Random class

A **random number** is a number that is almost impossible to predict, like the result of throwing a dice. A random number generator can provide you with such a number when you need it, and you will probably need it quite often. Random generators are widely used in cryptography, machine learning, games, and more.

Pseudorandom numbers and seeding

e random numbers we are going to discuss aren't truly random because they can always be determined by an initial value, called **seed**.

Every time we get a new random number, we actually get the next number in a predefined sequence. These numbers are often called **pseudorandom**, and they are not completely unpredictable! We can calculate them all if we know the initial value and the algorithm of the sequence. That initial value is called a **seed**.

It is guaranteed that the same seed produces the same sequence if the same Java runtime version is used because the algorithm is the same.
These generators are quite important because of their speed in number generation and their reproducibility.

Creating a pseudorandom generator

Java provides the Random class to generate pseudorandom values of different types, such as int, long, double, and even boolean.

import java.util.Random;

Random() creates a new random generator and sets the **seed** of the generator to a value that is very likely to be distinct from any other invocation of this constructor
Random random = new Random();

Random(long seed) creates a new random generator with the specified initial value of its internal state:
Random random = new Random(100000);

If we don't specify a seed, the generator will give us a new sequence every time. But if we specify the seed, the sequence will be calculated based on it.

Regardless of what constructor we used, we have a generator called random that can produce random numbers.

The basic methods

- int nextInt() returns a pseudorandom value of the int type;
- int nextInt(int n) returns a pseudorandom value of int type in the range from 0 (inclusive) to n (exclusive);
- long nextLong() returns a pseudorandom value of long type;
- double nextDouble() returns a pseudorandom value of double type between 0.0 and 1.0;
- void nextBytes(byte[] bytes) generates random bytes and places them into a user-supplied byte array.

If we need to reproduce the same sequence of random numbers, we may specify a seed to the constructor:

```
Random random = new Random(100000);
System.out.println(random.nextInt(5)); // it may print 0, 1, 2, 3, 4
System.out.println(random.nextInt(5)); // it may print 0, 1, 2, 3, 4
```

in this case, while starting the program multiple times, we will always get the same numbers in the output.

An object of the Random class can generate Gaussian distributed pseudorandom double numbers by invoking the nextGaussian() method. This distribution may be required for some statistical analysis and machine learning applications, but it is not that common in general programming.

An example: printing pseudorandom numbers

We want to use it to generate numbers within a specific range:

```
int next = random.nextInt(upper - lower + 1) + lower;
```