Name:

Instructions

This is a computer-based programming exam.

The **only** things you should have open on your computer are:

- VSCode, open to an empty folder
- optionally, the Java Documentation, at docs.oracle.com (https://tinyurl.com/cmsc150-docs)

Reminders

Here are some tips and reminders if you find yourself stuck or confused.

- **Start small.** See if there is any part of the problem where you can get a "foothold". Often staring at code that sort-of works is better than staring at a blank screen.
- **Test intermediate steps.** It will be easier to find problems if you are printing and testing outputs along the way to make sure your program works in smaller increments before you test it all at once.
- **Write it out.** Feel free to sketch things out on your paper if you feel like you know how it *should* work, but can't quite make it make sense in code. Sometimes stepping away from the screen for a moment helps.
- **Simplify.** Sometimes we get lost in our own code and get deep into a problem, then realize we are confused. You can always make a new program and try something new.
- **You will have other opportunities to demonstrate mastery.** This is not your one and only chance to succeed in this class, and this isn't meant to be high-stakes.

Turn In

Turn in your exam on Canvas under Exam 2. Turn it in as two .java files.

Problem Statement

In many tabletop role playing games, the player rolls dice to determine the outcome of a certain action. These dice have different face values – some have 6 sides, some have 12, some have 20, etc. The sum of the dice is the result of the roll.

You will simulate **rolling and summing** *a k*-sided die *n* times. *k* and *n* should be taken in either from a scanner or as command-line arguments.

You should implement the dice as a **class** called `Dice` with an **attribute** `numFaces` to set the number of faces of the dice, and an **instance method** `roll()` to simulate a random dice roll (implementation is your choice). When a Dice object is **printed** using `System.out.print()`, it should display the letter "d" followed by the number of faces; for example, a 10-sided die when printed should display as `d10`.

You can and should include other attributes and methods as you need them.

The actual rolling and summing of dice should happen in a second program called `Roll.java`. In this program, you should create **only one dice object** which you can re-roll as many times as you need.

Your program, when run, should output the **number of dice rolled and their number of faces**, the **resulting sum,** and the **individual rolls**. Those rolls need not be in any particular order, but should all be printed on a **single line.**

Example Input (command-line args):

```
java Roll 7 4
```

Example Output:

```
7d4: 20 (3, 2, 2, 2, 3, 4, 4)
```

Note that this problem is not autograded, so you do not have to follow this print layout exactly.

Assessment

Mastery

- □ Program takes input either with a scanner or as command-line arguments
- □ Dice is implemented as a class with attribute numFaces and method roll(), along with any other necessary attributes and methods
- □ When printed, a Dice object is displayed as “d” followed by the number of faces.
- □ There is exactly one Dice object instantiated in the Roll.java program
- □ Program accurately computes the resultant sum
- □ Program accurately prints the number of dice, the faces, the sum, and the individual rolls in a SINGLE line
- □ Logical and informative names for variables and methods
- □ Clear and consistent style and spacing

Proficiency

- □ Program produces the correct output but is missing one of the other mastery requirements, **OR**
- □ Program is missing up to two of the mastery requirements, AND you write up a short (3-4 sentence) explanation of where you got stuck and include a sample output for your code vs. what it should be outputting, **OR**
- □ Program is missing up to two of the mastery requirements, AND you give a plain-English explanation of the algorithm you would write but cannot figure out how to write in Java

Novice

- □ Program produces the correct output but is missing two or more of the other mastery requirements without written work