setSeed:

Ved å bruke setSeed er vi i stand til å rekonstruere tallsekvensen som random lager. Seed-et forteller random generatoren hvor den skal begynne

The main point of using the seed is to be able to reproduce a particular sequence of 'random' numbers. Generally speaking, if you don't need to be able to do that, you *wouldn't* set the seed. The seed itself carries no inherent meaning except it's a way of telling the random number generator 'where to start'. You might think of it a bit like the relationship between a PIN number and your bank account. The PIN is associated with a long string of numbers (your account number), but it's not inherently an interpretable quantity (there is *an* interpretation, but in setting it, you ignore that). – [Glen\_b](https://stats.stackexchange.com/users/805/glen-b" \o "188,975 reputation)**[♦](https://stats.stackexchange.com/users/805/glen-b" \o "188,975 reputation)** [Feb 12 '14 at 3:38](https://stats.stackexchange.com/questions/86285/random-number-set-seedn-in-r#comment168302_86285)

The seed number you choose is the starting point used in the generation of a sequence of random numbers, which is why (provided you use the same pseudo-random number generator) you'll obtain the same results given the same seed number.

Several others have mentioned reproducibility. Reproducibility is at the heart of debugging, you need to be able to reproduce the circumstances in which the bug occurred.

Another important use of reproducibility is that you can play some statistical games to reduce the variability of some estimates.

A specific seed will always give the same sequence of "random" numbers. So there are only 2^64 different sequences in Random! Besides setSeed one may also use a constructor with a seed. When not used the seed uses the computer's clock to select a really random seed.

So in the ordinary case use no specific seed, the constructor Random() and no setSeed. Especially refrain from setSeed(System.currentTimeMillis()).

For data dependent debugging, where you want to repeat the same pseudo-random data, use a specific seed.

Eksempel på bruk av metoden

ackage com.tutorialspoint;

import java.util.\*;

public class RandomDemo {

public static void main( String args[] ) {

// create random object

Random randomno = new Random();

// setting seed

randomno.setSeed(20);

// value after setting seed

System.out.println("Object after seed: " + randomno.nextInt());

**How to generate random numbers?**

import java.util.Random;

Random rand = new Random(); 🡪 creates a new Random object

The Random object provides you with a simple random number generator.

Picking random numbers from a certain range:

The nextInt() method also accept an int parameter. It denotes the upper limit range of numbers.

However, the upper limit number is not included as one of the numbers that can be picked.

F.ex: rand.nextInt(40) will pick random numbers from 0 to 39 inclusively.

To pick from a range that starts with 1, simply add 1 to the result of the nextInt() method.

F.ex: int pickedNumber = rand.nextInt(40) +1

If the range starts from a higher number than one you will need to:

* minus the starting number from the upper limit number and then add one.
* add the starting number to the result of the nextInt() method.

For example, to pick a number from 5 to 35 inclusively, the upper limit number will be 35-5+1=31 and 5 needs to be added to the result:

Random rand = new Random();

int pickedNumber = rand.nextInt(31) + 5;

**Just How Random Is the Random Class?**

I should point out that the Random class generates random numbers in a deterministic way. The algorithm that produces the randomness is based on a number called a seed. If the seed number is known then it's possible to figure out the numbers that are going to be produced from the algorithm. To prove this I'll use the numbers from the date that [Neil Armstrong](https://www.thoughtco.com/neil-armstrong-p2-3072206) first stepped on the Moon as my seed number (20th July 1969) :​

import java.util.Random;

public class RandomTest {;

  public static void main(String[] args) {

    Random rand = new Random(20071969);

    for (int j = 0; j

No matter who runs this code the sequence of "random" numbers produced will be:

3 0 3 0 7 9 8 2 2 5

By default the seed number that is used by:

Random rand = new Random();

is the current time in milliseconds since January 1, 1970. Normally this will produce sufficiently random numbers for most purposes. However, note that two random number generators created within the same millisecond will generate the same random numbers.

Also be careful when using the Random class for any application that must have a secure random number generator (e.g., a gambling program). It might be possible to guess the seed number based on the time the application is running. Generally, for applications where the random numbers are absolutely critical, it's best to find an alternative to the Random object. For most applications where there just needs to be a certain random element (e.g., dice for a board game) then it works fine.