Name: KEY 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.

KEY

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	______

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).

Solution A has low cohesion because LuggageCoolerOrPerson is doing more than one responsibility. "Doing too much"

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).

Solution B has high coupling because there are many dependencnies.

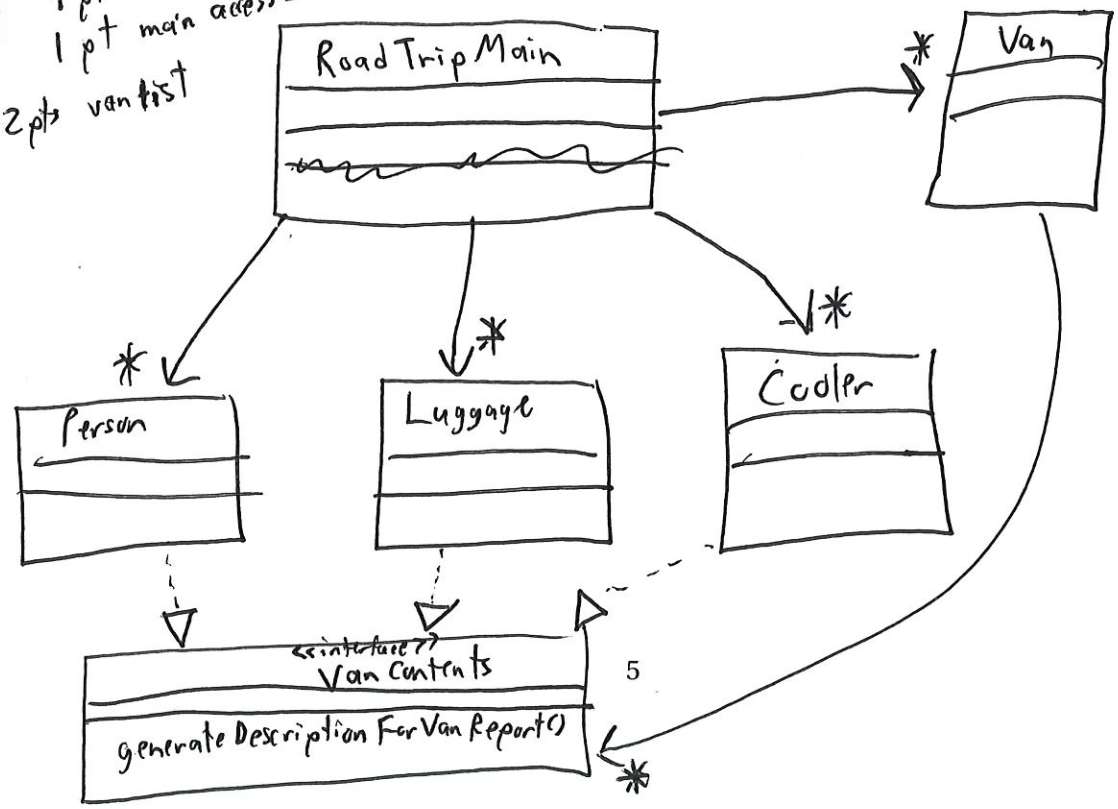
c. (4 points) For the design which has a problem with coupling, you can use an interface to reduce the coupling in the system. Make a UML diagram of how this might be done. For convenience, you can omit all fields and methods from the classes in your diagram EXCEPT the ones in the interface you add. Do make sure to draw all lines/arrows in your diagram.

4

1 pt implements

1 pt main accesses concrete

2 pts van list



a. (2 points) Explain the problems with Solution A using your design principles.

Order doing too much

There are ask methods in Item, Coupon instead of "tell"

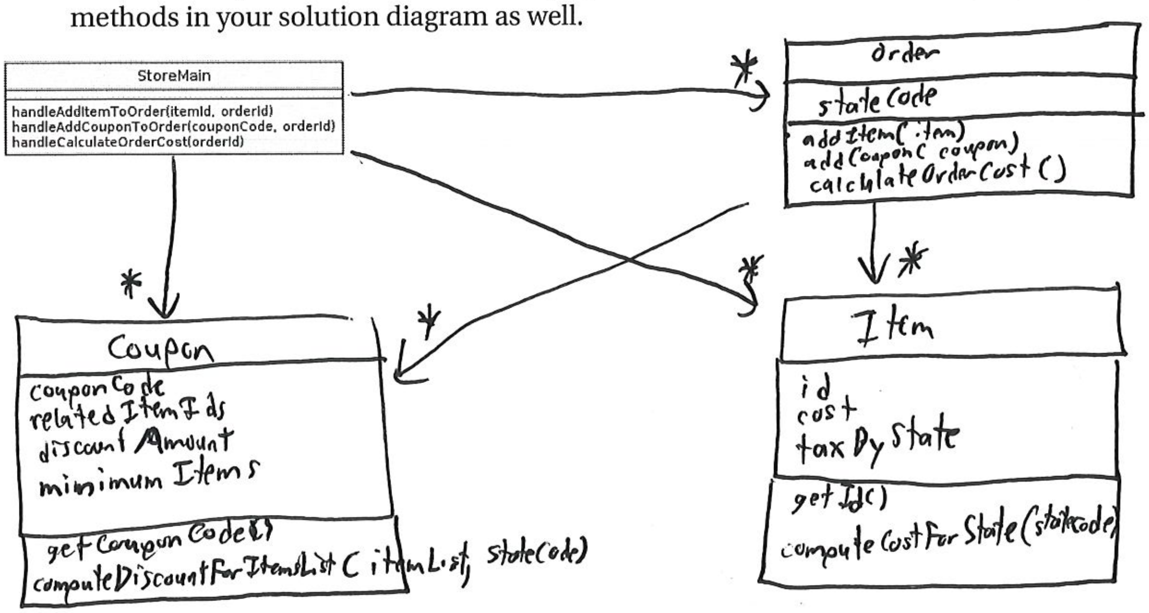
Item, Coupon are data holder classes

b. (2 points) explain the problems with Solution B using your design principles.

Too much interdependency

OrderedItem, orderedCoupon are not cohesive

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.


StoreMain
handleAddItemToOrder(itemId, orderId)
handleAddCouponToOrder(couponCode, orderId)
handleCalculateOrderCost(orderId)
*
Order
stateCode
addItem(item)
addCoupon(coupon)
calculateOrderCost()
*
*
Coupon
couponCode
relatedItemIds
discountAmount
minimumItems
getCouponCode()
computeDiscountForItemList(itemList, stateCode)
*
*
Item
id
cost
taxByState
getId()
computeCostForState(stateCode)

1/4 runs

2/4 coupon calcs

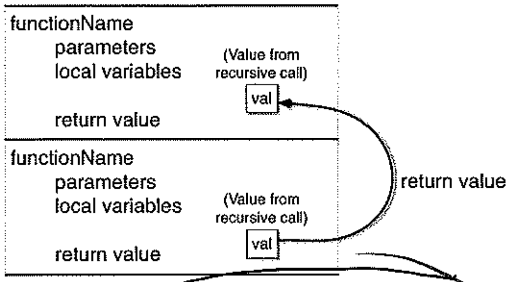
order calculate

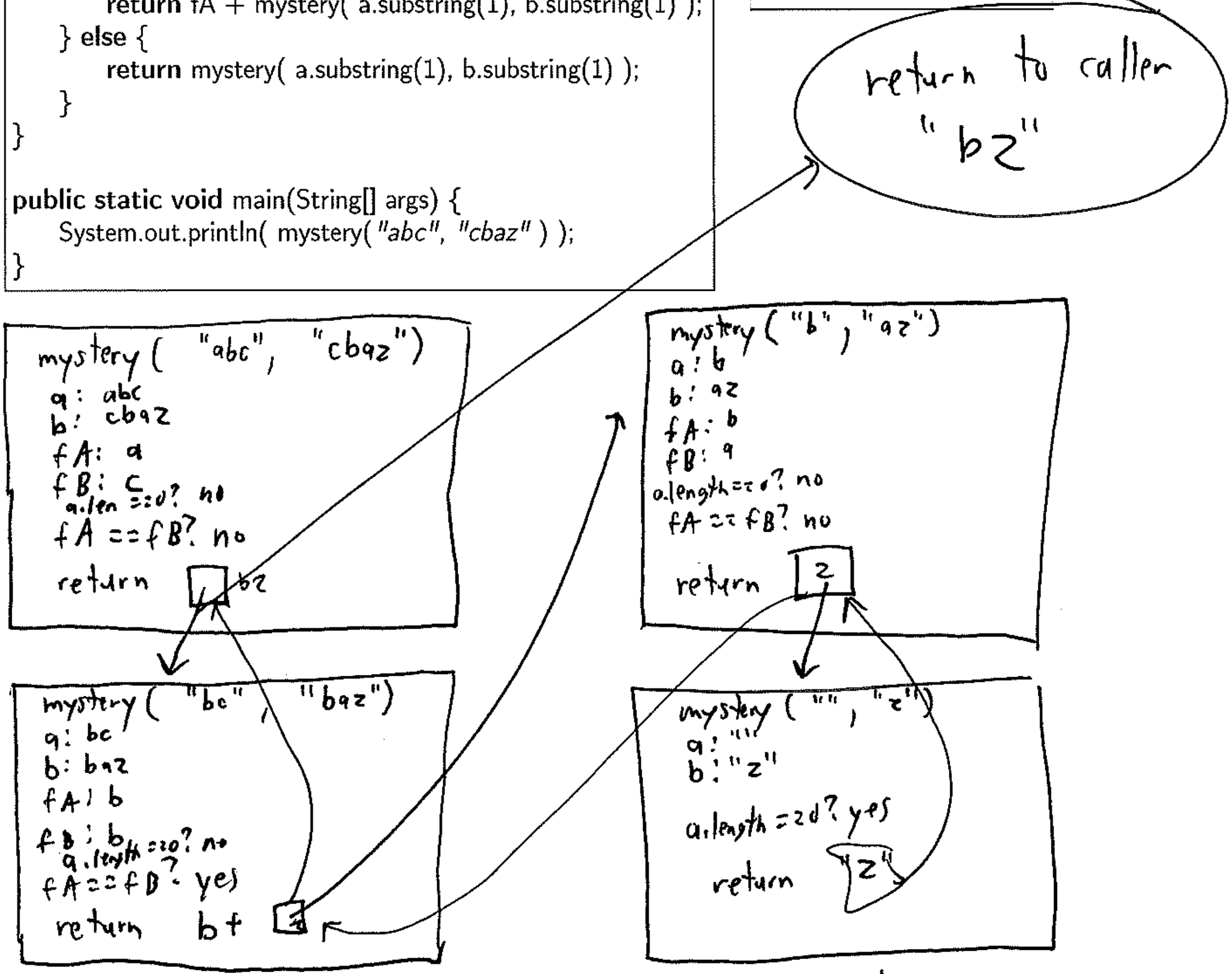
3/4

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 }
```





return to caller
"bz"

mystery ("abc", "cbaz")
a : abc
b: cbaz
fA: a
fB: c
a.len ==0? no
fA ==fB? no
return bz

mystery ("bc", "baz")
a: bc
b: baz
fA: b
fB: b
a.length ==0? no
fA==fB? yes
return b+ z

mystery ("b", "az")
a: b
b: az
fA: b
fB: a
a.length==0? no
fA == fB? no
return z

mystery ("", "z")
a: ""
b: "z"
a.length ==0? yes
return "z"

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

rubric
2 points arrows ↓↑
2 points correct output
6 points for correct trace with 4 frames

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();`	*runtime error*	*compile error*	A
`Beta b = new Alpha();`	*runtime error*	[✓ *compile error*]	
`gw.process();`	*runtime error*	*compile error*	B
`wa.dualProcess(new Alpha ());`	*runtime error*	[✓ *compile error*]	
`((Alpha) bx).process();`	*runtime error*	*compile error*	B
`((Gamma)bb).process();`	[✓ *runtime error*]	*compile error*	
`gw.dualProcess( new Wave() );`	*runtime error*	[✓ *compile error*]	
`gw.dualProcess( new Alpha() );`	*runtime error*	*compile error*	WAAW
`( (Beta)wa ).dualProcess(new Alpha() );`	[✓ *runtime error*]	*compile error*	
`bb.dualProcess( bb );`	*runtime error*	*compile error*	BB
`ay.dualProcess( ay );`	*runtime error*	*compile error*	YBBY