

Name: _____ Section: _____ CM: _____

CSSE 220—Object-Oriented Software Development

Exam 2, April 25th, 2018

This exam consists of two parts. Part 1 is to be solved on these pages. 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. You can also use your "UML Cheatsheet" and your "Design Principles" handouts if you brought them. Your computer *must be closed* the entire time you are completing Part 1.

Problem	Poss. Pts.	Earned
1	5	_____
2	8	_____
3	8	_____
4	4	_____
5	10	_____
6	15	_____
Paper Part Subtotal	50	_____
C1. Recursion problems	18	_____
C2. Polymorphism problem	12	_____
C3. GUI problem	20	_____
Computer Part Subtotal	50	_____
Total	100	_____

Part 1—Paper Part

1. (5 points total. 1 point each question.)

On this page there is a UML diagram and the code it represents. Please look at the code and diagram and then answer the questions below.

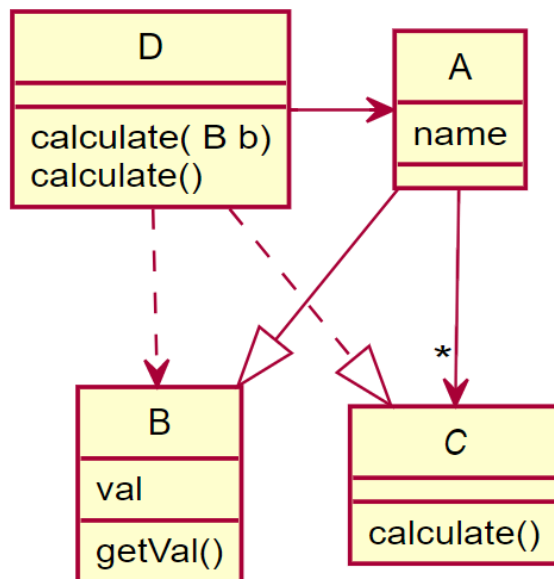
a. What line number(s) does the arrow starting at D and going to A correspond to?

b. What line number(s) does the arrow starting at D and going to B correspond to?

c. What line number(s) does the arrow starting at D and going to C correspond to?

d. What line number(s) does the arrow starting at A and going to B correspond to?

e. What line number(s) does the arrow starting at A and going to C correspond to?



```
class A extends B {
    private ArrayList<C> foo;
    private String name;
}

class B {
    private double val;

    public double getVal() {
        return this.val;
    }
}

interface C {
    public double calculate();
}

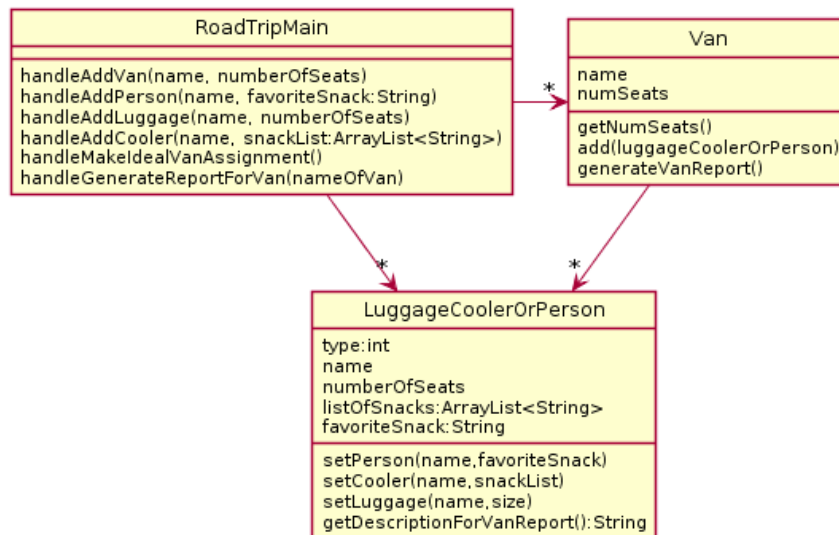
class D implements C{
    private A bar;
    public double calculate( B b){
        return b.getVal()*10;
    }

    public double calculate(){
        return 100;
    }
}
```

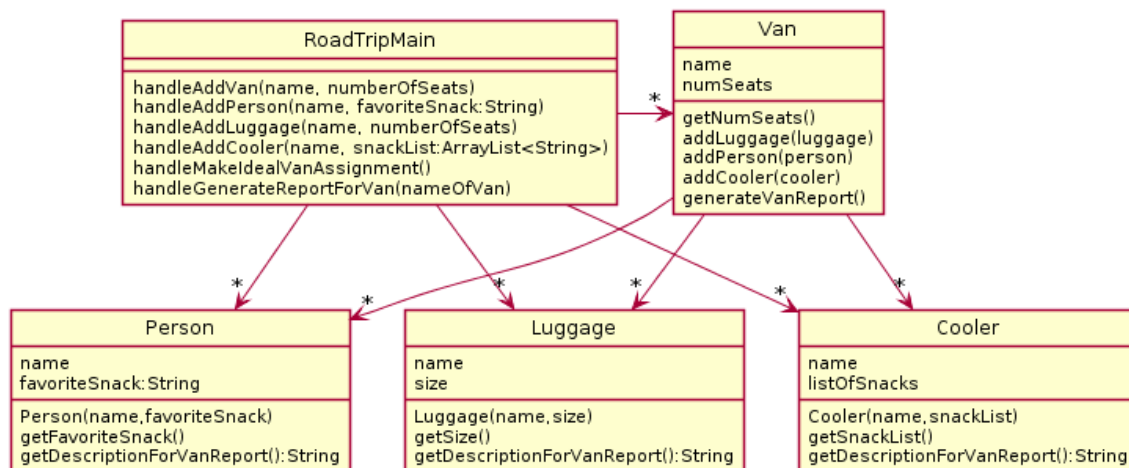

2. (8 points) A particular software system is designed to fill vans to go on a road trip. Three different kinds of things can be stored in a van - luggage, coolers, and people - each has different data that guides how they can be assigned. Luggage can take up 0-2 seats, depending on its size. Coolers take up 1 seat and have a list of snacks that the cooler contains. People take up 1 seat and have a favorite snack - they must be placed in a van that has a cooler with that snack. The code in `RoadTrip::handleMakeIdealVanAssignment()` allocates all luggage, coolers, and people to vans (prior to that, nothing is assigned to vans). Once everything has been allocated, no new things can be added but a report can be printed for each van which displays a description of each of their contents.

Here are 2 possible solutions:

Solution A



Solution B



- a. (2 points) Which of these two designs has a problem with cohesion? Explain both whether the issue is high or low cohesion and how you can tell from the diagram that the problem exists (One or two sentences is all that's needed).
- b. (2 points) Which of these two designs has a problem with coupling? Explain both whether the issue is high or low coupling and how you can tell from the diagram that the problem exists (One or two sentences is all that's needed).
- c. (4 points) For the design which has a problem with coupling, you can use an interface to reduce the coupling in the system. Make a UML diagram of how this might be done. For convenience, you can omit all fields and methods from the classes in your diagram EXCEPT the ones in the interface you add. Do make sure to draw all lines/arrows in your diagram.

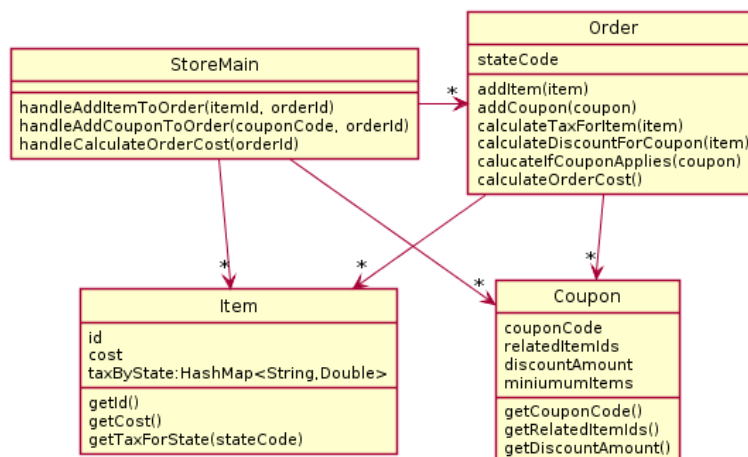
3. (8 points) In a particular online store, customers can add both items and coupons to their order. Items are added by selecting a particular item ID. In this store, items are unique (i.e. an item cannot be added to more than one order, and each itemID is unique).

Coupons are added by entering a coupon code. A coupon looks like this “COUPON code: foobar Entitles you to \$1 off PER ITEM of item75, item76, item77 if you order AT LEAST 2”. The same coupon can be applied to multiple orders.

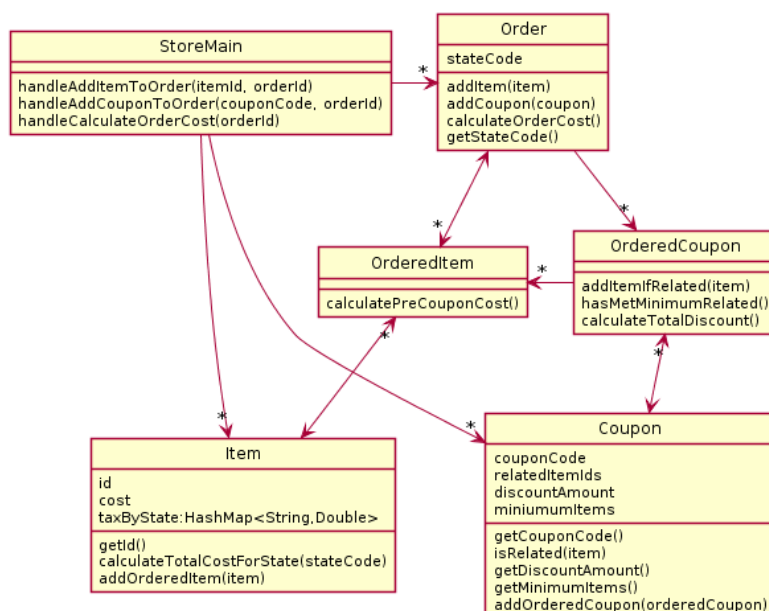
To calculate the total cost, the cost of each item must be added plus the tax (which can vary both by item and by state). Coupons must also be applied. Each coupon only applies to certain related items. Coupons require that a minimum number of related items be in the order before the coupon provides a discount. When a coupon does apply, it always provides a fixed amount off each related item and it does not affect tax.

Here are 2 possible solutions. **You can assume both of these designs function correctly (that is, exclude principle 1 from your consideration).**

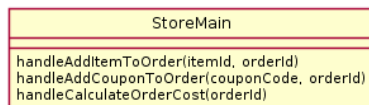
Solution A



Solution B



- a. (2 points) Explain the problems with Solution A using your design principles.
- b. (2 points) explain the problems with Solution B using your design principles.
- c. (4 points) Make a UML diagram of your proposed solution to the problem. For your solution, we have provided a StoreMain to get you started. Feel free to omit any regular getter methods in your solution diagram as well.



4. (4 points) Consider the following code.

```
public static void process(String word) throws IllegalArgumentException{
    if (word.length() == 0) {
        throw new IllegalArgumentException();
    }
    try {
        System.out.println( firstHalf(word) );
        System.out.println( middleChar(word) );
    } catch (IllegalArgumentException e) {
        System.out.println( "bad arg" );
    } catch (ArithmeticException e) {
        System.out.println( "bad math" );
    }
}

public static String firstHalf(String x) throws ArithmeticException {
    if (x.length() % 2 != 0) {
        throw new ArithmeticException();
    }
    return x.substring(0, x.length()/2);
}

public static char middleChar(String y) throws ArithmeticException {
    if (y.length() % 2 != 1) {
        throw new ArithmeticException();
    }
    return y.charAt( y.length()/2 );
}

public static void main(String[] args) {
    try {
        System.out.println( middleChar( "abc" ) );
        process( "abc" );
        process( "abcd" );
        process( "" );
        System.out.println( "Finished!" );
    } catch (IllegalArgumentException e) {
        System.out.println( "Invalid word" );
    }
}
```

What would this code print? List the output below. (You may add additional lines if needed)

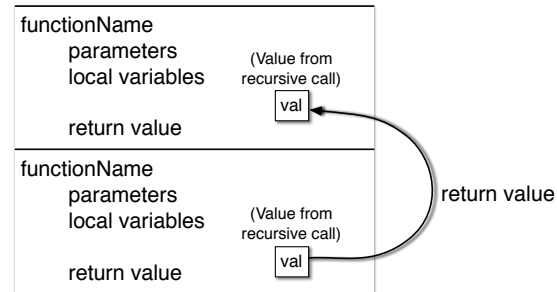
5. (10 points) For this problem, use the frame technique we practiced in the course to trace the execution of the recursive function call. Start your trace with the first call to `mystery` on line 17. A frame template is provided for your reference.

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

```

1 public static String mystery(String a, String b) {
2     if (a.length() == 0) {
3         return b;
4     }
5     char fA = a.charAt(0);
6     char fB = b.charAt(0);
7
8     if ( fA == fB) {
9         return fA + mystery( a.substring(1), b.substring(1) );
10    } else {
11        return mystery( a.substring(1), b.substring(1) );
12    }
13 }
14
15 public static void main(String[] args) {
16     System.out.println( mystery( "abc", "cbaz" ) );
17 }

```



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

6. (15 points) Consider the following related declarations. @Override annotations are omitted to save space:

```
interface Wave {  
    public void process();  
}  
class Alpha implements Wave {  
    public void process() {  
        System.out.print( "A" );  
    }  
    public void dualProcess(Wave w) {  
        w.process();  
        w.process();  
    }  
}  
class Beta extends Alpha {  
    public void process() {  
        System.out.print( "B" );  
    }  
}
```

```
class Gamma extends Beta {  
    private String woo;  
  
    public Gamma(String woo) {  
        this.woo =woo;  
    }  
  
    public void dualProcess(Wave w) {  
        System.out.print(woo);  
        super.dualProcess( w );  
        System.out.print(woo);  
    }  
}
```

- 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. In places where lines representing fields are appropriate, use lines and do NOT re-list the same field in the field list.

- b. (11 points) Continuing the same problem, suppose we declare and initialize these variables:

```
Gamma gw = new Gamma("W");
Beta bx = new Gamma("X");
Alpha ay = new Gamma("Y");
Wave wz = new Gamma("Z");
Wave wa = new Alpha();
Beta bb = new Beta();
```

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		
wa.process();	<i>runtime error</i>	<i>compile error</i>	_____
Beta b = new Alpha();	<i>runtime error</i>	<i>compile error</i>	_____
gw.process();	<i>runtime error</i>	<i>compile error</i>	_____
wa.dualProcess(new Alpha ());	<i>runtime error</i>	<i>compile error</i>	_____
((Alpha) bx).process();	<i>runtime error</i>	<i>compile error</i>	_____
((Gamma)bb).process();	<i>runtime error</i>	<i>compile error</i>	_____
gw.dualProcess(new Wave());	<i>runtime error</i>	<i>compile error</i>	_____
gw.dualProcess(new Alpha());	<i>runtime error</i>	<i>compile error</i>	_____
((Beta)wa).dualProcess(new Alpha());	<i>runtime error</i>	<i>compile error</i>	_____
bb.dualProcess(bb);	<i>runtime error</i>	<i>compile error</i>	_____
ay.dualProcess(ay);	<i>runtime error</i>	<i>compile error</i>	_____
