

Name: KEY Section: _____

CSSE 220—Object-Oriented Software Development

Exam 2, October 23, 2015

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. 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: 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.

You must turn in Part 1 to receive a copy of the Part 2 description, and only then may you open your computer.

Problem	Poss. Pts.	Earned
1	5	_____
2	10	_____
3	9	_____
4	4	_____
5	9	_____
6	16	_____
7	12	_____
Paper Part Subtotal	65	_____
C1. Recursion problems	15	_____
C2. Polymorphism problem	5	_____
C3. Adder problem	15	_____
Computer Part Subtotal	35	_____
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).

None Two classes share a common superclass, and so you move a method they both have duplicated into that superclass.

Low cohesion You notice a class called JButtonFileIOJFrameHanlder, which deals with responding to button presses, loading and saving files, and building three different GUI windows with complicated layouts.

High coupling When modifying the code in the NetworkProtocolHandler, you have to remember to also modify the 12 other classes that depend on it.

Low coupling Another team makes a large number of changes to a system you both share, but your code does not need to be updated because you only interact using a few well-established interfaces.

High cohesion Correctly parsing email addresses requires several complicated methods, so you decide to make an EmailAddress class that encapsulates the parsing rather than storing email addresses as Strings.

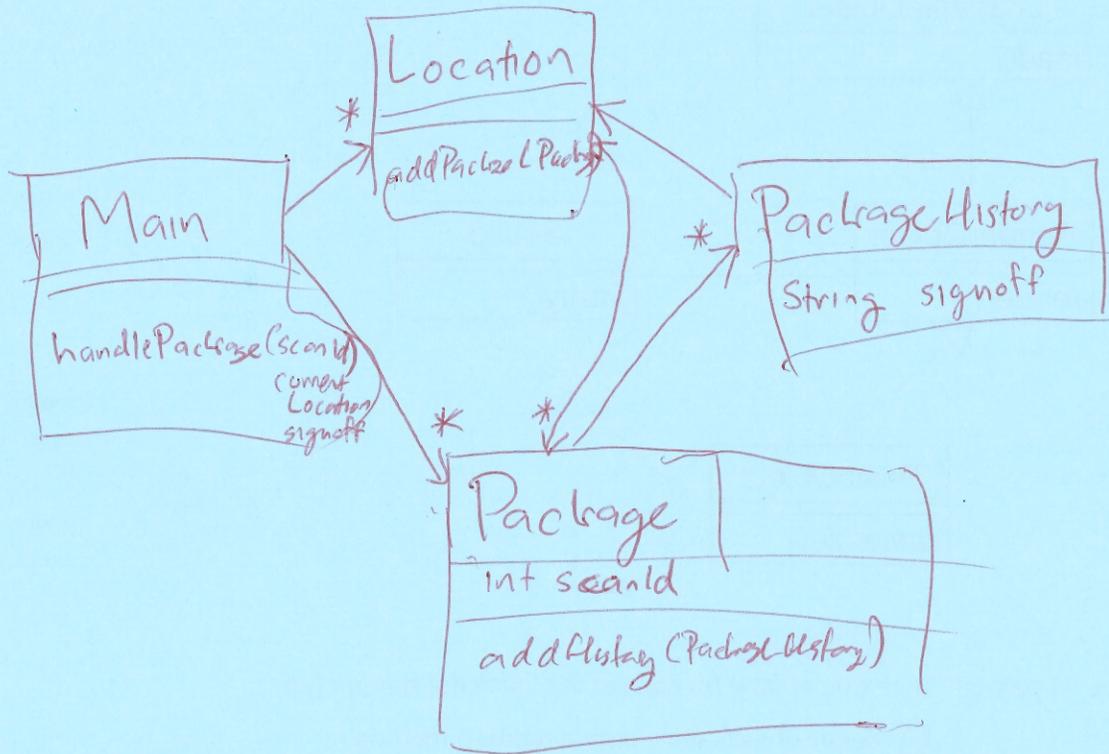
2. (10 points) This problem is a design exercise. First, read the problem description below. Then answer the questions.

Imagine a shipping company that only ships packages to certain predefined locations (could be intermediate shipping centers, could be holding centers where customers stop by to get their packages). As packages move, they need to keep a running record of every location they've visited, the time they arrived there, and who "signed off" that the package arrived in good condition.

When a delivery arrives truck arrives at a location with packages, the unloaders scan each package. This does two things: (a) it updates the list of all packages at that location, and (b) it updates the package history of the package to reflect it's new position.

Don't worry about a GUI for this problem – assume the employees access the system using a very simple web-based system built entirely in the Main class.

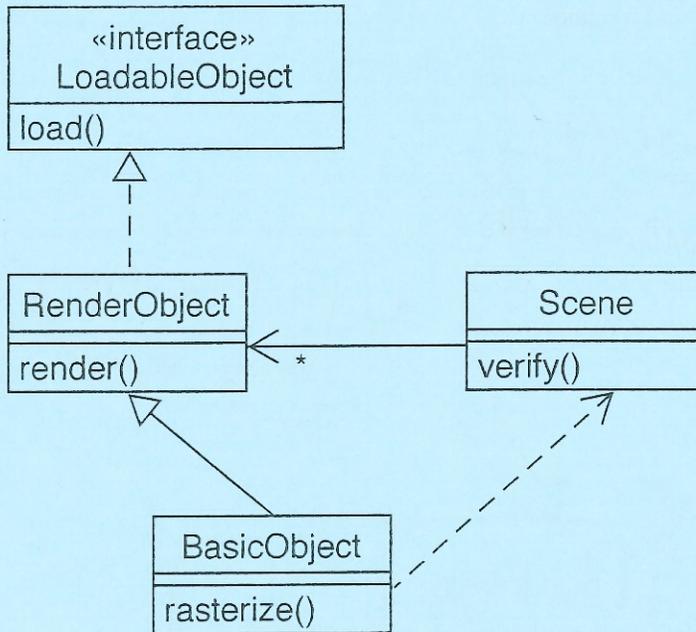
- a. (5 points) Draw a UML class diagram showing how you would design this system. You do not need to include every method or field - just the important ones. It should be clear from your diagram how all the data mentioned above is stored, and it should follow the OO principles we've discussed in class.



- b. (5 points) Describe in a few sentences how when an item is scanned on delivery to a location, the package's record is updated and the location's record is updated. Make it clear what methods call what other methods, and reference what you wrote in your UML diagram.

- ① When the package is scanned it calls `handlePackage` in main, passing both the package scan id and the location object for the current location
- ② The corresponding package is looked up by id
- ③ A new package history object is created with location and signoff, and this is added to the package
- ④ Then the `addPackage` is called ^{on location} with the package which updates the location's package list.

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 Scene objects must have a `render()` method
- F RenderObject objects must have a `load()` method
- T F RenderObject objects must have a `rasterize()` method
- T F BasicObject objects must have a field of type Scene
- F Scene objects inherit from the built-in java class Object

b. (2 points) Which of these best explains the "*" on the line between RenderObject and Scene?

- (a) Scene has a field of type `ArrayList<RenderObject>`
- (b) Scene has a field of type `RenderObject`, but it could be null
- (c) RenderObject has a field of type `ArrayList<Scene>`
- (d) Each `RenderObject` has many corresponding `Scene` objects, though `RenderObjects` do not have a reference to `Scene` objects
- (e) None of the above correctly explains the *

c. (2 points) Imagine that someone modifies `RenderObject` to be abstract, but does not add any abstract methods. At minimum, what other classes must change? (If no other classes need to change, write "none")

None

4. (4 points) Consider this code.

```
public class Reader {
    public ArrayList<String> readNumLines(Scanner s, int numLines)
        throws EOFException {
        ArrayList<String> toReturn = new ArrayList<String>();
        for (int i=0;i<numLines;i++) {
            if (!s.hasNextLine()) {
                throw new EOFException();
            }
            toReturn.add(s.nextLine());
        }
        return toReturn;
    }
}
```

Which of these statements are true?

T F This code will not compile without a try/catch that surrounds the "throw new EOFException".

T F This code will throw an exception if someone tries to read more lines than the Scanner source has remaining.

T F If an exception is thrown by this method, the method will return an empty ArrayList<String>.

T F This code does not need to include the "throws EOFException" because it is included inside the if statement.

5. (9 points) Imagine you came across this code in a functioning program:

```
ClassOne var1 = new Foo();  
ClassTwo var2 = var1;  
var2.magic();  
Mystery m = (ClassOne) var2;
```

Which of the following statements are true, given that the above code compiles?

- T F ClassOne might be a superclass of ClassTwo
- T F This line is definitely legal: var1.magic();
- T F Foo might be an interface
- T F The method magic might be abstract in ClassTwo
- T F The method magic might be abstract in Foo
- T F Mystery might be an interface
- T F Mystery might be a class
- T F This line is definitely legal: Mystery m2 = new Foo();
- T F This line is definitely legal: m.magic();

This page is intentionally left blank.

6. (16 points) Consider the following related declarations:

```
public interface I {
    void printHere();
}

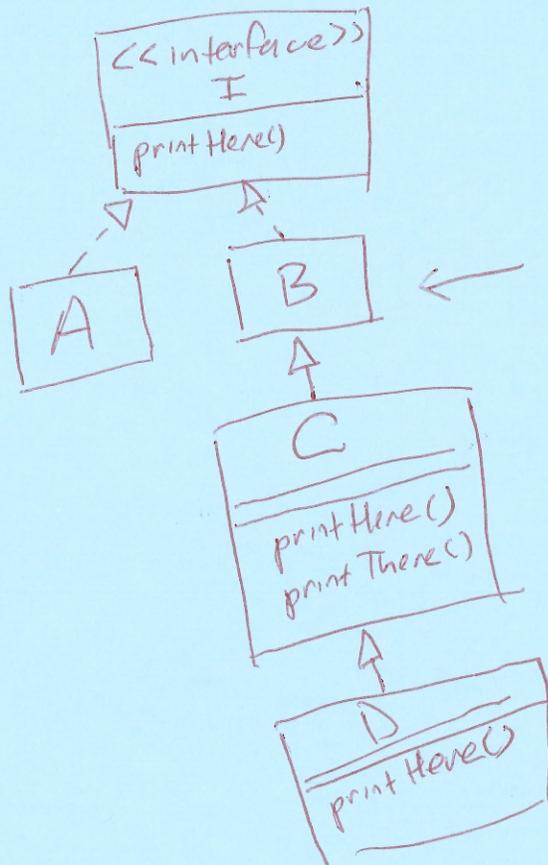
public class A implements I {
    public void printHere() {
        System.out.println("A");
    }
}

public class B implements I {
    public void printHere() {
        System.out.println("B");
    }
}
```

```
public class C extends B {
    public void printHere() {
        super.printHere();
        System.out.println("C");
    }
    public void printThere() {
        System.out.println("C there");
    }
}

public class D extends C {
    public void printHere() {
        System.out.println("D");
    }
}
```

a. (4 points) Draw a UML diagram to represent the given interface and classes (include all methods):



can have methods (but not required)

1 wrong ~~arrow~~ arrow
-1
2+ wrong arrows
-2

Wrong inheritance relationship OR Backwards arrows
-2

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

```

I i1 = new A();
I i2 = new B();
I i3 = new C();
I i4 = new D();
B b1 = new D();
C c1 = new C();

```

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	
i1.printHere();	runtime error	compile error <u>A</u>
A three = new B();	runtime error	<u>compile error</u>
c1.printHere();	runtime error	compile error <u>BC</u>
((D)b1).printHere();	runtime error	compile error <u>D</u>
i2.printHere();	runtime error	compile error <u>B</u>
c1.printThere();	runtime error	compile error <u>C there</u>
((A)i2).printHere();	<u>runtime error</u>	compile error <u>B</u>
i4.printHere();	runtime error	compile error <u>D</u>
i3.printHere();	runtime error	compile error <u>BC</u>
i3.printThere();	runtime error	<u>compile error</u>
C c2 = new D();	runtime error	compile error <u>nothing</u>
((B)i4).printHere();	runtime error	compile error <u>D</u>

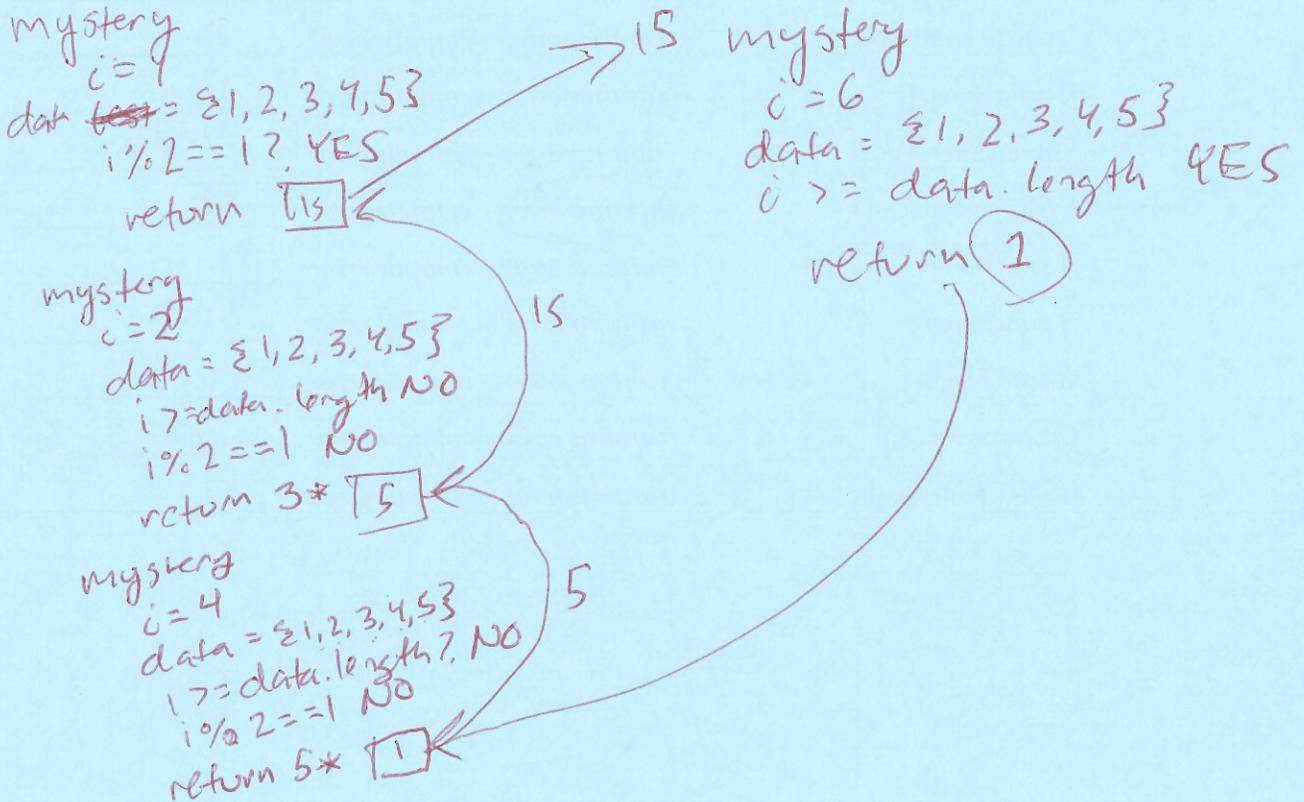
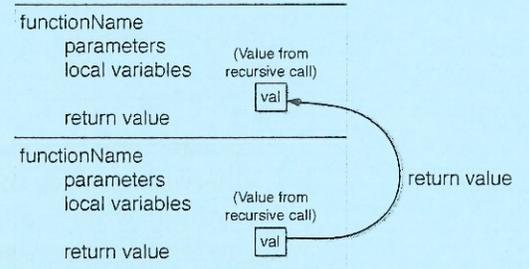
7. (12 points) For this problem, use the frame technique we practiced in the course to trace the execution of the recursive function call. Start your trace with the first call to mystery on line 14. A frame template is provided for your reference.

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

```

1 public class Foo {
2     public static int mystery(int i,int[] data) {
3         if (i>=data.length) {
4             return 1;
5         }
6         if (i%2==1) {
7             return mystery(i+1,data);
8         }
9         return data[i]*mystery(i+2, data);
10    }
11
12    public static void main(String[] args) {
13        int[] test = new int[] {1,2,3,4,5};
14        System.out.println(mystery(1, test));
15    }
16 }

```



For the code above, what would the final output be? 15

Use this page for additional workspace if you need it.

