University of Maryland College Park
Department of Computer Science
CMSC131 Spring 2024
Exam #2

FIRSTNAME, LASTNAME (PRINT IN UPPERCASE):

STUDENT ID (e.g. 123456789):

YOU MAY LOSE POINTS IF EXAM IS MISSING YOUR FULL NAME IN ALL CAPS AND ID # ON FIRST PAGE (SEE ABOVE). ALSO, YOU MAY LOSE POINTS IF EXAM IS MISSING YOUR DIRECTORY ID ON BOTTOM OF ODD PAGES. DO THIS AS SOON AS TIME STARTS. WE WILL NOT WAIT AT THE END FOR YOU TO WRITE THIS INFORMATION DOWN.

Instructions

- Please print your answers and use a pencil.
- Do not remove the staple from the exam. Removing it will interfere with the Gradescope scanning process.
- To make sure Gradescope can recognize your exam, print your name, write your directory id at the bottom of pages with the text DirectoryId, provide answers in the rectangular areas provided, and do not remove any exam pages. Even if you use the provided extra pages for scratch work, they must be returned with the rest of the exam.
- This exam is a closed-book, closed-notes exam, with a duration of 50 minutes and 100 total points.
- Your code must be efficient.
- Multiple choice questions have only one answer unless indicated otherwise.
- You don't need to use meaningful variable names; however, we expect good indentation.

Grader Use Only

#1	Problem #1 (Short Answers - 2pts each)	16
#2	Write a method	20
#3	Write a class	64
Total	Total	100

Problem #1 (Short Answers – 2 pts each)

1. (2 pts) To prevent falling through to the next **`case`**, use a ___________________ in a **`switch`** statement.
2. (2 pts) To permanently exit a looping construct before the looping condition is false, use a _____________________ statement.
3. (2 pts) Returning a reference to a mutable object in a get method creates a ____
4. (2 pts) Which statement is true?
   a. All **`static`** fields in a class also have to be **`final`**.
   b. The control expression of a **`switch`** statement can be a **`String`**.
   c. A local **`String`** variable will have a default value of **null**.
   d. A shallow copy occurs when you make an alias of an object by simply copying its reference over to the new variable.
5. (2 pts) Which statement is false?
   a. If the programmer writes no constructor for a class, it will still have a default constructor.
   b. The keyword **`this`** can be the return value of a non-static method.
   c. If the programmer writes no **`toString()`** for a class, it will be a compile error to call it.
   d. A non-static method of a class can access a **`static private`** field of the class.
6. (2 pts) Assume the following code fragment in a **`main`** method, what is the output (might be multiple lines)?

```
try {
      System.out.println("Line 1");
      int x = 5/0;
      System.out.println("Line 2");
}
catch(NullPointerException e){
      System.out.println("Line 3");
}
catch(Exception e){
      System.out.println("Line 4");
}
finally {
      System.out.println("Line 5");
}
```

7. (2 pts) In no more than 2 sentences, explain the role of automatic garbage collection in Java programming?

8. (2 pts) Assume the following code fragment in a **`main`** method, what is the output?

```
int [] x = {2,3};
String s = "hi";
System.out.println( (s.equals(null) ? "1": "5") + x[0] + x[1]);
```

Problem #2 (Write a method – 20 pts)

1. Complete the code for the method below. This method takes in an array of **`String`** called **`names`**, which you can assume to not be **`null`**, contain at least one **`String`**, and only have Strings (not null) with at least 2 characters. Simply return a **<u>new</u>** array with the same length as **`names`** in which at each index ***i*** of the new array you will have the id of the name found at index ***i*** of the **`names`** array. An id is made of the String "CS", the first letter of the name, the last letter of the name, and the index in which the name is found. For example, the name "mary" found at index 2 would have id CSmy2. See sample code below:

```
String names [] = {"joe", "john", "mary", "pete"};
System.out.println(Arrays.toString(makeIDs(names)));  // [CSje0, CSjn1, CSmy2, CSpe3]
```

The only thing allowed from the Java library is the `length` field of array, and from the `String` class, the `length` and `charAt` methods. **<u>You will lose significant points if you are calling other library methods.</u>** Making a new array or the literal “CS” is fine.

```
public static String [] makeIDs(String[] names) {
```

Directory id:

Problem #3 (Write a class)

Complete the implementation of a class called **`CafeReviews`** that represents 7 reviews of a cafe . You MAY NOT add any methods other than the ones specified below (not even private). Also, you MAY NOT add any additional instance or static variables. Local variables in your methods are fine. **No library methods are allowed.** Making a string, like String s = "hi", or an array and using the length field of the array is always allowed.

```
public class CafeReviews {
      private String name;
      private int [] reviews;

      private String makeStars(int num) {
            String retVal = "";
            for(int i = 1; i<=num; i++) {
                  retVal +="*";
            }
            return retVal;
      }

      public CafeReviews(String name) {
            //answer to P1 }

      public CafeReviews(CafeReviews orig) {
            //answer to P2 }

      public void setStar(int index, int star) {
            //answer to P3 }

      @Override
      public String toString() {
            //answer to P4 }

      public boolean oneOfEach(){
            //answer to P5 }
}

public class Driver {
      public static void main(String[] args) {
            CafeReviews r1 = new CafeReviews ("Terp Cafe");
            System.out.println(r1);
            r1.setStar(3, 4); //set review at index 3 to 4
            r1.setStar(0, 5); //set review at index 0 to 5
            r1.setStar(-1, 10); //set review at index -1 to 10 - nothing happens
            System.out.println(r1);

            CafeReviews r2 = new CafeReviews (r1);
            System.out.println(r2);
            r2.setStar(1, 2); //set review at index 1 to 2
            r2.setStar(6, 3); //set review at index 6 to 3
            System.out.println(r1);
            System.out.println(r2);
            System.out.println(r1.oneOfEach());
            System.out.println(r2.oneOfEach()); } }
```

Output

```
Terp Cafe: * | * | * | * | * | * | * |
Terp Cafe: ***** | * | * | **** | * | * | * |
Terp Cafe: ***** | * | * | **** | * | * | * |
Terp Cafe: ***** | * | * | **** | * | * | * |
Terp Cafe: ***** | ** | * | **** | * | * | *** |
false
true
```

1. This **constructor** should assign the parameter to the **`name`** field and create an **int** array with 7 elements (all assigned the number 1) that will be assigned to the **`reviews`** field.

```
public CafeReviews(String name) {
```

2. Define a **copy constructor** method for the class. Changes to the copy should not affect the original object and vice versa. Assume **`orig`** is not null.

```
public CafeReviews(CafeReviews orig) {
```

Directory id:

3. If the parameter **`index`** is between 0 (inclusive) and 6 (inclusive) and the parameter **`star`** is between 1 (inclusive) and 5 (inclusive), set the element of the **`reviews`** array at **`index`** to **`star`**. Otherwise, make no change.

```
public void setStar(int index, int star) {
```

4. It will return a **`String`** that has the **`name`**, followed by a **`:`**, followed by a space, followed by the content of the **`reviews`** array where each is separated by **`"  |  "`**. However, when printing the content of the **`reviews`** array, do not print the integer, instead print as many * as correspond to the integer. For example, the second line of output from the driver is:

```
Terp Cafe: ***** | * | * | **** | * | * | * |
```

Since at this point the first element (at index 0) of **`reviews`** has been set to 5 and the 4th element (at index 3) has been set to 4. Invoke the given **`makeStars`** as needed to turn an **`int`** to the correct number of *.

```
@Override
public String toString() {
```

5. The method returns **`true`** if at least one of each possible number (that is 1,2,3,4,5) is in the **`reviews`** array, and **`false`** otherwise. Hint: Think about how an array of **`boolean`** can help you keep track of the numbers in the **reviews** array.

```
public boolean oneOfEach(){
```

Directory id:

EXTRA PAGE IN CASE YOU NEED IT

LAST PAGE