

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

KEY

# CSSE 220—Object-Oriented Software Development

## Exam 2, February 2, 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	8	_____
6	17	_____
7	12	_____
<b>Paper Part Subtotal</b>	<b>65</b>	_____
C1. Recursion problems	15	_____
C2. GUI problem	20	_____
<b>Computer Part Subtotal</b>	<b>35</b>	_____
<b>Total</b>	<b>100</b>	_____

## 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).

HIGH COUPLING The GUI class has a reference to every object in the system, and most of them have a reference back to the GUI class.

HIGH COHESION A class that has many methods and fields, but all of them are about a single unifying idea.

NONE Point2D actually has two inner classes within it - Point2D.Double and Point2D.Float, which you use depending on how you want to precise you want your point data to be.

LOW COHESION Because your mail system supports different storage formats, the Mailbox class actually has 6 methods for saving the mail as different kinds of files, plus a large amount of network authentication code for communicating with cloud storage providers.

LOW COUPLING Using an interface like ActionListener, a JButton can call your code but it doesn't actually have a direct dependency on your code.

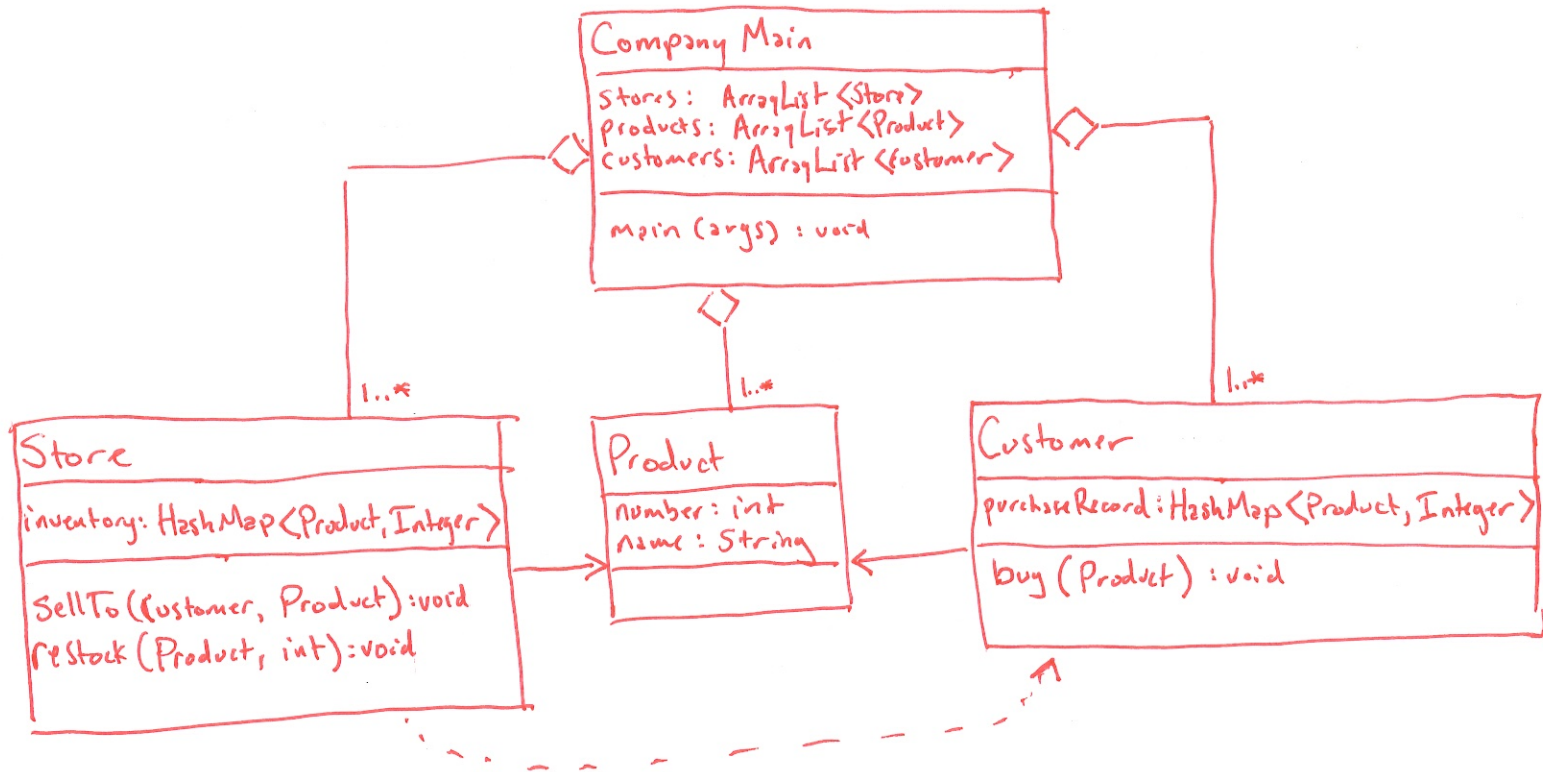
2. (10 points) This problem is a design exercise. First, read the problem description below. Then answer the questions.

A company has several stores that sell a variety of different business supplies like copiers, printers, and paperclips, etc. Each store has the same kinds of products and each product has a unique product number. Each store has different numbers of each product in stock, and each store needs to track how many items are in its inventory so it knows when to reorder.

When a customer buys something, a record must be kept of who bought what and the store's inventory needs to be modified to reflect whatever has been removed.

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.

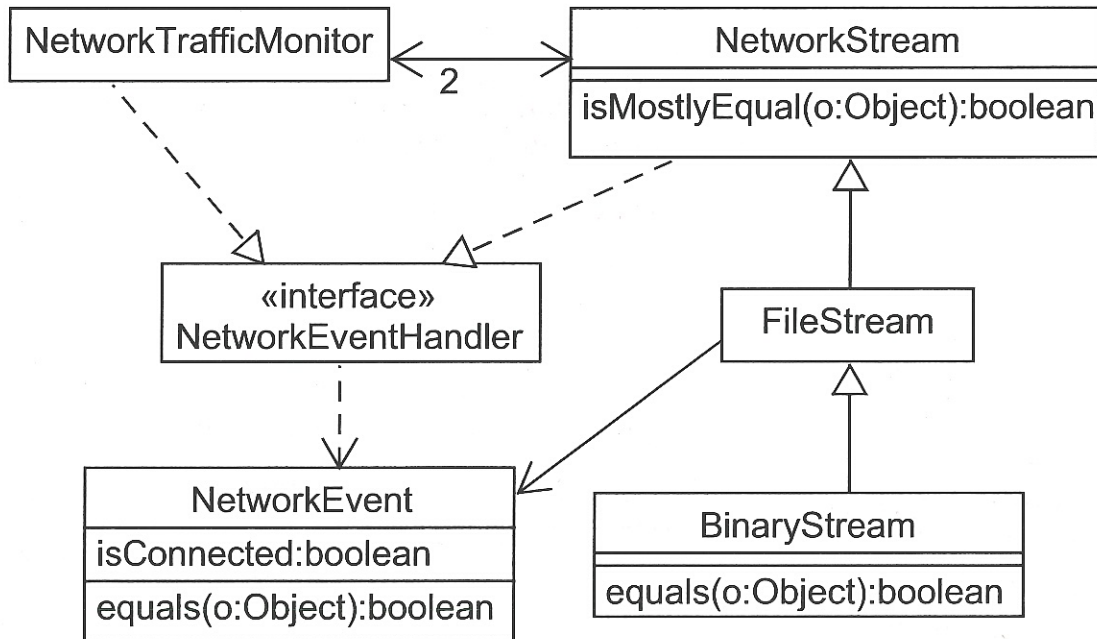
- a. (5 points) Draw a UML class diagram showing how you would design this system. You do not need to include every method or field - just the important ones. Be sure that your design easily and efficiently accommodates all features.



- b. (5 points) Describe in a few sentences how when an item is purchased the count of the item are modified in the store's inventory.

Employee triggers sale in `main()`, which calls `Store::sellTo()` with `z` product and customer. `sellTo()` reduces inventory for the product by 1, then calls `Customer.buy()` with the Product. `Customer.buy()` adds the product to `purchaseRecord` (or increments the count in that `HashMap`).

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



a. (3 points) Circle true or false for each of the following statements.

- T ☒ F Code in `NetworkEventHandler` creates `NetworkEvent` objects
- ☒ T F `FileStream` is a specific kind of `NetworkStream`
- T ☒ F `NetworkEvent` has a method `isConnected()` that returns a boolean

b. (3 points) A programmer attempting to make modifications notices that `FileStream` objects have a `equals` method, even though `FileStream.java` does not have the code for it. Which of these is the correct explanation for where the code for the `equals` method is located:

- (a) `FileStream` uses the code of `isMostlyEqual` in `NetworkStream`, but it is automatically renamed to `equals` in `FileStream`
- (b) `FileStream` uses the version implemented in `BinaryStream`
- (c) `FileStream` uses the version implemented in `NetworkEvent`
- ☒ (d) `FileStream` uses the version implemented in Java's `Object` class
- (e) `equals` is from implementing the `NetworkEventHandler` interface



- c. (3 points) Imagine that someone modifies NetworkEventHanlder to add and additional method. At minimum, what other classes must change? (If no other classes need to change, write "none")

NetworkStream and NetworkTrafficMonitor must change.

4. (4 points) Consider this code.

```
class Foo {  
  
    public int method() {  
        try {  
            Scanner s = new Scanner(new File("foo.txt"));  
            System.out.println("hello");  
        } catch (FileNotFoundException e) {  
            // code to handle the exception  
        }  
        System.out.println("goodbye");  
        return 99;  
    }  
}
```

Which of these statements are true?

T ☒ F This code will not compile unless someone adds a "throws FileNotFoundException" to the method header.

T ☒ F If a FileNotFoundException occurs and is handled successfully by the catch block, hello will be printed.

☒ T F If a FileNotFoundException occurs and is handled successfully by the catch block, goodbye will be printed.

T ☒ F If an exception was thrown in this method and was NOT caught, the method would return 0.

5. (8 points) Imagine you came across this code in a functioning program:

```
Mystery var1 = new Foo();  
var1 = new Bar();  
var1.magic();  
Foo var3 = new Foo();
```

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

- ☒ T ☐ F Mystery might be an interface
- ☒ T ☐ F Mystery might be a non-abstract class
- T ☒ F Foo might be an interface
- T ☒ F Foo might be an abstract class
- ☒ T ☐ F Foo and Bar could have the same superclass
- ☒ T ☐ F This line is definitely legal: var3.magic();
- T ☒ F This line is definitely legal: var3 = new Bar();
- ☒ T ☐ F If Foo is the superclass of Bar, the method called on the line "var1.magic()" could be implemented in ANY of: Mystery.java, Foo.java, or Bar.java.

6. Consider the following related declarations:

```
class W {
    public void doThis() {
        System.out.print("W");
    }
}

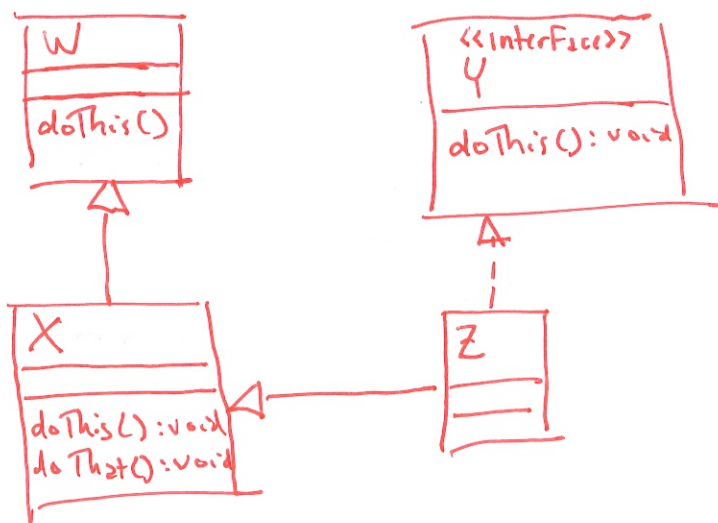
class X extends W {
    public void doThis() {
        super.doThis();
        System.out.print("X");
    }

    public void doThat() {
        System.out.print("That X");
    }
}
```

```
interface Y {
    public void doThis();
}

class Z extends X implements Y {
}
```

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



- b. (13 points) Continuing the same problem, suppose we declare and initialize these variables:

```
W w1 = new W();
W w2 = new X();
W w3 = new Z();
Y y1 = new Z();
Z z1 = new Z();
```

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.

Code	Either circle the error or provide the output	
w2.doThis();	runtime error	compile error <u>WX</u>
w2.doThat();	runtime error	compile error <u></u>
y1.doThis();	runtime error	compile error <u>WX</u>
y1.doThat();	runtime error	compile error <u></u>
((X) w1).doThis();	runtime error	compile error <u></u>
((X) w2).doThis();	runtime error	compile error <u>WX</u>
((X) w2).doThat();	runtime error	compile error <u>That X</u>
((X) w3).doThat();	runtime error	compile error <u>That X</u>
Y y3 = new X();	runtime error	compile error <u></u>
Y y4 = w3;	runtime error	compile error <u></u>
Y y5 = (Z) w3;	runtime error	compile error <u>"nothing"</u>
z1.doThis();	runtime error	compile error <u>WX</u>
z1.doThat();	runtime error	compile error <u>That X</u>



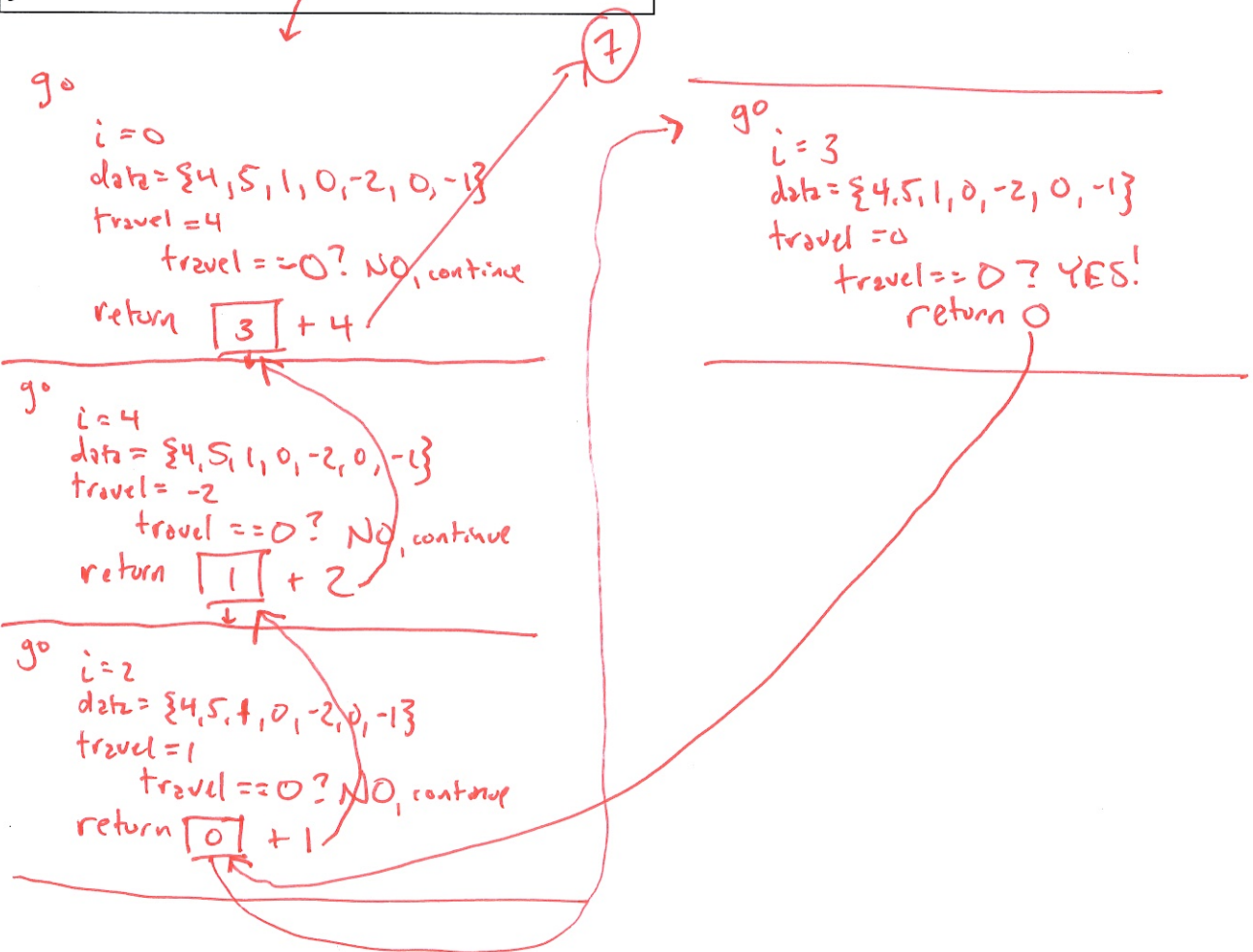
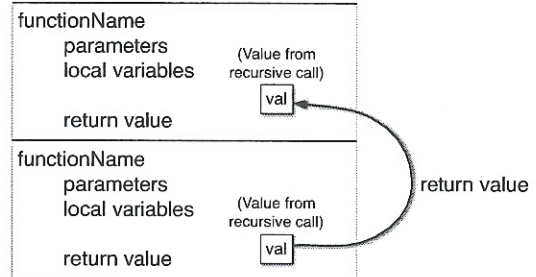
7. (12 points) For this problem, use the frame technique we practiced in the course to trace the execution of the recursive function call. Start your trace with the first call to go on line 10. A frame template is provided for your reference.

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

```

1 public static int go(int i, int[] data) {
2     int travel = data[i];
3     if(travel == 0) return 0;
4     return go(i + travel, data) + Math.abs(travel);
5 }
6
7 public static void main(String[] args) {
8     Exam2Fun obj = new Exam2Fun();
9     int[] input = new int[] {4,5,1,0,-2,0,-1};
10    System.out.println(go(0,input));
11 }

```



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

7

Use this page for additional workspace if you need it.