Theory: Random

24 minutes 0 / 5 problems solved

Skip this topic Start practicing

3482 users solved this topic. Latest completion was about 5 hours ago.

Sometimes you need to get a random number or another random value. It is useful for testing purposes or may be applied as an initial value in different algorithms or even in a game. For this purpose, Java provides the `Random` class. It represents a generator of **pseudorandom** sequences. Actually, these sequences are not truly random, because they are always determined by an initial value, called **seed**. However, that is good enough for practical tasks. These generators are quite important because of their speed in number generation and their reproducibility.

§1. Creating a pseudorandom generator

The class `Random` can generate random values of different types, such as `int`, `long`, `double`, and even `boolean`. We will consider how to use this class for numbers.

First of all, we need to do the import:

```
import java.util.Random;
```

We have two constructors to create an object of this class:

- `Random()` creates a new pseudorandom 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 pseudorandom generator with the specified initial value of its internal state:

```
Random random = new Random(100000);
```

Now, we have a generator object called `random` that can produce random numbers.

§2. The basic methods

After we've created a generator, we can invoke one of the following methods of it:

- `int nextInt()` returns a pseudorandom value of the `int` type;
- `int nextInt(int n)` returns a pseudorandom value of `int` type in the range of 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.

All the listed methods produce uniformly distributed values.

Let's take a look at an example:

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

If we start this code multiple times, the result is different (or it may happen to be the same).

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

Current topic:

Random Stage 5

Topic depends on:

✓ Objects Stage 3

Table of contents:

```java
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
```

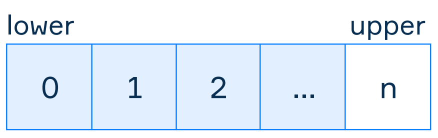
in this case, while starting the program multiple times, we will always get the same numbers in the output.

Note: 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 statistic analysis and machine learning applications, but it is not that common in general programming.

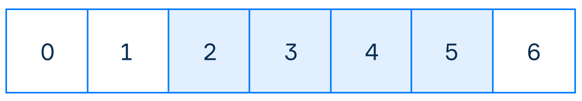
§3. An example: printing pseudorandom numbers

Let's suppose that we need a program that prints out the specified number of pseudorandom integers from the given range (inclusive both lower and upper borders). Unfortunately, the `Random` class does not provide a method to generate numbers in a range. Let's use it as an opportunity to practice and create it from scratch!

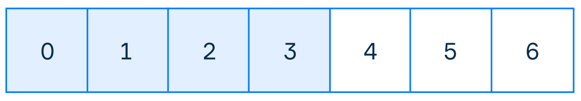
As you remember, the `nextInt(n)` method produces a pseudorandom integer from `0` (inclusive) to `n` (exclusive).



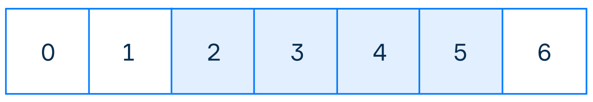
We want to use it to generate numbers from a specific range, for example, from 2 to 5, inclusive on both borders.



Let's take the length of the interval plus one: 5 – 2 + 1 = 4. It allows us to generate any number from 0 to 3 by using the `nextInt(4)` method.



Now imagine that shift the interval to the value of the lower border (2) as we need.



This way we can generate any numbers from 2 to 5 inclusive both borders.

The illustrated idea can be implemented by a simple code line:

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

Here is a complete program that prints 4 pseudorandom integers from the given range:

```java
import java.util.*;

public class RandomNumbersDemo {
    public static void main(String[] args) {
        Scanner scanner = new Scanner(System.in);

        int lower = scanner.nextInt();
        int upper = scanner.nextInt();
        Random random = new Random();

        int intervalLength = upper - lower + 1;

        System.out.println(random.nextInt(intervalLength) + lower);
        System.out.println(random.nextInt(intervalLength) + lower);
        System.out.println(random.nextInt(intervalLength) + lower);
        System.out.println(random.nextInt(intervalLength) + lower);
    }
}
```

For example, we have to generate exactly numbers in the range from 20 to 30 (inclusive):

```
20 30
```

An output example:

```
25
26
30
20
```

As you can see, dealing with the `Random` class is simple enough. Don't be afraid to introduce a bit of randomness into your programs :)

Report a typo

385 users liked this theory. **7** didn't like it. **What about you?**

Start practicing

Comments (5) Hints (0) Useful links (0) Show discussion