Due: Tuesday, Sept 29th by 9pm. Late turn-in is 9pm, Thursday, Oct. 1.

Name your program HW2.java. This program will contain many methods, detailed in the sections below. You will need to invoke your functions from the main method in your program. You may comment methods out during your development process. When you submit your work, make sure all methods are uncommented so that we can grade them. We only grade work which is uncommented and able to be run.

Strategy: I HIGHLY recommend you write one method at a time, test it, and verify it works before moving on to another method.

1. Background:
   In this assignment you will create variations of heads for totem poles. You'll do this by creating variations of eyes, hair, noses, mouths, hats, etc and then create a head by putting all the parts together. The heads will be generated by using methods and print statements. You will then build random totem heads. Finally, you'll build a totem pole composed of 3 random heads.

   What is a totem pole? According to Wikipedia Totem Poles are "monumental sculptures" largely from the Pacific Northwest.

2. Totem Poles in CS170:
   The goal of this assignment is to introduce students to the notion of encapsulating different functionalities in separate methods. We do so by writing a program that (in the end) will print a totem pole to the terminal screen.

   This assignment is designed so that you'll be able to demonstrate your understanding of writing and calling methods, returning values from methods, and adhering to design standards.

   It's also designed so that you can demonstrate some creativity, one of the important aspects of creating computational artifacts. While you're being creative, you'll need to design your program so that it can interact with other creators, part of understanding how programs interoperate when they're scaled up beyond being used by one person.

   First of all let's define what a totem pole is. A totem pole consists of several 'heads' each of which on its own has different parts. Moreover, each totem 'head' consists of:
   - The top of the head - e.g. its hair.
   - The middle of the head - e.g. its eyes/ears/nose.
   - The bottom of the head - e.g. its nose/mouth/chin

   Example:
   Below you can see two examples of a totem head written with ASCII characters
Again, do not confuse a totem head with a totem pole! A totem pole is 3 totem heads, one on top of the other.

3. **Builder Methods:**
   In this part you will write Java methods that implement different parts of a head. Each method you write must return a String that is 15 characters wide and as many tall as you want it to be. (Note the examples above are not 15 characters wide. They are only for demonstration purposes.)

3.1. **Basic Parts:** Initially you need to write one Java method for each of the following parts:
   a) Hair
   b) Side
   c) Eyes
   d) Mouth
   e) Chin
   f) Nose

   Each method should be a single word name, corresponding to the part of the head listed above. Each method should return a String that - given some imagination - looks like the correlated part of the head.

   An example for the hair function would be:
   ```java
   public static String hair() {
       return "|||||||||||||||
   }
   ```

3.2. **More Diversity**
   Choose at least 3 parts for which you already wrote a method in section 2.1, **EXCEPT** for the nose. For each of these parts write an additional builder method which implements the same part of the totem head but in a different way.

   An example of the second function that implements the hair part would be:
   ```java
   public static String hair2() {
       return "\\\\\\\\\\\\\\\\\\\\\\\\\\":
   }
   ```

   Note here that we are using escape characters like in Lab3 to achieve the 15 characters wide requirement.

4. **Caller Methods**
Now that we have the basic ingredients, we need to somehow combine them into building a complete totem pole. In this section you are needed to write two different methods, both of which call the builder methods that you wrote in section 2.

4.1. `totemHead()`: This function should print a random totem head. Use the `Math` class and its `random` method provided by Java. Based on random values that you get from this method, choose different head parts that you wrote in parts 2.1 and 2.2 in order to build each head.

An example of how `totemHead()` makes a probabilistic decision:

```java
public static void totemHead() {
    //Math.random() generates a random double [0.0-1.0)
    double r = Math.random();
    int rand = ...;
    //initialize w/ a random int of 0 or 1
    //see (a) below for details on how to
    //complete this using the variable r.
    System.out.println(rand);
    if (rand == 0) {
        System.out.println( hair() );
    } else {
        System.out.println( hair2() );
    }
    // more code here which generates another random number and
    // continues printing other head parts
}
```

a) Calculating a random integer in `totemHead()`:

You can use the `Math.random()` function to give you a random double value between 0.0 (inclusive) and 1.0 (exclusive). How can we turn this into an integer? If we simply cast the value to an int, we will always end up with 0. What happens if we multiply the random double value by 2.0? Well, we would end up with a value somewhere between (0.0*2.0) and (0.999999...*2.0). Stated another way, we would end up with a double value between 0.0 and 1.9999999. What would happen if we cast that value to an integer? Think about how you could generalize the math to to generate a random integer value of 0, 1 or 2. This is how you will generate a random integer value for your code when all the `Math.random()` function gives us is double values.

b) Include code which makes `totemHead()` sometimes include a nose, but sometimes not.

Again, you can use the random number concept explained above, but your logic might look something like:

```java
//generate random integer value of 0 or 1 for rand
if(rand == 1) {
    System.out.println( nose() );
}
//otherwise, don't call nose() method so no nose is printed.
```
4.2. `totemPole()`: This function must print a totem pole that consists of three different heads. Because your `totemHead()` method contains random elements, each head should be different than the other two heads. This method should not return any value, nor does it need any arguments.

5. **Comments**: You should include appropriate Javadoc comments for all of your methods (except the `main` method). Include a description of what visual effect the builder methods have. For example, descriptions like "This method makes curly hair" or "This method makes a very surprised mouth" are appropriate.

6. **Submission**:
   6.1. Be sure to include your name, Emory userID, section number, and collaboration statement as comments at the top of your file. Failure to do so will result in a 10 point deduction.
   6.2. Submit your `HW2.java` file to Blackboard by attaching the file to the HW2 assignment. Do not copy/paste your work into a text box. Be sure to click "Submit" to ensure your work is submitted for grading. We cannot view your work if you only click "Save" and save a draft. Do **NOT** submit `HW2.class`.
   6.3. Due deadline is Tuesday, Sept 29th by 9pm. Late turn-in is 9pm, Thursday, Oct. 1 for 10% (10 points) off.

*This assignment is based off an assignment originally developed in Duke University's CompSci 6 course.*