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

FIRSTNAME, LASTNAME (PRINT IN UPPERCASE): KEY

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) An instance of a **`class`** is a (n) **object** .

2. (2 pts) The three ways to copy an object are reference copy, **shallow** copy, and deep copy.

3. (2 pts) Show how to declare an **`int`** array variable called **`myArr`** using the C/C++ syntax that Java still supports.

```
int myArr [];   //the [] must be after the name
```

4. (2 pts) Which statement is true?
   a. **`new`** is an operator in java.
   b. A boolean field would have the default value of **true**.
   c. Calling **`toString`**( ) without overriding the inherited definition (i.e. the default definition) will cause a compilation error.
   d. It is not possible to write a class without a default constructor.

5. (2 pts) Which statement is false?
   a. If you use the **`this`** to call one constructor from another, it should be the first statement in the constructor.
   b. It is not always safe to simply return the reference to a private field when coding the getters.
   c. The **`this`** keyword is not available in static methods.
   d. A **`switch`** statement in Java cannot have a String as the control expression.

6. (2 pts) Assume the following code fragment in a **`main`** method, what is the output (might be multiple lines)?

```
int x = 3;
try {
      System.out.println("Line 1");
      System.out.println("Line 2");
      x = x > 2 ? 2 : 2/0;
}
catch(ArithmeticException e){
      System.out.println("Line 3");
}
catch(Exception e){
      System.out.println("Line 4");
}
finally {
      if(x!=2)
            System.out.println("Line 5");
}
```

Line 1
Line 2

7. (2 pts) In **one statement**, show how to declare an **`int`** array called **`myArr`** with 3 elements that are initialized to 5, 6, and 7.

```
int myArr [] = {5,6,7}; or int [] myArr  = {5,6,7};
```

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

```
 int variable = 76;
int counter = 0;
while (variable < 86) {
    counter++;
    if (variable % 2 == 0) {
        variable++;
        continue;}
    variable++;
    if (counter > 5) {
        break; }
}
System.out.println(variable);
```

82

Problem #2 (Write a method – 20 pts)

Complete the code for the method below. This method takes in 2 **`String`** parameters. If they have the same length, just return **`null`**. Otherwise return a character array that has the characters of the longer string in reverse order. The only thing allowed from the Java library is the **`length`** field of array, and from the **`String`** class, the **`length`** and **`charAt`** methods. **You will lose significant points if you are calling other library methods.** Making a new array is fine.

```
import java.util.Arrays;
public class ArrayClass {
          public static char[] reverseLongerString(String str1, String str2) {


              if (str1.length() == str2.length()) {
                  return null;
              }

              String longerString = str1.length() > str2.length() ? str1 : str2;
              char[] reversedArray = new char[longerString.length()];

              for (int i = 0; i < longerString.length(); i++) {
                  reversedArray[i] = longerString.charAt(longerString.length() - 1 - i);
              }

              return reversedArray;

          }

          public static void main(String[] args) {
              char[] result = reverseLongerString("hello", "world!");
              if (result != null) {
                    System.out.println(Arrays.toString(result)); //   [!, d, l, r, o, w]
              }
          }
}
```


Directory id:

Problem #3 (Write a class)

Complete the implementation of a class called **`JokeTeller`** that represents a person that has some jokes . 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. You can use the following library methods as needed:

From **StringBuffer**: **`StringBuffer(), StringBuffer copy constructor, append(String str), toString()`**
From **String**: **`equals(Object anObject)`** and **`split(String regex)`**
From **Random**: **`Random()`** and **nextInt(int bound)**

Making a string, like String s = "hi", or an array and using the length field of the array is always allowed.

```
import java.util.Random;
public class JokeTeller {
    private static int jokeCount = 0;  // Keeps count of the number of jokes

    private String name;               // Name of the joke teller
    private StringBuffer jokes;        // Field to store jokes
    private Random random;             // For generating random indices

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

    public JokeTeller(JokeTeller other) {
        //answer to P2
    }

    public void addJoke(String joke) {
       //answer to P3
    }

    public String tellRandomJoke() {
        //answer to P4
    }

    @Override
    public boolean equals(Object obj) {
      if (obj == this)
            return true;
      if (obj == null || getClass() != obj.getClass())
            return false;
      //answer to P5

    }

    @Override
    public String toString() {
      return "JokeTeller{name='" + name + "', jokes=\n" + jokes.toString() + "}";
    }

    public static int getJokeCount() {
        return jokeCount;
    }
```

```
 public static void main(String[] args) {

    JokeTeller teller = new JokeTeller("Alice");
    teller.addJoke("Why don't scientists trust atoms? Because they make up
everything!");
    teller.addJoke("Why did the scarecrow win an award? Because he was
outstanding in his field!");

    System.out.println(teller);
    System.out.println("Random Joke: " + teller.tellRandomJoke());
    System.out.println("Total jokes: " + JokeTeller.getJokeCount());

    JokeTeller tellerCopy = new JokeTeller(teller);
    System.out.println(tellerCopy);
    System.out.println("Total jokes: " + JokeTeller.getJokeCount());
    System.out.println("Are both tellers equal? " + teller.equals(tellerCopy));


    tellerCopy.addJoke("Why did the bicycle fall over? Because it was two-
tired!");
    System.out.println(tellerCopy);
    System.out.println("Total jokes: " + JokeTeller.getJokeCount());
    System.out.println("Are both tellers equal? " + teller.equals(tellerCopy));


    }
}
```

Output

```
JokeTeller{name='Alice', jokes=
Why don't scientists trust atoms? Because they make up everything!
Why did the scarecrow win an award? Because he was outstanding in his field!
}
Random Joke: Why did the scarecrow win an award? Because he was outstanding in his field!
Total jokes: 2
JokeTeller{name='Alice', jokes=
Why don't scientists trust atoms? Because they make up everything!
Why did the scarecrow win an award? Because he was outstanding in his field!
}
Total jokes: 4
Are both tellers equal? true
JokeTeller{name='Alice', jokes=
Why don't scientists trust atoms? Because they make up everything!
Why did the scarecrow win an award? Because he was outstanding in his field!
Why did the bicycle fall over? Because it was two-tired!
}
Total jokes: 5
Are both tellers equal? false
```

Note: Assume that calls in main such as:
`teller.addJoke("Why don't scientists trust atoms? Because they make up everything!");`
are on one line so they compile.

Directory id:

1. This **constructor** should assign the parameter to the **`name`** field and invoke the appropriate default constructors to assign objects to the other 2 fields. Assume **`name`** will not be null.

```
public JokeTeller(String name) {

this.name = name;

this.jokes = new StringBuffer();

this.random = new Random();

}
```

2. Define a copy constructor method for the class. Changes to the copy should not affect the original object and vice versa. Assume **`other`** is not **`null`**. For the new **`random`** field you can just invoke the default constructor to assign a new **`Random`** object. Update the **`jokeCount`** field as appropriate. Each copy of a joke counts toward the **`jokeCount`**. **`jokeCount`** is the total number of jokes for all jokers, not an individual joker.

```
 public JokeTeller(JokeTeller other) {

this.name = other.name;

this.jokes = new StringBuffer(other.jokes);

this.random = new Random(); // Separate Random instance

jokeCount+=jokes.toString().split("\n").length;



}
```

3. Append the **joke** parameter (assume it is not **null**) to the **jokes** field followed by appending a newline. Update **jokeCount**.

```
public void addJoke(String joke) {

jokes.append(joke).append("\n");

jokeCount++;

}
```

4. Use the **split** method to create an array of strings where each element is a joke. If the length of the array is zero, return **"No jokes available!"** Otherwise use the **nextInt** method to pick a random joke within the bounds of the array and return it.

```
public String tellRandomJoke() {

String[] jokeArray = jokes.toString().split("\n");

if (jokeArray.length == 0) {

return "No jokes available!";

}

int index = random.nextInt(jokeArray.length);

return jokeArray[index];

}
```

5. Two **JokeTeller** objects are equal if they have the same name and jokes.

```
public boolean equals(Object obj) {

        if (obj == this)
                return true;
        if (obj == null || getClass() != obj.getClass())
                return false;

JokeTeller other = (JokeTeller) obj;

return this.name.equals(other.name) &&
this.jokes.toString().equals(other.jokes.toString());

}
```