {
    void thing();
    void value();
}

public class First implements Int {
    public void thing() {
        System.out.println("First");
    }
    public void value() {
        System.out.println("One");
        this.thing();
    }
}

public class Second extends First {
    public void thing() {
        System.out.println("Second");
    }
    public void stuff() {
        System.out.println("Last");
        super.thing();
    }
}
```

```
public abstract class Third
implements Int{

    public void thing() {
        System.out.println("Second");
        this.value();
    }
    public abstract void value();
}

public class Fourth extends Third {
    public void value() {
        System.out.println("Four");
    }
    public void item() {
        System.out.println("Quad");
    }
}
```

- a. (4 points) Draw a UML diagram to represent the given interface and classes. Include all methods, but when writing subclass methods, only show a method on the subclass if the subclass method overrides the parent class's method, or if the method is specific only to the subclass:

- b. (12 points) Continuing the same problem, suppose we declare and initialize these variables:

```

Int i = new First();
Int j = new Second();
Int k = new Fourth();
Third t = new Fourth();
Fourth h = new Fourth();

```

For each line of code below, if the line results in an error, **circle** the appropriate error; otherwise, provide the output in the provided blank. If the code works but does not print anything, write “nothing”. Consider each line of code separately. That is, if a line would give an error, then assume that line doesn’t affect any others. If the result would print on multiple lines, remove the newline from your result and show it on a single line.

Code	Either circle the error or provide the output	
Int l = new Third();	<i>runtime error</i>	<i>compile error</i> _____
k.thing();	<i>runtime error</i>	<i>compile error</i> _____
h.item();	<i>runtime error</i>	<i>compile error</i> _____
Int r = new Int();	<i>runtime error</i>	<i>compile error</i> _____
((Second)i).thing();	<i>runtime error</i>	<i>compile error</i> _____
h.thing();	<i>runtime error</i>	<i>compile error</i> _____
((Second)j).stuff();	<i>runtime error</i>	<i>compile error</i> _____
i.thing();	<i>runtime error</i>	<i>compile error</i> _____
((Fourth)j).item();	<i>runtime error</i>	<i>compile error</i> _____
Second s = new First();	<i>runtime error</i>	<i>compile error</i> _____
j.thing();	<i>runtime error</i>	<i>compile error</i> _____
t.value();	<i>runtime error</i>	<i>compile error</i> _____

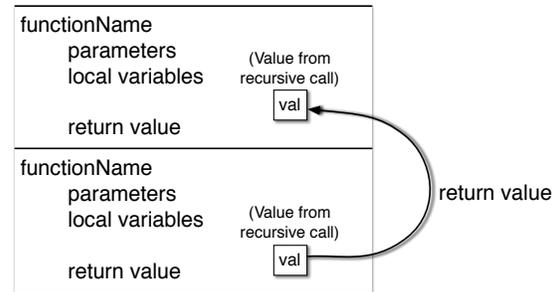
6. (12 points) For this problem, use the frame technique we practiced in the course to trace the execution of the recursive function call. Start your trace with the first call to `mystery` on line 14. A frame template is provided for your reference.

Once you are finished, answer the question at the bottom of the page.

```

1 private static String mystery(String input) {
2     if (input.length() == 0) {
3         return "";
4     }
5     char first = input.charAt(0);
6     String rest = input.substring(1);
7     if (Character.isLowerCase(first)) {
8         return Character.toUpperCase(first) + mystery(rest);
9     }
10    return Character.toLowerCase(first) + mystery(rest);
11 }
12
13 public static void main(String[] args) {
14     System.out.println(mystery("AbC"));
15 }

```



For the code above, *what would the final output be?* \_\_\_\_\_

Use this page for additional workspace if you need it.