

Name: _____

KEY

CSSE 220—Object-Oriented Software Development

Exam 2, October 24, 2014 ~~2014~~ 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	12	_____
3	12	_____
4	4	_____
5	8	_____
6	17	_____
7	12	_____
Paper Part Subtotal	70	_____
C1. Recursion problems	15	_____
C2. MineSweeper GUI	15	_____
Computer Part Subtotal	30	_____
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).

high coupling In an object design with 30 classes, the class GameFactory depends on 25 of those classes.

low coupling When you need to add a new kind of product, you simply make a new class that implements the Product interface and almost no other classes need to be modified.

low cohesion Through the use of anonymous inner classes, it's possible to write an entire system in one single method which creates a GUI, creates data objects, handles events, and reads and writes files.

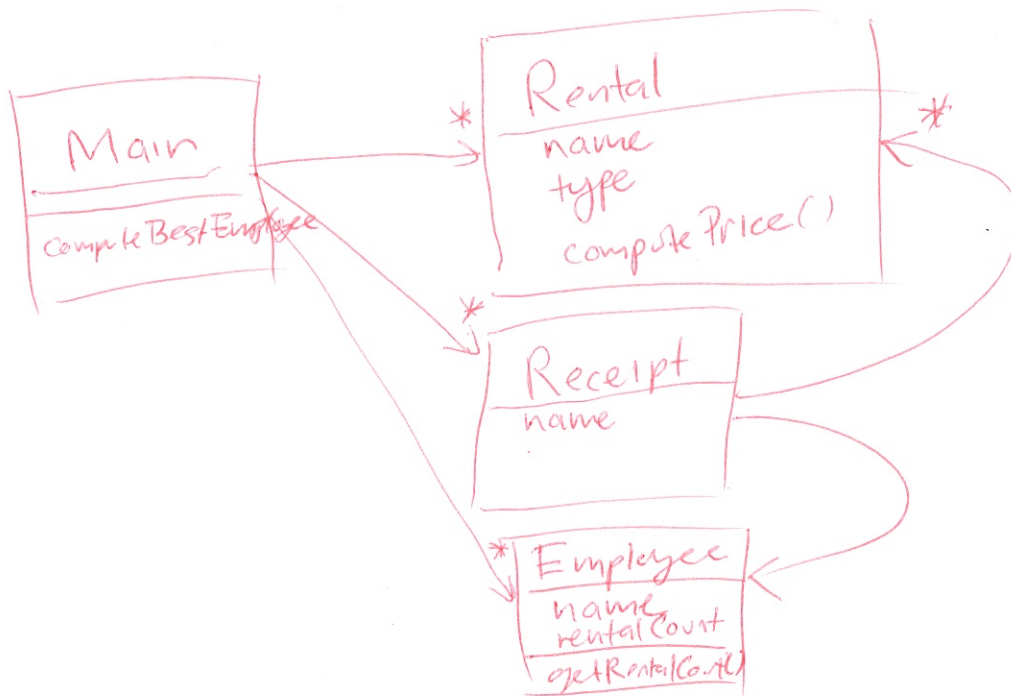
none An abstract class has two abstract methods and two non-abstract methods, but subclasses occasionally override the non-abstract methods as well.

high cohesion A small class called GPA stores a student's GPA and handles the complexity of computing the GPA for honors courses, which is a bit more complicated than you'd think.

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

Design a program to manage a media rental store. The store rents DVDs, Blu-Rays, and video games — each of them has a different price. There are two main features of the system - (1) printing out a receipt that shows how much the customer owes, and (2) determining which of the employees has rented out the most and gets "Employee of the Week." Don't worry about the GUI of this system - you can assume the employees interact using a text system entirely contained in the Main class.

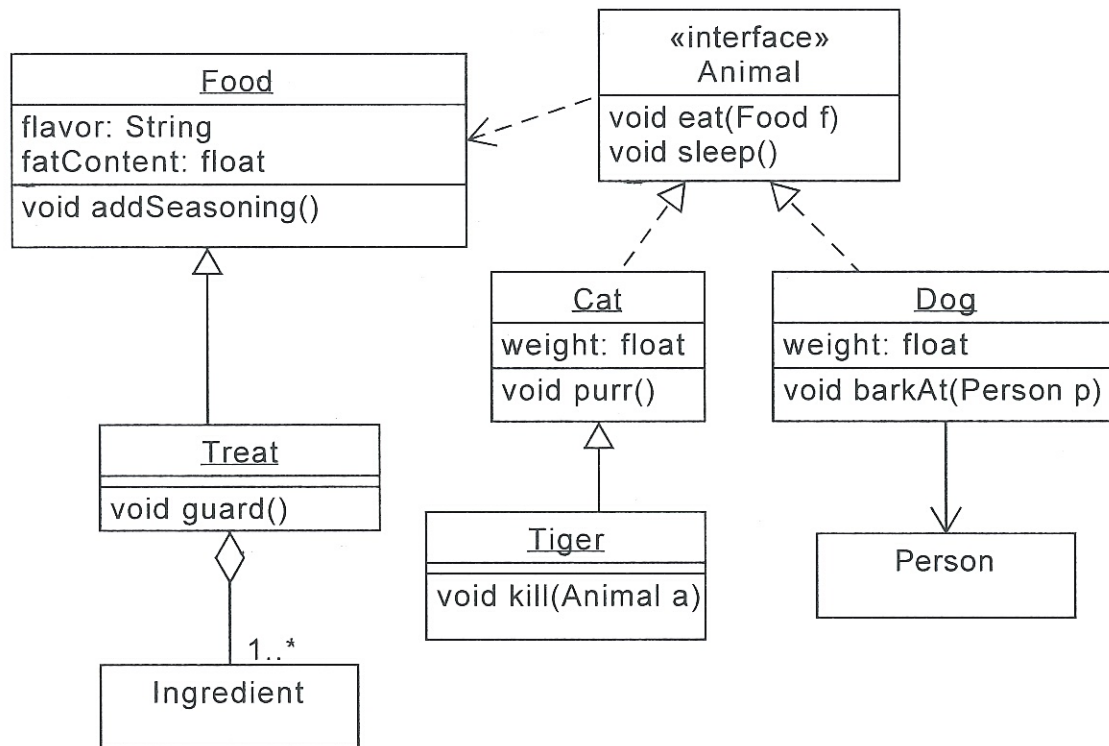
- a. (6 points) Draw a UML class diagram showing how you would design this system. You do not need to include every method or field - just the important ones. Be sure that your design easily and efficiently accommodates both features.



- b. (6 points) Describe in a few sentences how "Employee of the Week" is calculated, referencing your diagram above.

main calls computeBestEmployee
 computeBestEmployee goes through the employee list
 looking for the one with the highest
 rental count. (rental count is incremented
 each time an item is added to a
 receipt)
 computeBestEmployee returns the employee and it is
 displayed

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



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

- T ☒ F Ingredient inherits the field flavor from the Food class
- ☒ T F Treat inherits the field fatContent from the Food class
- ☒ T F If you had two Tiger objects, you could pass the first Tiger as a parameter to the second Tiger's kill method.
- T ☒ F By the diagram, within some Dog method, a Person object is constructed

b. Because of 1..* on the line between Treat and Ingredient, if you looked at the code you would most expect to see (circle your answer):

- (a) Ingredient objects have an ArrayList of Treat objects
- ☒ (b) Treat objects have an ArrayList of Ingredient objects
- (c) Treat objects have a single field of Ingredient type
- (d) Ingredient objects may inherit from Treat
- (e) None of the above statements can be true, given this diagram

- c. Given the diagram and your knowledge of objects, which of these classes *must* have the method `sleep()`?

4. (4 points) Consider this code.

```
class Foo {  
    private int a;  
    protected int b;  
  
    protected void setA(int aVal) {  
        this.a = aVal;  
    }  
  
    public void setB(int bVal) {  
        this.b = bVal;  
    }  
}  
  
class Bar extends Foo {  
  
    public void trickyMethod() {  
        // code goes here  
    }  
}
```

Which of these lines would be legal (i.e., would compile) if placed in `trickyMethod`? Select “T” if they would compile, “F” otherwise.

- T ☒ F `this.a = 7;`
- ☒ T F `this.b = 7;`
- ☒ T F `super.setA(7);`
- ☒ T F `this.setB(7);`

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

```
//Mover is an abstract class  
Mover dude = new Runner();  
dude.pass(otherObject);
```

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

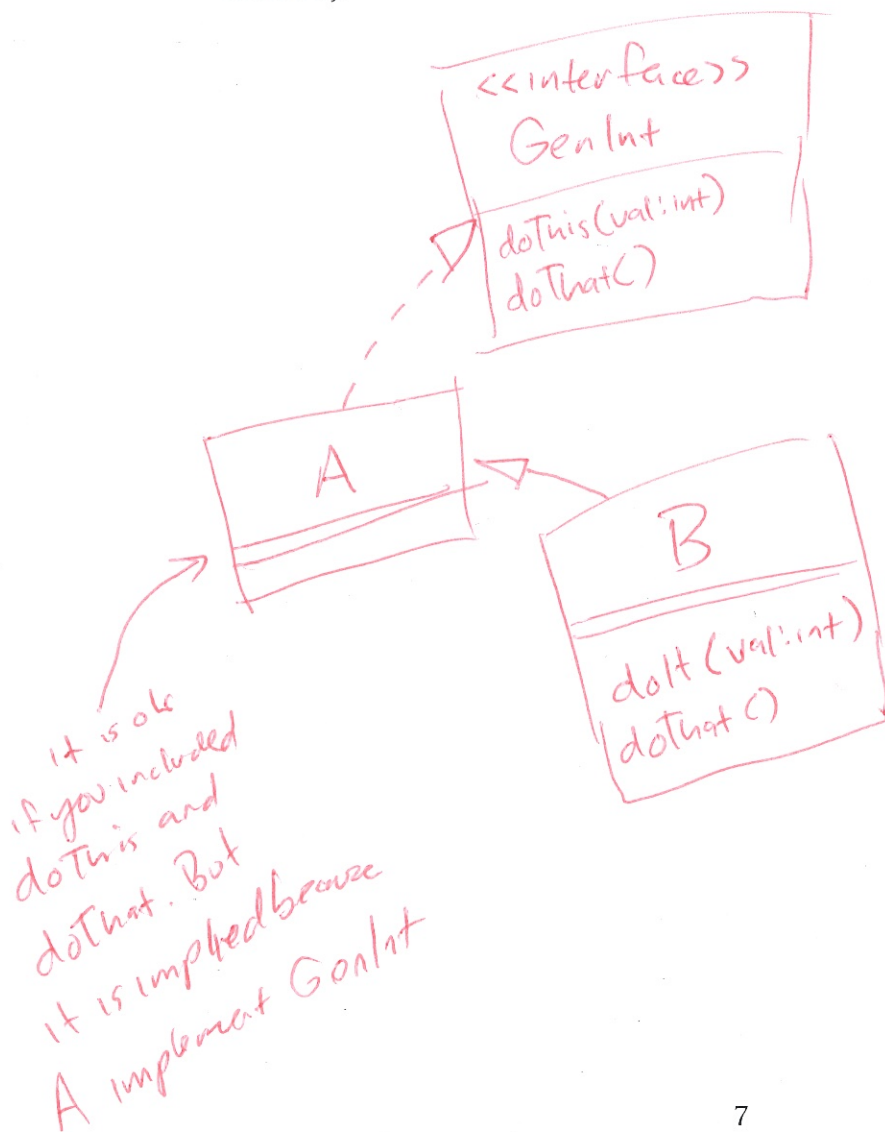
- ☒ T ☐ F Runner is not an abstract class
- T ☒ F Runner might be an Interface
- ☒ T ☐ F The method "pass" could be abstract in Mover
- ☒ T ☐ F If you had a function that took a Mover as a parameter, you could pass a Runner object and the code would compile
- T ☒ F If you had a function that took a Runner as a parameter, you could pass a Mover object and the code would compile
- ☒ T ☐ F Runner could have a subclass
- T ☒ F Runner's ^{direct}superclass must be Mover -
- ☒ T ☐ F If Runner had a method called faster() but Mover did not, you could call it by saying ((Runner)dude).faster()

6. Consider the following related declarations:

```
public interface GenInt {  
    void doThis(int val);  
    void doThat();  
}  
  
class A implements GenInt {  
    public void doThis(int val) {  
        System.out.print("A" + val);  
        doThat();  
    }  
    public void doThat() {  
        System.out.print("A2");  
    }  
}
```

```
class B extends A {  
    public void doIt(int val) {  
        System.out.print("B" + val);  
    }  
    public void doThat() {  
        System.out.print("B2");  
    }  
}
```

- 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:

```
A a1 = new A();
A a2 = new B();
B b1 = new B();
A c1 = new B();
GenInt g1 = new A();
GenInt g2 = new B();
```

For each line of code below, if the line results in an error, **circle** the appropriate error; otherwise, provide the full output in the provided blank (note that the use of “print” rather than “println” does not result in the output spanning multiple lines). 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	
b1.doThat();	runtime error	compile error <u>B2</u>
a2.doThat();	runtime error	compile error <u>B2</u>
a2.dolt(3);	runtime error	<u>compile error</u>
a1.doThis(6);	runtime error	compile error <u>A6 A2</u>
a2.doThis(9);	runtime error	compile error <u>A9 A2 A9 B2</u>
g1.doThis(1);	runtime error	compile error <u>A1 A2</u>
g1.doThat();	runtime error	compile error <u>A2</u>
((B) a1).dolt(5);	<u>runtime error</u>	compile error
((B) a2).dolt(5);	runtime error	compile error <u>B5</u>
g2.doThis(4);	runtime error	compile error <u>A4 B2</u>
g2.doThat();	runtime error	compile error <u>B2</u>
g2.dolt(6);	runtime error	<u>compile error</u>
c1.doThis(8);	runtime error	compile error <u>A8 B2</u>

7. (12 points) For this problem, use the frame technique we practiced in the course to trace the execution of the function call `elc(55,300)` in `main()` below. Then, answer the question at the bottom of the page. A frame template is provided for your reference.

```

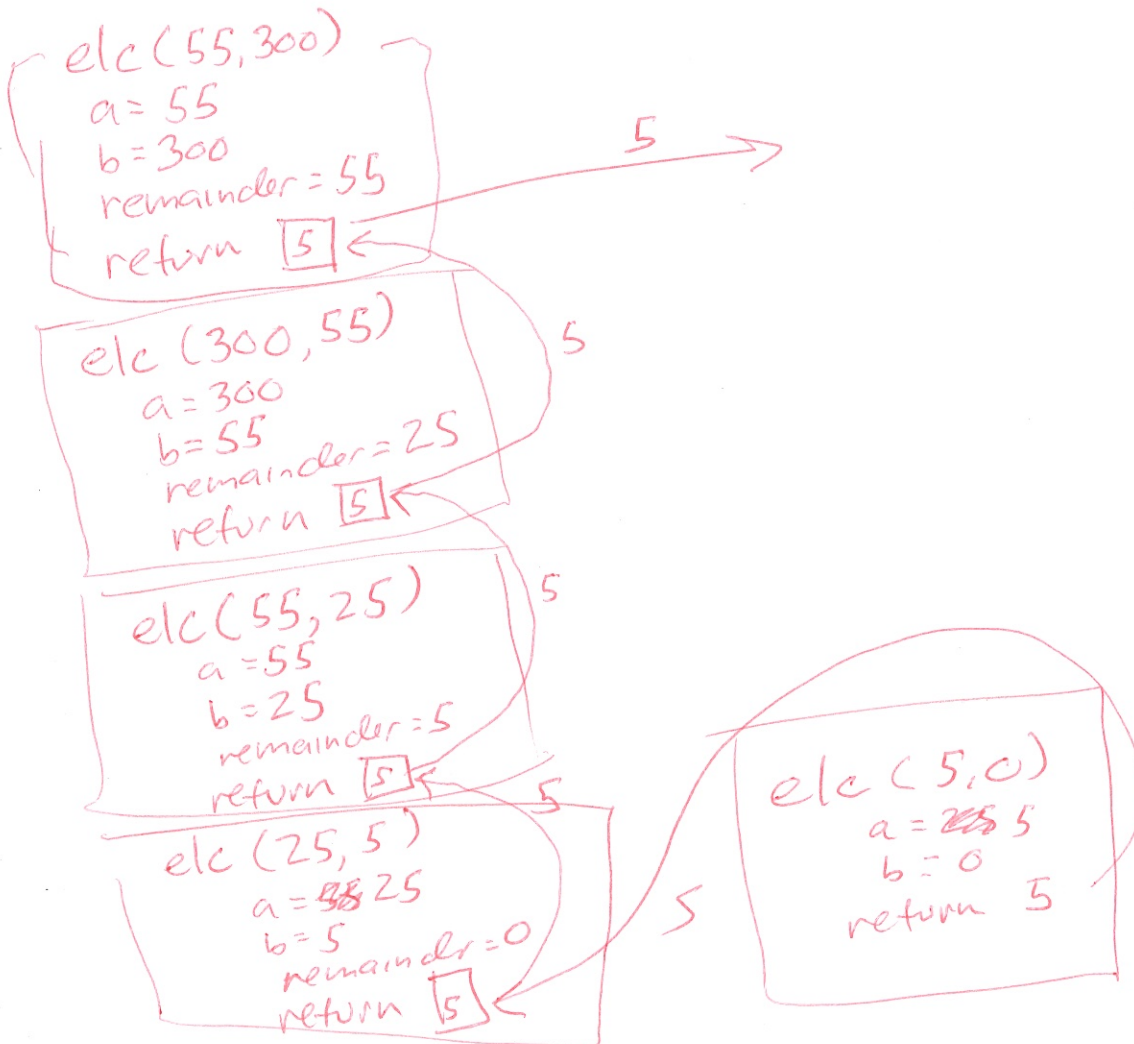
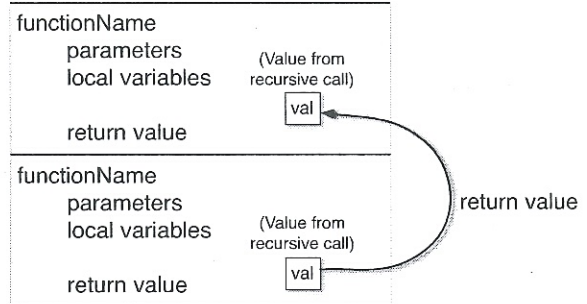
1 public class Exam2Fun {
2
3     public static int elc(int a, int b) {
4         if (b==0) return a;
5
6         // remainder when a / b
7         int remainder = a % b;
8
9         return elc(b,remainder);
10
11     }
12
13 }

```

```

public static void main(String[] args) {
    Exam2Fun obj = new Exam2Fun();
    System.out.println(obj.elc(55, 300));
}

```



For the code above, what would the final output be? 5