

04.05 Virtual Lecture Notes: The Random Class

Java's Random class creates lists of pseudorandom numbers and then provides methods to take a number from the list

The following program generates random integer values from 0 (inclusive) to 100 (exclusive):

```
<1> import java.util.Random;
<2> public class RandomNumbers
<3> {
<4>     public static void main(String[] args)
<5>     {
<6>         int randNum = 0;
<7>         Random randNumList = new Random();
<8>
<9>         System.out.println(" Table of Random Numbers  ");
<10>         for(int row = 0; row < 5; row++)
<11>         {
<12>             for(int col = 0; col < 10; col++)
<13>             {
<14>                 randNum = randNumList.nextInt(100);    //random numbers from 0 to 99
<15>                 System.out.print(randNum + "\t ");
<16>             } //end column loop
<17>             System.out.println();
<18>         } //end row loop
<19>     } //end main
<20> } //end class
```

Run this program and observe the output. Study the following line-by-line explanation:

Lines

<1>	imports the Random class from the java.util library.
<2>	declares a class named RandomNumbers .
<3>	opening curly brace marking the beginning of the class (matches up with Line <20>).
<4>	declares the main() method where program execution begins.
<5>	opening curly brace to start the main() method (matches up with line <19>).
<6>	declares and initializes the randNum variable.

<7>	constructs a new object of the Random class called randNumList . This object holds the list of random numbers generated.
<8>	white space to improve program readability.
<9>	print a String literal as the heading of the table.
<10>	beginning of the for loop, which will count from 0 to 4. There will be 5 rows in the table.
<11>	opening curly brace of the for loop (matches up with line <18>).
<12>	beginning of the for loop, which will count from 0 to 9. There will be 10 columns per row in the table.
<13>	opening curly brace of the for loop (matches up with line <16>).
<14>	the nextInt() method of the Random class selects a number from 0 (inclusive) to 99 (exclusive) and assigns it to the randNum variable.
<15>	prints the value of the randNum variable.
<15>	closing curly brace marking the end of the for loop (matches up with line <13>).
<17>	print to go to a new line.
<18>	closing curly brace marking the end of the for loop (matches up with line <11>).
<19>	closing curly brace marking the end of the main() method (matches up with line <5>).
<20>	closing curly brace marking the end of the class (matches up with line <3>).

Hand tracing code in this manner is the best way to fully understand a program's purpose and design. It makes you think!

Background

The **Random** class can pick random decimals and integers with the **nextDouble()** and **nextInt()** methods, respectively. Carefully study the following Method Summary excerpts from the **Random** class API:

```
double      nextDouble()
            Returns the next pseudorandom, uniformly
            distributed double value between 0.0 and 1.0
            from this random number generator's sequence.
```

```
int         nextInt(int bound)
            Returns a pseudorandom, uniformly distributed
            int value between 0 (inclusive) and the specified
            value (exclusive), drawn from this random number
            generator's sequence.
```

In this course, you will only need these two **Random** class methods. Do the method names look familiar? So far you have used them to accept integer and decimal values input from the keyboard and a file. Java frequently re-uses method names that do similar things in different classes. These two methods perform a similar task in all three classes; they retrieve the

next available value.

These use of these two methods is straightforward.

- The `nextInt()` method takes a parameter that indicates the maximum value of the integers in the list, **minus one** since the upper end of the range is not inclusive. Consequently, `nextInt(53)` would return random numbers in the range of 0–52.
- The `nextDouble()` method returns decimal values from 0.0 inclusive to 1.0 exclusive. Like the `random()` method of the `Math` class, decimal numbers returned will never equal 1.0.

Scaling is required for higher ranges of decimal numbers. For example, if you want random decimals between 0.0 and 1,000.0, you would simply multiply by the scaling factor, as shown below:

```
double doubleValue = randNumList.nextDouble() * 1000;
```

The context of a specific program will determine whether you use the `nextDouble()` and `nextInt()` methods of the `Math` class or the `Random` class.

The real key to generating random numbers with the `Random` class is the following statement in the program above:

```
Random randNumList = new Random();
```

You use a similar statement anytime you need to create a `Scanner` object to accept input from the keyboard. Objects will be covered in depth in an upcoming module, but in the meantime, simply realize that this statement assigns a list of pseudorandom numbers to an object called `randNumList`. When the `nextInt()` or `nextDouble()` methods are invoked, the next available number is retrieved from the list.

Modifications

As always, you will learn more about these methods from experimenting. Make the following modifications to the `RandomNumbers` class.

- Modify the program to display 10 rows of random numbers instead of 5.
- Change the program to choose random integers between 0 and 4319.
- Modify the range of integers picked with your own values.
- How can the random integers be converted to negative numbers? (Hint, multiply by negative one.)
- How could you pick random integers between -50 and 50? (Hint: Generate random numbers in the range of 0 to 101 and then subtract 50.)
- Change the program to pick random numbers between 0.0 and 1.0.
- Modify the program to pick random numbers between 0.0 and 433.0.
- How could you pick random numbers between 100.0 and 200.0?
- What happens if you put a decimal number in the parentheses of the `nextDouble()` method?
- What happens if you delete `randNumList` on line <14>?

Continue to experiment with the `nextInt()` and `nextDouble()` methods until you are satisfied that you understand their basic features.

