

Name: KEY Section: \_\_\_\_\_ CM: \_\_\_\_\_

CSSE 220---Object-Oriented Software Development  
Exam 2 -- Part 1, October 20, 2018

This exam consists of two parts. Part 1 is to be solved on these pages. If you need more space, please ask your instructor for blank paper.

*Allowed Resources on Part 1:* You may use a single sheet of 8.5" x 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.

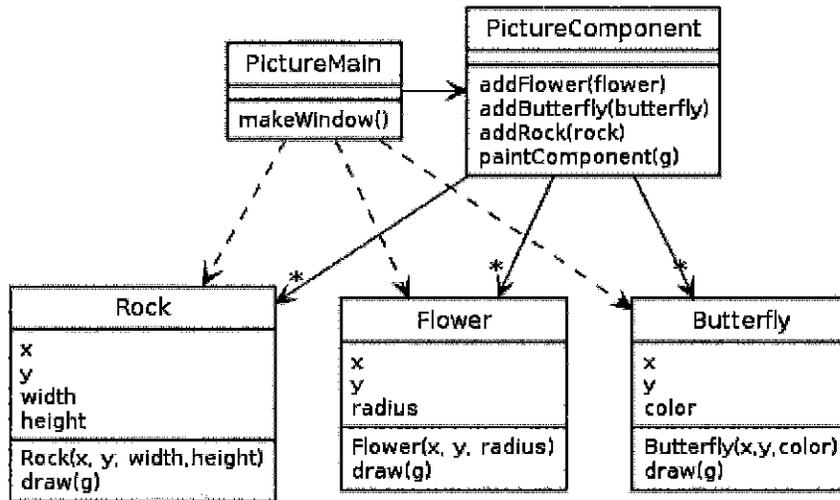
**You will have 50 minutes (the first Rose hour of class) to complete Part 1.  
Part 2 will be completed in the next class.**

Please, begin by writing your name on every page of the exam. We encourage you to skim the entire exam before answering any questions.

Problem	Points Possible	Earned
1	6	_____
2	8	_____
3	10	_____
4	5	_____
5	13	_____
<b>Paper Part Subtotal</b>	42	_____
<b>Computer Part Subtotal</b>	58	_____
<b>Total</b>	100	_____

1. (6 points) A UML diagram for a simple drawing system similar to the Scene assignment is shown below.

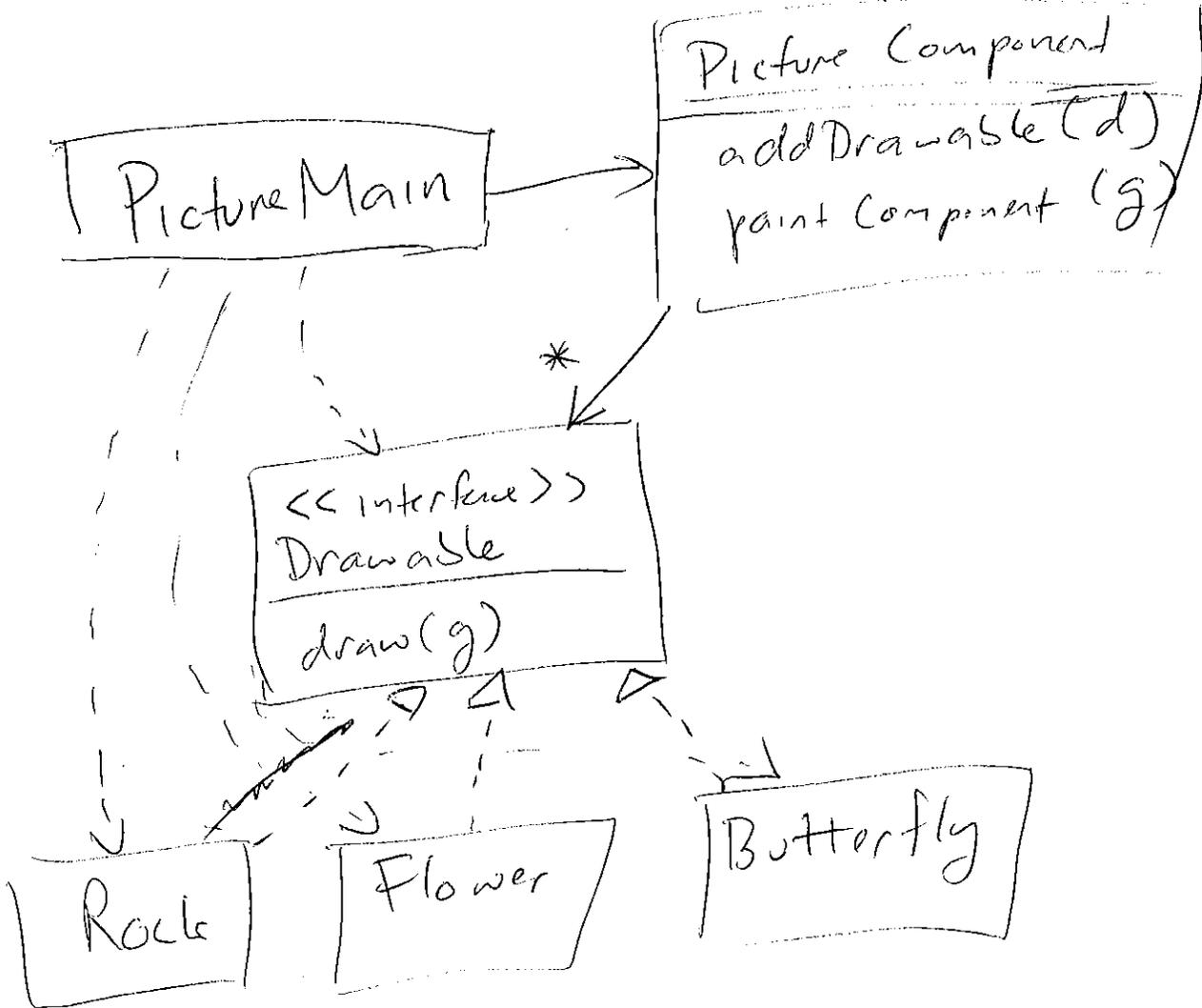
In this diagram, PictureMain constructs Rock, Flower, and Butterfly objects and adds them to PictureComponent. The paintComponent method of PictureComponent loops through all rocks, flowers, and butterflies in three loops and calls draw on each of them.



- a. (2 points) There is a problem with this design. Which of Encapsulation, Coupling, or Cohesion does the problem with this design seem related? In one sentence, explain what in the diagram suggests this problem exists.

This design has high coupling.  
You can see an excessive amount  
of connections between the  
classes.

- b. (4 points) Now, fix the problem you identified in the previous part with the introduction of an interface. Please do include all relevant dependency lines in your diagrams. To make your diagram shorter, feel free to just draw a rectangle with a Class Name and nothing else only if the class's fields and methods are unchanged from the original diagram above.

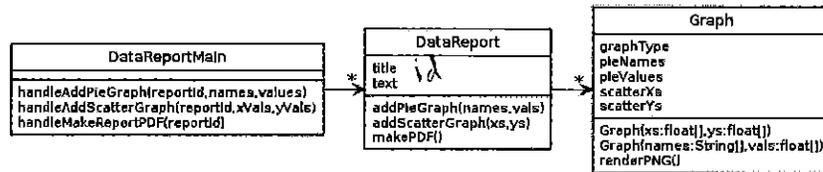


2. (8 points)

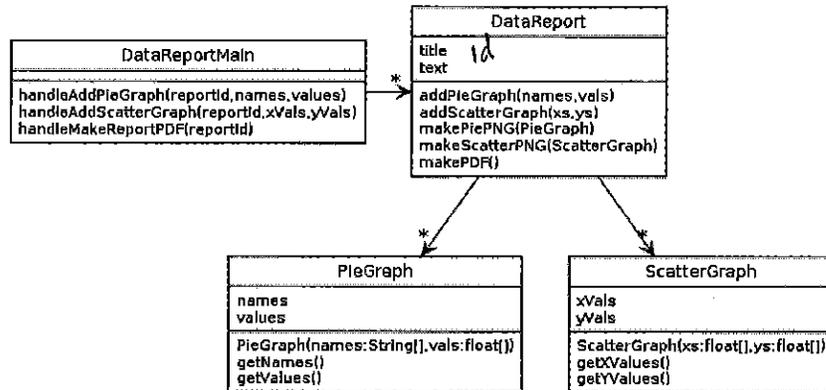
A particular system is designed to produce data reports for a company. A data report consists of an id, title, and some text followed by a series of pie graphs and scatter graphs. Each pie graph and scatter graph has its own data. For pie graphs, the data consists of a list of category names and values (e.g., "Misc expenses", 300.57). For scatter graphs, the data consists of pairs of x y coordinates. To produce a final data report, each graph must be rendered to a PNG image file and then the PNGs and the title/text must be combined into a final PDF file.

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



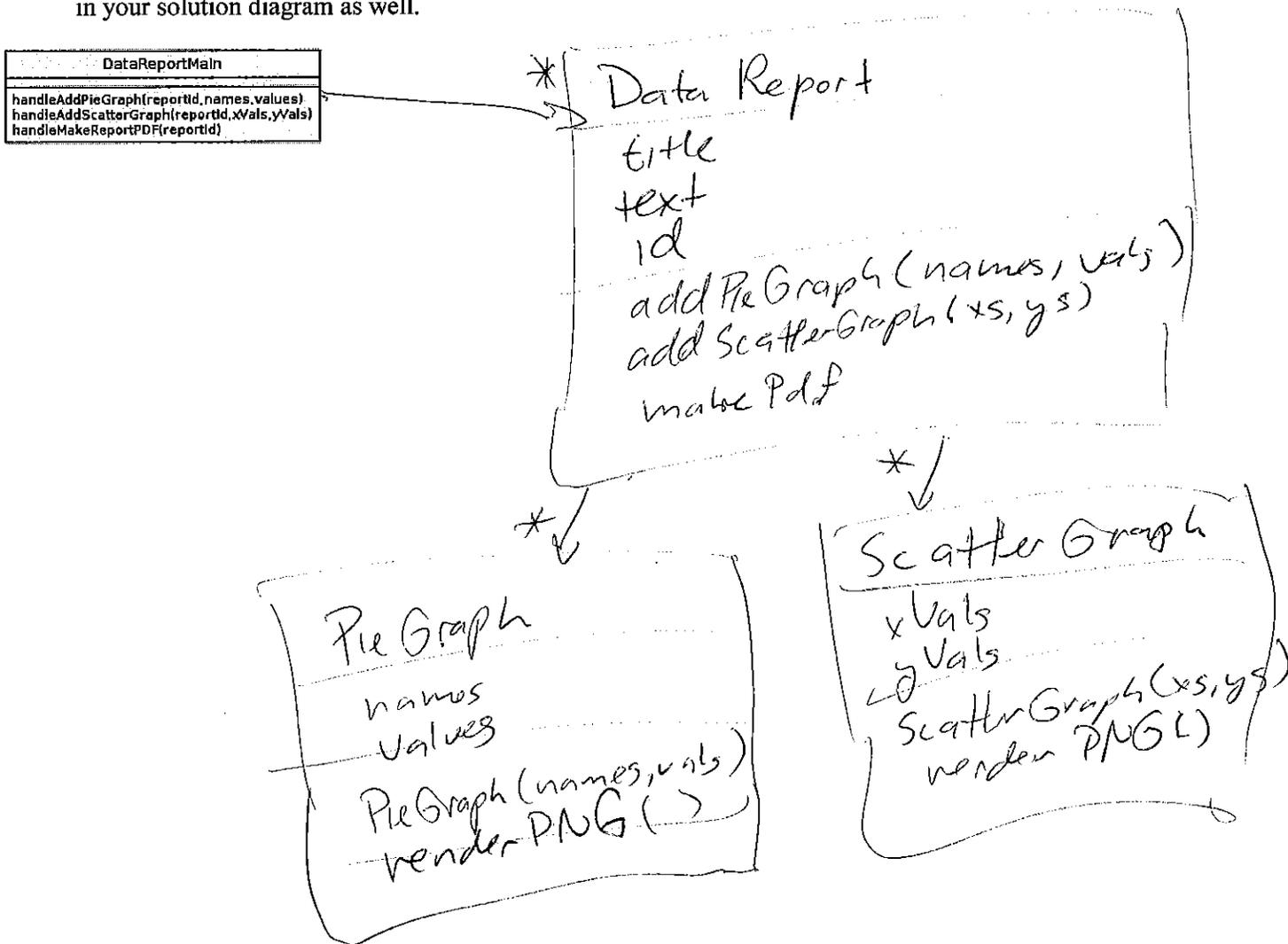
a. (2 points) Explain the problems with Solution A using your design principles.

Graph is conceptually doing 2 things.

b. (2 points) Explain the problem with Solution B using your design principles.

Data Report is asking too much from the graph classes, which are data classes.

c. (4 points) Make a UML diagram of your proposed solution to the problem. For your solution we have provided a DataReportMain to get you started. Feel free to omit any regular getter methods in your solution diagram as well.



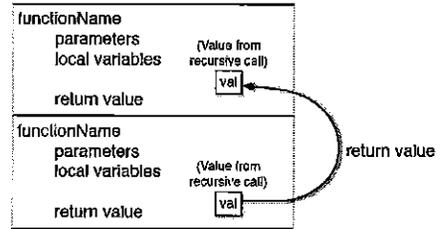
3. (10 Points)

For this problem, determine the output from this program by tracing the call to operation *fun* which appears on line 2.

```

1 public static void main(String[] args) {
2     System.out.println(fun("AXB", "CCX"));
3 }
4
5 public static String fun(String one, String two) {
6     if(one.isEmpty()) return "";
7     String rest = fun(one.substring(1), two.substring(1));
8     if(one.charAt(0) == 'X')
9         return rest + two.charAt(0);
10    else
11        return rest + one.charAt(0);
12 }

```



*fun* ("AXB", "CCX")  
 one = AXB  
 two = CCX  
 rest = BC  
 is Empty? No  
 X? No  
 return BC + A

*fun* ("XB", "CX") BC  
 one = XB  
 two = CX  
 rest = B  
 is Empty? No  
 X? yes  
 return "B" + "C"

*fun* ("B", "X")  
 one = B  
 two = X  
 rest = ""  
 is Empty? No  
 X? No  
 return B + ""

*fun* ("", "")  
 one = ""  
 two = ""  
 is Empty? yes  
 return ""

BCA →

4. (5 points)

Write what this code outputs.

```
class Q {
    public static double getPercentage(double one, double two) {
        if(two == 0)
            throw new IllegalArgumentException();
        if(one > two)
            throw new ArithmeticException();
        return (one/two)*100;
    }

    public static void printPercentage(double one, double two) {
        try {
            double result = -1;
            result = getPercentage(one, two);
            System.out.println("Percentage was " + result);
        } catch (IllegalArgumentException e) {
            System.out.println("IA error solved");
        }
        System.out.println("print done");
    }

    public static void someCode() {
        printPercentage(1, 0);
        printPercentage(2, 1);
        printPercentage(1, 2);

        System.out.println("code done");
    }

    public static void main(String[] args) {
        try {
            someCode();
        } catch (ArithmeticException e) {
            System.out.println("AE error solved");
        }
    }
}
```

Write your answer here:

IA error solved } 2.5 points if  
print Done } correct

AE error solved } 2.5 points if correct

5. (13 Points)

```

public interface Plastic {
    void methodA();
}

public class Gold implements Plastic {
    public void methodA() {
        System.out.println("GoldA");
    }

    public void methodC() {
        System.out.println("GoldC");
    }
}

public class Silver extends Gold {
    public void methodA() {
        System.out.println("SilverA");
        methodB();
    }

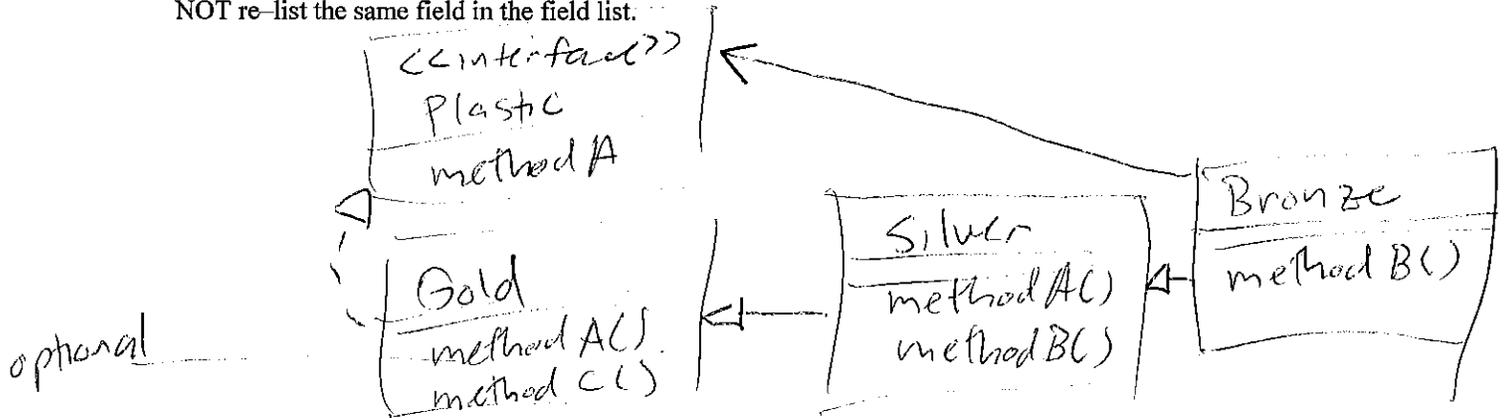
    public void methodB() {
        System.out.println("SilverB");
    }
}

public class Bronze extends Silver {
    private Plastic plastic;

    public void methodB() {
        System.out.println("BronzeB");
    }
}

```

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. (9 points) 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	
1. Gold x = new Silver();			
2. Silver x2 = new Bronze();			
3. Plastic p1 = new Plastic();	runtime error	compiler error	
4. Plastic p2 = new Gold();	runtime error	compiler error	nothing
5. x.methodA();	runtime error	compiler error	silverA silverB
6. x.methodB();	runtime error	compiler error	
7. ((Silver) x).methodB();	runtime error	compiler error	silver B
8. Silver y = x;	runtime error	compiler error	
9. x2.methodC();	runtime error	compiler error	goldC
10. ((Bronze) x).methodB();	<del>runtime error</del>	compiler error	BronzeB
11. ((Silver) x2).methodB();	runtime error	compiler error	silver A BronzeB
12. x2.methodA();			nothing
13. Silver y2 = x2;			

