

Random class

<https://csci-1301.github.io/about#authors>

May 22, 2024 (01:30:41 PM)

Contents

1	Generating Random Numbers	1
2	Manipulating Two Arrays	2
3	Pushing Further (Optional)	3
3.1	Cryptographically secure random numbers	3

This lab serves multiple goals:

- To illustrate how programs can generate random numbers,
- To introduce you to using existing libraries,
- (Optional) to understand what a cryptographically secure random number generator is and why it matters.

1 Generating Random Numbers

The `Random` class from the C# standard library can be used to generate random numbers in any given range. In this lab, you will practice using the `Random` class.

Start by reading the corresponding chapter in the lecture notes¹, then create a new project and practice generating and displaying to the screen different random numbers:

1. Generate any random integer
2. Generate a random integer between -10 and 10 including these boundary values
3. Generate a random double

Note you only need 1 instance of the `Random` class to generate these numbers.

Solution:

```
Random rand = new Random(); // Creation of a random number generator.
Console.WriteLine("A random number:" + rand.Next()); // This is any random
    ⇨ (int) number.
Console.WriteLine("A random number between -10 and 10:" + (rand.Next(21)-10));
    ⇨ // This number will be between 0 and 20, then we subtract 10 from it.
```

¹<https://csci-1301.github.io/book.html#random>

```

Console.WriteLine("A random number between -10 and 10:" + rand.Next(-10, 11));
↪ // Alternate solution
Console.WriteLine("A random double:" + rand.NextDouble()); // This is any
↪ random (double) number.

```

Execute the program a few times to make sure the outputs are different each time.

Once you have successfully generated the 3 random numbers described above, add the following enhancements to the program:

1. Generate any random integer *5 times*
2. Generate a random integer between -10 and 10 including these boundary values, *10 times*
3. Generate a random double, *5 times*

Execute the program again, a few times, to make sure these values change on each execution.

2 Manipulating Two Arrays

This problem combines random number generation with arrays. Using a Random object, write a program that:

1. declares two arrays of `int` of size 8,
2. initializes the values of the first array with random numbers between 0 and 9,
3. initializes the values of the second array with random numbers between 0 and 9,
4. displays the contents of the two arrays in a table, and for each index, a letter indicating whether the first array "won" or "lost" a contest with the second array:
 - "W" if the value in the first array is greater than the value in the second array
 - "T" if they are equal, and
 - "L" if it is less

An example execution of this program would display:

0	8	L
5	3	W
3	3	T
1	2	L
3	1	W
9	0	W
9	0	W
1	5	L

In this example, the first array contains "0 5 3 1 3 9 9 1" and the second contains "8 3 3 2 1 0 0 5".

3 Pushing Further (Optional)

3.1 Cryptographically secure random numbers

Random number generation is only pseudo-random, meaning these are algorithmically generated numbers that approximate a sequence of truly random numbers. Using the default `Random` class is not recommended for applications that need cryptographically secure random numbers (e.g, to generate suggested passwords).

When an application needs cryptographically secure random numbers, `RandomNumberGenerator` class should be used instead. It works as follows:

```
using System;
using System.Security.Cryptography; // include definition!

class Program
{
    static void Main()
    {
        // choose secure (!) random integer
        // between 0 (inclusive) and 100 (exclusive)
        int secureRandom = RandomNumberGenerator.GetInt32(100);

        // display cryptographically secure int
        Console.WriteLine(secureRandom);

        // choose secure (!) random integer
        // between 50 (inclusive) and 500 (exclusive)
        int anotherSecureRandom = RandomNumberGenerator.GetInt32(50, 500);

        // display cryptographically secure int
        Console.WriteLine(anotherSecureRandom);
    }
}
```

You can learn more about secure random numbers by reading through:

- the technical description of `RandomNumberGenerator` class²
- discussion on how to choose which Random generator to use³

²<https://docs.microsoft.com/en-us/dotnet/api/system.security.cryptography.randomnumbergenerator>

³<https://stackoverflow.com/q/1257299>