

KEY

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

## CSSE 220---Object-Oriented Software Development

Exam 1 -- Part 1, September 19, 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. After you finish Part 1, please turn in your Part 1 answers and then open your computers and wait quietly for the programming exam review section of class to begin.

*Allowed Resources on Part 1:* You are allowed one 8.5" by 11" sheet of paper with notes of your choice. This section is *not* open book or open notes; and you are not allowed to use your computer for this part.

**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	9	_____
2	8	_____
3	4	_____
4	9	_____
5	5	_____
<b>Paper Part Subtotal</b>	35	_____
<b>Computer Part Subtotal</b>	65	_____
<b>Total</b>	100	_____

```

public class Crash {
    private String date;
    private String driverName;
    private boolean otherCarInvolved;

    public Crash (String date, String driverName, boolean otherCarInvolved) {
        this.date = date;
        this.driverName = driverName;
        this.otherCarInvolved = otherCarInvolved;
    }

    public void setName(String name) {
        this.driverName = name;
    }
}

public class Car {
    private String vin;
    private Crash [] crashes;
    private int crashCount;
    private static final int CRASH_MAX = 3;

    public Car (String vin) {
        this.vin = vin;
        crashes = new Crash[CRASH_MAX];
        crashCount = 0;
    }

    public void addCrash(Crash c) {
        crashes[crashCount] = c;
        crashCount++;
    }
}

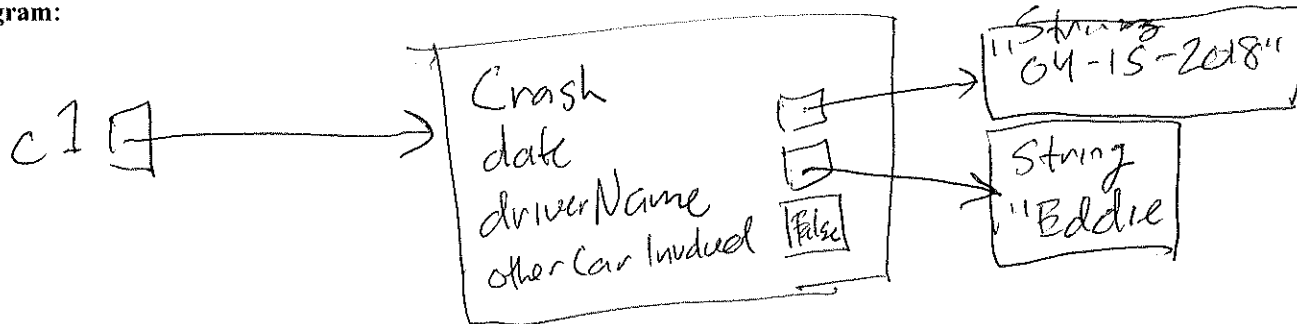
```

The next question refers to the classes on this page. The Javadocs are omitted to save space.

1. (9 points, 3 points each) Below are several code snippets that use the `Crash` and `Car` classes. For each snippet, first *draw a box-and-pointer diagram* (in the blank area below the snippet) showing the *final* result of executing the code. Changes to pointers or values for variables defined in the code snippets below CAN be shown but are NOT required if you produce the exactly correct final answer.

```
Crash c1 = new Crash("04-15-2018", "Eddie", false);
```

a) Diagram:

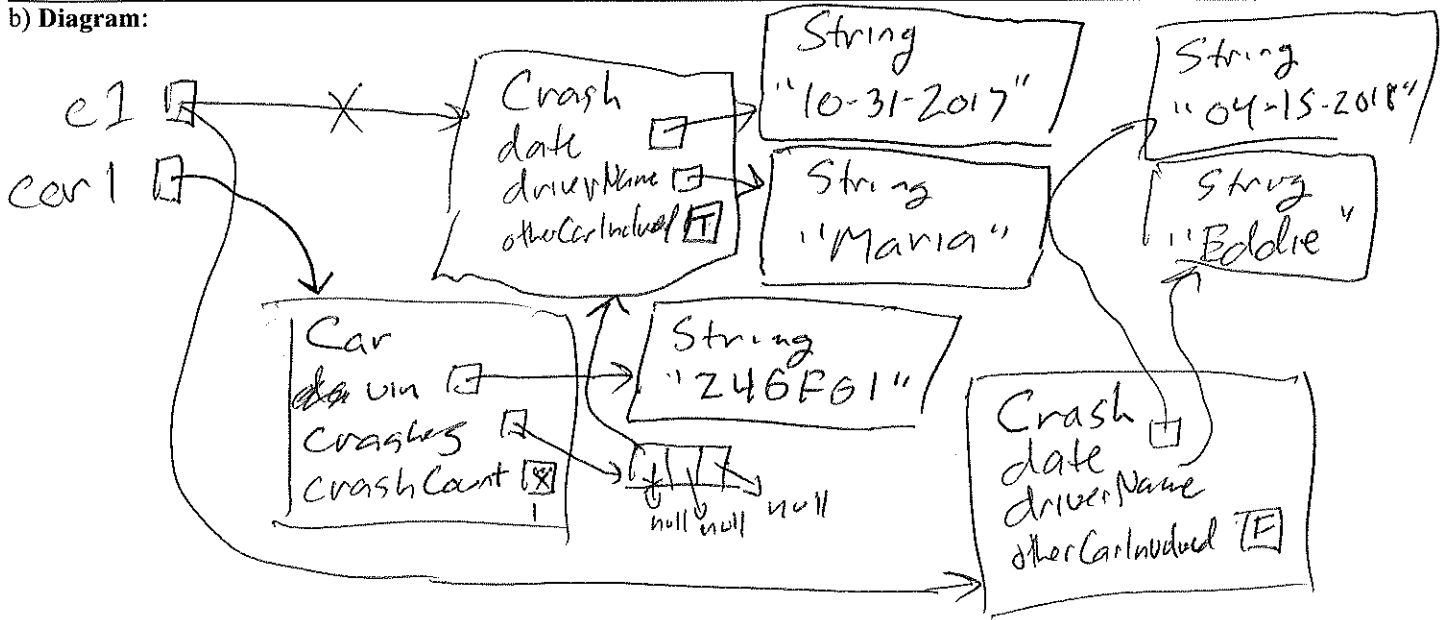


```

Crash c1 = new Crash("10-31-2017", "Maria", true);
Car car1 = new Car("2HGFG1");
car1.addCrash(c1);
c1 = new Crash("04-15-2018", "Eddie", false);

```

b) Diagram:

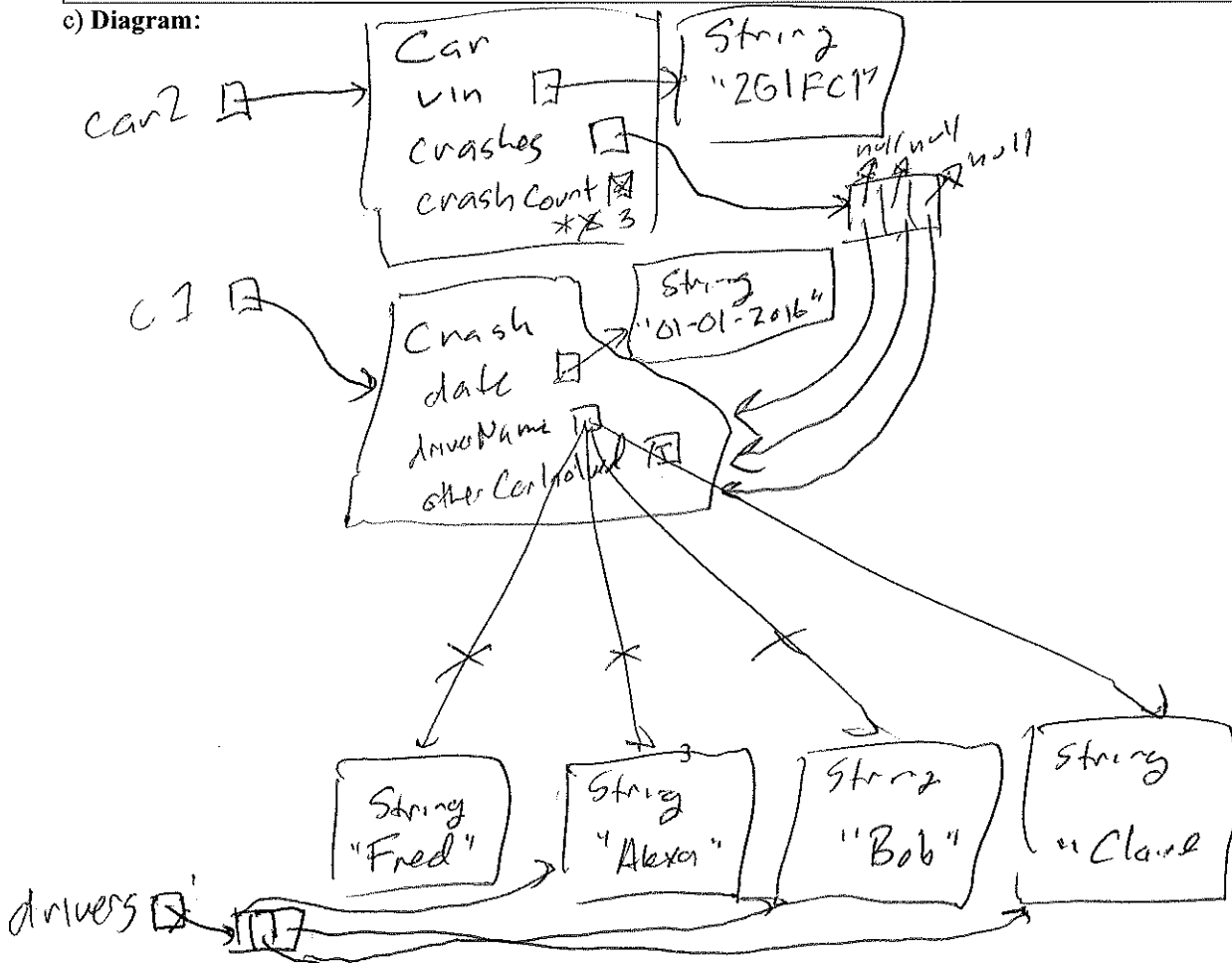


```

Car car2 = new Car("2G1FC1");
Crash c1 = new Crash("01-01-2016", "Fred", true);
String [] drivers = {"Alexa", "Bob", "Claire"};
for (String driver : drivers) {
    c1.setName(driver);
    car2.addCrash(c1);
}

```

c) Diagram:



2. (8 points, 2 points Each) Predict the output for each code snippet below.

- Each code snippet has no errors.
- You do not need to draw a diagram, but you may if it might help you.
- DO NOT TYPE THE CODE SNIPPETS FOR THIS QUESTION INTO ECLIPSE.
- If output spans multiple lines, write additional lines below the Output: line.

```
int a = 3;
int b = 2;
double d = 2.0;
System.out.println(a/b);
System.out.println(a/d);
```

(a) Output: \_\_\_\_\_

1  
1.5

```
HashMap<Integer, Integer> stuff = new HashMap<>();
stuff.put(1, 10);
stuff.put(2, 20);
stuff.put(10, 100);
stuff.put(100, 20);
int current = 1;
while(stuff.containsKey(current)) {
    System.out.println(current);
    current = stuff.get(current);
}
```

(b) Output: \_\_\_\_\_

1  
10  
100

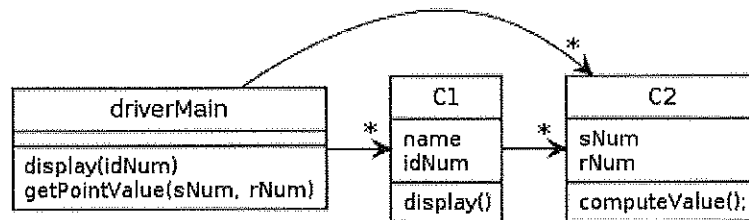
: \_\_\_\_\_

```
HashMap<String,Integer> map1 = new HashMap<>();
map1.put("X", 1);
HashMap<String,Integer> map2 = map1;
if(map1.equals(map2)) {
    System.out.println("equals1");
}
if(map1 == map2) {
    System.out.println("equals2");
}
map2 = new HashMap<>();
map2.put("X", 1);
if(map1.equals(map2)) {
    System.out.println("equals3");
}
if(map1 == map2) {
    System.out.println("equals4");
}
```

(d) Output: \_\_\_\_\_

equals1  
equals2  
equals3

3. (4 points) Write Java code that corresponds to the given UML diagram.



```

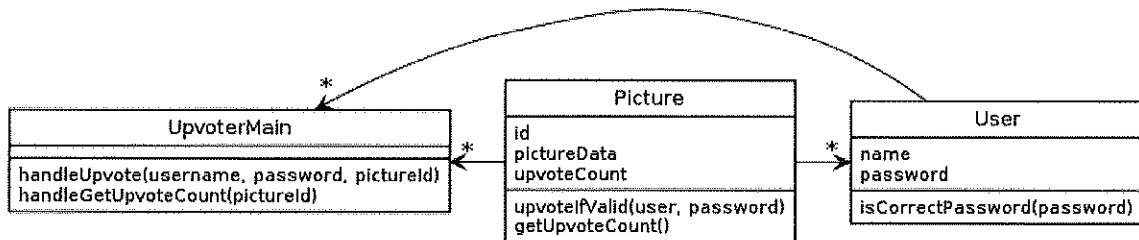
public class driverMain {
    private ArrayList<C1> c1s;
    private ArrayList<C2> c2s;
    public display(int idNum) {
        // ...
    }
    public getPointValue(int sNum, int rNum) {
        // ...
    }
}

public class C1 {
    private String name;
    private int idNum;
    private ArrayList<C2> c2s;
    public void display() {
        // ...
    }
}

public class C2 {
    private int sNum;
    private int rNum;
    public void computeValue() {
        // ...
    }
}
    
```

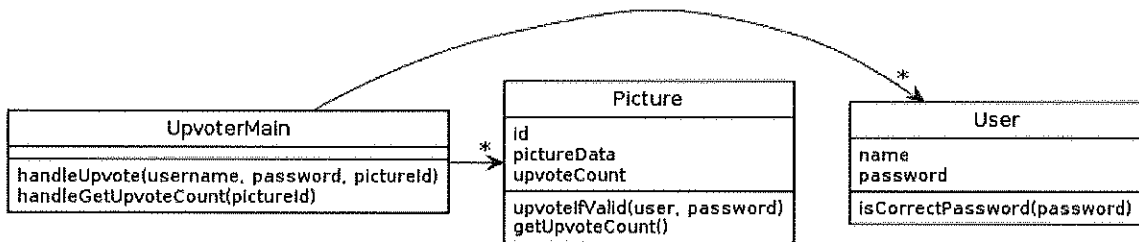
4. (9 points, 3 points each) On a particular website, users upvote funny pictures. When a user tries to upvote a particular picture, the system must ensure that the upvote is valid. To be valid, the given password must match the user's password AND the user must not have already upvoted this picture. If the upvote is valid, the picture's upvote count should be increased. Another command returns the upvote count for a particular picture.

(a) This solution does not function correctly. Explain why.



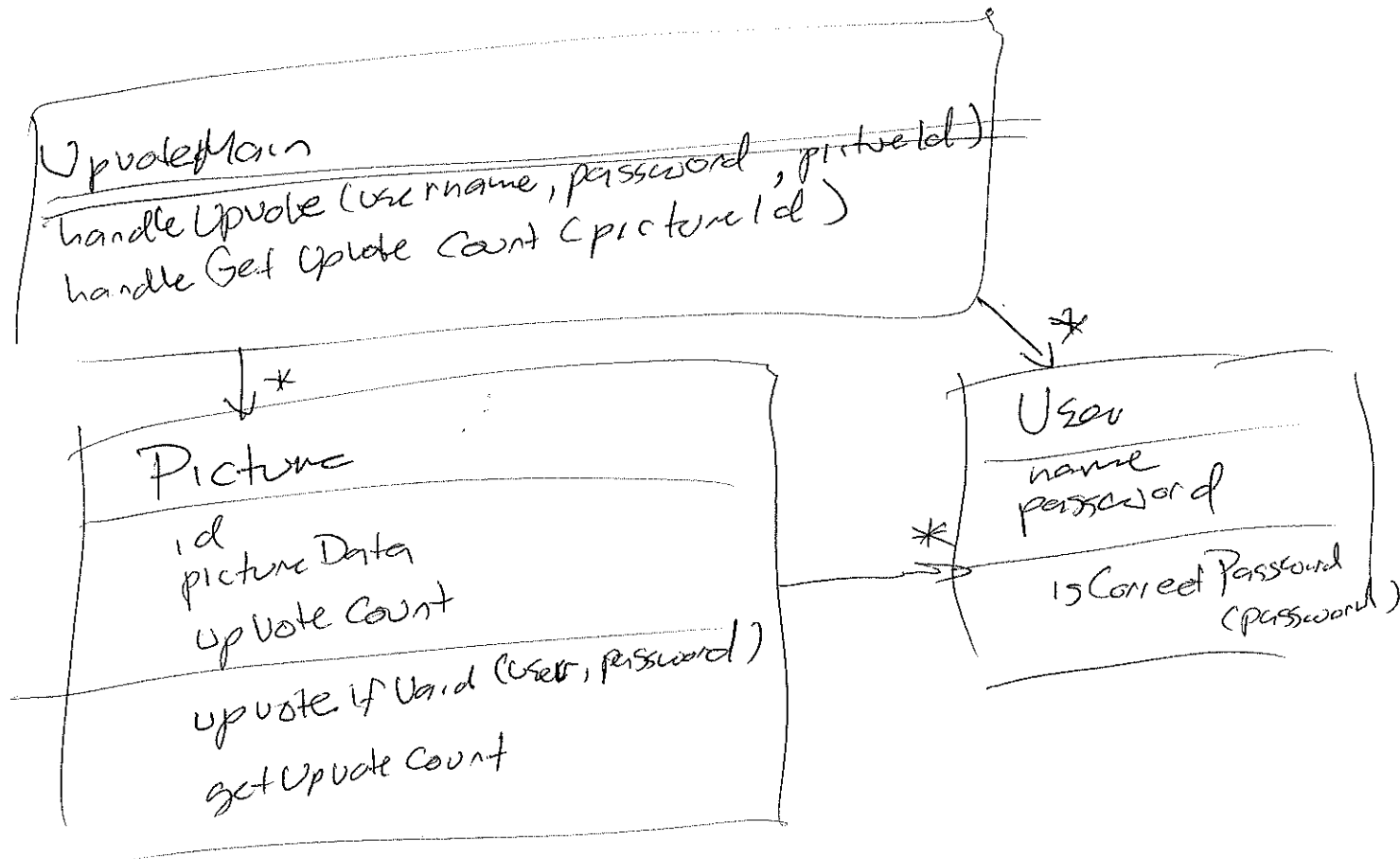
*Main has no access to Picture/User*

(b) This solution does not function correctly. Explain why.



*There is no way to tell if a user has upvoted a picture already in upvoteIfValid*

(c) Draw a UML diagram of your own proposed solution here.



5. (5 points) Write T next to the statements that are true in Java, F next to the statements that are false.

1. T The add function of the class ArrayList is definitely not static.
2. T The following is the correct way to check if a string is null:  
`if (myString == null) { /* do something */ }`
3. T If you have a class with a (standard) main function, you cannot access the fields of the class from the main function directly because the main function is static.
4. T If the function myFunc() in the class MyClass is static, you can call it like this:  
`MyClass.myFunc();`
5. F A class can only have one constructor.