Generating Random numbers with the Random class

- Random numbers come in handy for many operations in a program, such as rolling dice, dealing cards, timing the appearance of a nemesis in a game, or other simulations of seemingly random events.

- There’s one problem in using random numbers in programs, however: Computers are deterministic. In essence, this means that given a specific input to a specific set of instructions, a computer will always produce the same output. The challenge, then, is generating random numbers while using a deterministic system. Many talented computer scientists have worked on this problem, and some innovative and complex solutions have been proposed.

- The Random class, which is in the java.util package, uses a mathematical formula to generate a sequence of numbers. The constructor generates a seed value, which determines where in that sequence the set of random numbers will begin. As such, the Random class generates numbers that appear to be, but are not truly random. These numbers are called pseudorandom numbers, and they work just fine for our purposes.

- The default constructor creates a random number generator using a seed value. Once the random number generator is created, we can ask for a random number by calling the nextInt( ) method. Other methods, nextDouble, nextBoolean, nextByte, and nextLong return each of their respective data type values.

Graphical user interface, application

Description automatically generated

- To demonstrate how to use the random number generator, let’s take rolling a die as an example. To simulate the roll of a six-sided die, we need to simulate random occurrences of the numbers 1 through 6. If we call the nextInt( ) method with an argument of 6, it will return an integer between 0 and 5. To get randomly distributed numbers from 1 to 6, we can simply add 1 to the value returned by the nextInt method. Thus, if we have instantiated a Random object named random, we can generate random numbers from 1 to 6 by calling the nextInt method in this way:

int die = random.nextInt(6) + 1;

- In general, assuming n is smaller than m, if we want to generate random numbers from n to m, we should call the nextInt method with the number of possible random values (m – n + 1), and then add the first value of our sequence (n) to the returned value. Thus, this statement generates a random number between 10 (n) and 100 (m) inclusive:

int randomNumber = random.nextInt(100 – 10 + 1) + 10;

Text

Description automatically generated

Graphical user interface, text, application, chat or text message

Description automatically generated

- When we subtract the start value from the end value (and then add 1), we get a number that is from 1 to (end – start). But then, once all of that is done, outside of that parenthetical operation we add the starting value. This is so the first possible generation of the “random” number is start+1. We must add 1 due to the indices. This causes the number to be exactly in between the start and end values and the +1 within the parentheses will make sure our output will not include zero.