Name: ______________________________

# CSSE 220—Object-Oriented Software Development

## Exam 2, Oct. 28, 2013

This exam consists of two parts. Part 1 is to be solved on these pages. Ask your instructor for scratch paper 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.

*Resources for Part 2*: Open book, notes, and computer. Limited network access. You may use the network only to access your own files, the course Moodle site and web pages, the textbook's site, Sun'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.

Parts 1 and 2 are included in this document. You should read over all of the questions before beginning work, but...

You must turn in part 1 before accessing the resources for part 2.

| Problem | Poss. Pts. | Earned |
|---:|:---:|:---:|
| 1 | 5 | ______ |
| 2 | 12 | ______ |
| 3 | 12 | ______ |
| 4 | 4 | ______ |
| 5 | 3 | ______ |
| 6 | 6 | ______ |
| 7 | 16 | ______ |
| 8 | 12 | ______ |
| **Paper Part Subtotal** | **70** | ______ |
| C1. CompassComponent | 15 | ______ |
| C2. Recursion problems | 15 | ______ |
| **Computer Part Subtotal** | **30** | ______ |
| Total | 100 | ______ |

## Part I—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: High Coupling, Low Coupling, High Cohesion, Low Cohesion, and None (for the description that isn't really describing coupling or cohesion). You'll use each phrase exactly once.

high coupling A program with 15 different very small classes, all of which have dependencies on each of the other 14

high cohesion A class called Date which has many methods all of which are involved with creating, comparing, and converting dates into various formats

low cohesion An entire video game implemented in a single class which different functions that have do with saving, drawing graphics, enemy AI, and many other different features

none A class Vechicle, which has subclasses Car and Bike, and Car has subclasses RacingCar and SchoolBus

low coupling A class AppointmentCalendar which uses a class Appointment but otherwise does not depend on any other class in the system

2. (12 points) This problem is a design exercise. First read the problem description below, then answer the questions.

*Design a program to track orders for cashiers in a pizza delivery restaurant. The system needs to track which pizzas each order contains, what toppings and crust–type the pizzas have, and print receipts that itemize the cost of each pizza and the cost of the overall order with taxes and delivery fees. The system must also track where each order ought to be delivered.*

a. List at least **five** *candidate classes* that you might use in implementing a solution to the problem:

Order Receipt
[✓ Pizza] Address
Topping Crust-Type

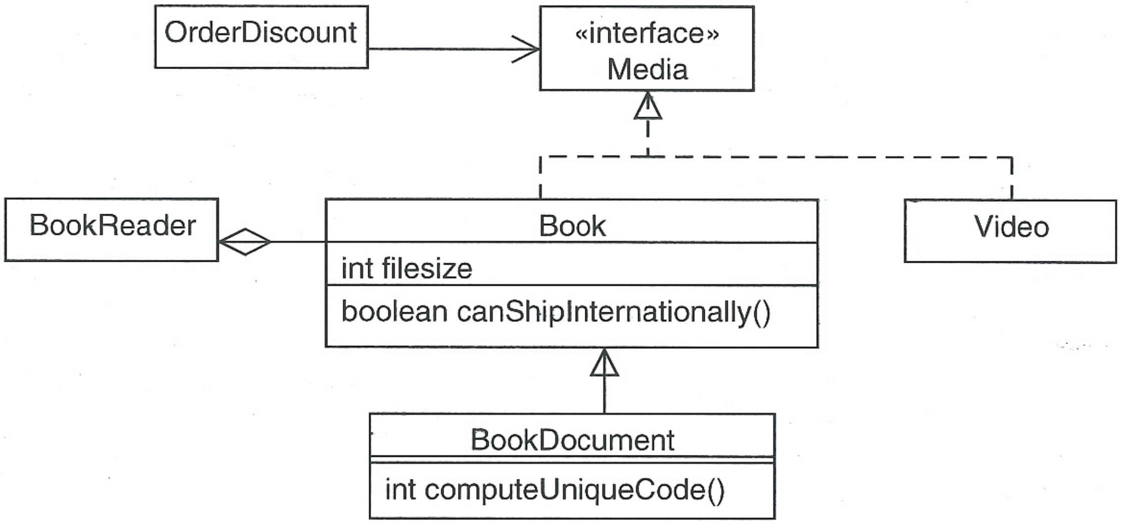
b. Pick one of your candidate classes, **circle it above** and briefly describe three responsibilities that it might have:

- add Topping
- getReceipt Listing
- get Cost()

c. Still considering your chosen class from part b, with what other classes might it need to collaborate?

Order
Topping
Receipt
Crust Type

3. (12 points) Use this UML class diagram to answer the subsequent questions.



a. Because of the line between BookReader and Book, if you looked at the code you would expect to see (circle your answer):

[✓ (a)] A field of type Book in the BookReader class

(b) A field of type BookReader in the Book class

(c) That the BookReader class inherits from Book

(d) That the Book class inherits from BookReader

(e) None of the above statements can be true, given this diagram

Look at Media, Book, Video, and OrderDiscount in the diagram. Given your the diagram and your knowledge of objects, which of these statements is most likely to be true (circle your answer):

b. [✓ (a)] OrderDiscount objects can apply to either Books or Video

(b) OrderDiscount objects can apply only to instances of the Book class

(c) OrderDiscount objects can only apply to instances of the Media class

(d) OrderDiscount objects can be used in place of Books or Videos

(e) OrderDiscount objects construct Book and Video classes

c. Circle true or false for each of the following statements.

[✓ T] F BookDocument inherits the field fileSize from Book

T [✓ F] Book inherits the method computeUniqueCode from BookDocument

[✓ T] F BookDocument inherits the method canShipInternationally from Book

d. If we added a method to the Media interface, *at a minimum*, what other classes or interfaces must be changed?

Book Video

4. (4 points) What is the difference between private and protected methods?

Private methods can only be used in the class they are declared.

Protected methods can be used in that and in sublasses.

5. (3 points) Say you were looking at some code and you saw this:

```
try {
    this.doCoolFunction();
} catch (BadLocationException e) {
    // a bunch of complicated
    // exception handling code
}
```

What would have to happen in doCoolFunction for that exception handling code to get invoked?

doCoolFunction would have to throw an exception of type BadLocationException (or a subclass)

6. (6 points) Circle true or false for each of the following statements.

[✓ T] F Classes can implement more than one interface

[✓ T] F Abstract classes cannot be instantiated

[✓ T] F Interfaces cannot be instantiated

[✓ T] F If you have an abstract class A that is a superclass of B, you can store an object of Type B in a variable of type A

[✓ T] F If you make a non-abstract subclass of an abstract class, you must implement every abstract method in the superclass

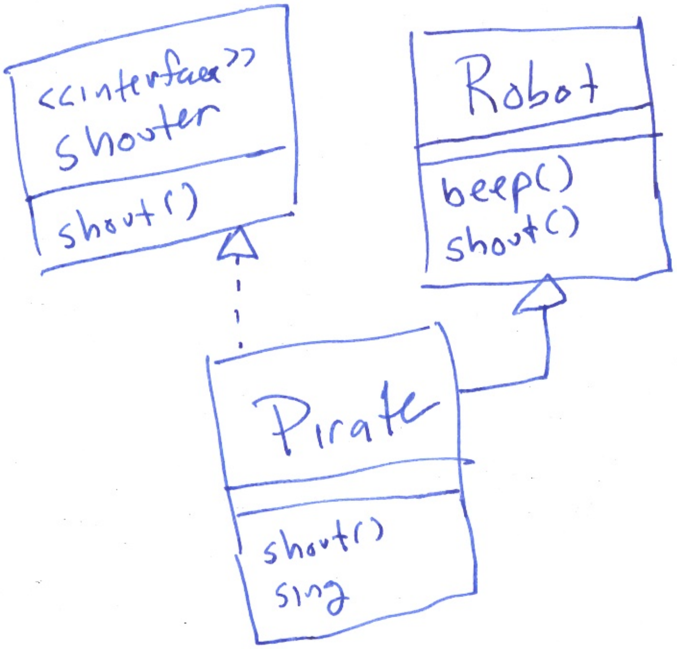
[✓ T] F Abstract methods can methods with code, interfaces cannot

7. Consider the following related declarations:

```
class Robot {
    public void beep() {
        System.out.print( "Beep");
    }
    public void shout() {
        System.out.print( "Zap");
    }
}

interface Shouter {
    void shout();
}
```

```
class Pirate extends Robot
    implements Shouter {

    public void shout() {
        super.shout();
        System.out.print( "Arr");
    }

    public void sing() {
        System.out.println( "Yo-ho");
    }
}
```

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



b. Continuing the same problem, suppose we declare and initialize these variables:

```
Robot r1 = new Robot();
Robot r2 = new Pirate();
Pirate p = new Pirate();
Shouter s;
```

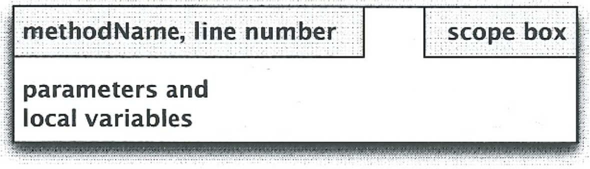
(12 points) For each line of code below, **circle** what would be output. If a line would give an error, then circle the type of error. Consider each line separately. That is, if a line would give an error, then assume that line doesn't affect any others.

| Code | Output Choices (circle one in each line) | | | | | | |
|---|---|---|---|---|---|---|---|
| r1.shout(); | Beep | [✓ Zap] | Arr | Yo–ho | ZapArr | *runtime error* | *compile error* |
| r2.shout(); | Beep | Zap | Arr | Yo–ho | [✓ ZapArr] | *runtime error* | *compile error* |
| p.shout(); | Beep | Zap | Arr | Yo–ho | [✓ ZapArr] | *runtime error* | *compile error* |
| r1.sing(); | Beep | Zap | Arr | Yo–ho | ZapArr | *runtime error* | [✓ *compile error*] |
| r2.sing(); | Beep | Zap | Arr | Yo–ho | ZapArr | *runtime error* | [✓ *compile error*] |
| r1.beep(); | [✓ Beep] | Zap | Arr | Yo–ho | ZapArr | *runtime error* | *compile error* |
| r2.beep(); | [✓ Beep] | Zap | Arr | Yo–ho | ZapArr | *runtime error* | *compile error* |
| p.beep(); | [✓ Beep] | Zap | Arr | Yo–ho | ZapArr | *runtime error* | *compile error* |
| ((Pirate) r2).sing(); | Beep | Zap | Arr | [✓ Yo–ho] | ZapArr | *runtime error* | *compile error* |
| s = r1; s.shout(); | Beep | Zap | Arr | Yo–ho | ZapArr | *runtime error* | [✓ *compile error*] |
| s = p; s.shout(); | Beep | Zap | Arr | Yo–ho | [✓ ZapArr] | *runtime error* | *compile error* |
| s = p; s.beep(); | Beep | Zap | Arr | Yo–ho | ZapArr | *runtime error* | [✓ *compile error*] |

8. (12 points) For this problem use the frame technique we practiced in class and the class declaration at left. Trace the execution of the call func("1a2") in main() and answer the question at the bottom of the page. A frame template is provided for your reference.

```
1  public class Exam2Fun {
2      public String func(String input) {
3          if(input.isEmpty()) return "";
4          char firstChar = input.charAt(0);
5          String remove1st = input.substring(1);
6          String recurse = this.func(remove1st);
7          if(Character.isDigit(firstChar)) {
8              return recurse + firstChar;
9          } else {
10             return firstChar + recurse;
11         }
12     }
13 }
```

```
public static void main(String[] args) {
    Exam2Fun obj = new Exam2Fun();
    System.out.println(obj.func( "1a2"));
}
```



"a21"

8 func
input = "1a2"
firstChar = '1'
remove 1st = "a2"
recurse = "a2"

"a2"

10 func
input = "a2"
first char = 'a'
remove 1st = "2"
recurse = "2"

"2"

8 func
input = "2"
firstchar = '2'
remove First = ""
recurse = ""

""

3 func
input = ""
first char =
remove First =
recurse =

For the code above, *what would the final output be*? a21