# Random Number Generation

**The goal for this exercise** is to understand how you can generate random numbers, efficiently.

Within the provider starter project, you’ll notice that there is a class named something like RandomNumbersBasic. Within it, you will need to fill in the code for the runExercise() method so that the code generates an integer value that lies between 1 and 6, inclusive, and then prints that number out. It simulates the rolling of a die.   
  
For this exercise, you may generate these random numbers using any version of the Next method that you want – if you want to use the version that has a lower & upper bound, feel free. If you want to use the version that doesn't, and then you want to ensure that the numbers fall within the range of 1-6 using the modulus operator, feel free to do that, too.

**What you need to do for this exercise:**

1. In RandomNumbersBasic.RunExercise, start by generating a single, random number between 1 and 6 (including, potentially, 1 or 6). Print this number out.
2. Next, generate random numbers several hundred times using a loop. Print each random number to the screen.  
   1. First, make sure that you're getting numbers that look pretty random.
   2. Second, try moving the following line of code INTO the loop, so that it gets executed each and every time through the loop:  
        
      Random dieToRoll = new Random();  
      1. What output do you see?  
         You must include a short comment (1-2 sentences) that concise explains what you see, and correctly explains why the program is behaving in this way.  
           
         You are encouraged to discuss this with your peers.
   3. Thirdly, undo the change from the prior part (so that your program generates random numbers again), and modify your code so that it counts the number of times that each value is randomly generated. Once the loop is done, print out those values. Are the counts roughly equal? What if you call it thousands of times? Tens of thousands of times?  
        
      An example of the output you might generate is:  
      Generated 1 a total of 199 times  
      Generated 2 a total of 178 times  
      Generated 3 a total of 201 times  
      Generated 4 a total of 222 times  
      Generated 5 a total of 213 times  
      Generated 6 a total of 187 times  
      1. Do this however you want. Simply having 6 variables (one to keep track of the number of times "1" was generated, another to keep track of the number of times "2" was generated, etc) is fine. If you want to use an array (and know how to), feel free to do that, instead.
   4. Finally, modify the program so that instead of creating the Random object inside the RunExercise function (which will be inefficient, memory-wise, because you’ll need to create a brand-new Random every time you call this method), instead move the variable declaration and object creation to the class, like so:  
        
      class RandomNumbersBasic  
      {  
       Random dieToRoll = new Random(); // MOVE TO HERE!!  
        
       public void RunExercise()  
       {  
       ~~Random dieToRoll = new Random();~~ // DELETE FROM HERE!!  
        
      This means that dieToRoll will be an **instance variable,** and when the RandomNumbersBasic object is created (in the **Main** method), only 1 Random object will be created. This is much more memory-efficient than creating a brand-new Random object each time that you call the RunExercise method – make sure that you use this ‘Random object as an instance variable’ pattern whenever you need to use Random numbers!
      1. (In this case, since we’re only calling RunExercise just once, it really doesn’t matter which way we go. But we want to practice this more-efficient pattern so that when our programs get big enough for it to matter, we will already be used to doing this)