15-07-09 - Hoe random is Random?

Random, het Engelse woord voor willekeurig. Maar hoe random is random nu eigenlijk? Als je honderd mensen vraagt een getal onder de 10 te kiezen, komt dan ieder getal even vaak voor? Ik moet je helaas het antwoord verschuldigd blijven, maar het is wel een leuk psychologisch spelletje. Sommige getallen zijn nu eenmaal populairder dan andere getallen, zoals 3.
Bij computers zouden voorkeuren geen rol moeten spelen. Computers kunnen namelijk (nog) niet zelf denken. Alles wat ze doen wordt gevraagd door hun gebruiker. Die gebruiker kan een mens zijn, maar ook software of andere hardware, zoals we vaak zien in de meet- en regeltechniek. Een computer kan hierdoor ook nooit een random getal selecteren. Een onwetende programmeur zal nu direct van z'n stoel springen, maar het is echt waar.

Random bestaat niet

Wanneer een computer om een random getal wordt gevraagd, levert dat een probleem op. Zoals gezegd, kan een computer niet uit zichzelf iets doen en dus geen willekeurig nummer selecteren. Om toch random nummers te kunnen gebruiken, hebben bijna alle programmeertalen een trucje ingebouwd. In de random-functie, worden getallen gegenereerd op basis van hun voorganger.

Eigen random functie

Een vrij eenvoudig voorbeeld van een random functie is:

```
int last = 1;
function random ( int max ) {
   last = last * 7 % max;
   return last;
}
```

In dit voorbeeld wordt het laatst gegenereerde getal steeds met 7 vermenigvuldigd. Om te zorgen dat het getal niet hoger is dan de aangegeven max, wordt er een modulus op los gelaten. Met modulus haal je een getal zo vaak mogelijk van een ander getal af en geef je vervolgens de restwaarde. 100 % 33 = 1, want 100 ? 33 ? 33 ? 33 = 1.
Bij dit voorbeeld lopen we tegen twee problemen aan.
1. We beginnen met 1, dus de gegenereerde random lijst, is niet random, maar zal iedere keer na het opstarten een zelfde reeks geven. Visual Basic had hier altijd last

van. De makkelijkste oplossing was om niet te beginnen met 1, maar met een timestamp. Die is namelijk altijd anders.
2. Wanneer we een max van 100000 nemen, zullen we altijd een stijgende lijn zien in de reeks nummers. We beginnen bijvoorbeeld met:
1 7 49 343 2401 16807
Het getal daarna wordt 117649, maar dat zou de max overschreiden, dus wordt het:
17649 23543 64801 53607 75249 26743 87201 10407
er is duidelijk een patroon te herkennen en voor een Hoger-Lager-spelletje is zoiets natuurlijk funest. Om dit probleem op te lossen, zouden we een veel groter priemgetal moeten gebruiken dan 7.

Random in Java

Om random getallen te genereren in Java, kun je gebruik maken van java.util.Random. Deze klasse, kun je random getallen laten genereren met nextInt(max). Om te bewijzen dat de gekozen nummers echt random zijn en bovendien gelijk verdeeld, heb ik een paar testjes gedaan.

Test 1: 50-50

```
import java.util.Random;Array Arraypublic class HogerLager
{Array public static void main ( String[] args )
{Array  System.out.println (
"+-----------------------------------------+"
);Array  System.out.println ( "+ Zijn random getallen gelijk
verdeeld?      +" );Array  System.out.println (
"+-----------------------------------------+"
);Array Array  int above = 0;Array  int beneath =
0;Array  double n = 100000.0;Array  Random r = new Random (
);Array  for ( int i = 0; i < n; i++) {Array   int x =
r.nextInt ( 10 ) + 1;Array   if ( x > 5
)Array    above++;Array   elseArray    beneath++;Array  }Array
  System.out.println ( "Van de " + (int)n + " random getallen
(1 t/m 10), was" );Array  System.out.println ( "- " +
((above/n)*100) + "% 6 of hoger" );Array  System.out.println (
"- " + ((beneath/n)*100) + "% 5 of lager" );Array }Array}
```

Ik heb de percentages bewust niet afgerond, omdat ik geïnteresseerd ben in het meest precieze data. Wanneer ik het programma draai, krijg ik de volgende uitkomst:

```
+-----------------------------------------+
+ Zijn random getallen gelijk verdeeld?     +
+-----------------------------------------+
Van de 100000 random getallen (1 t/m 10), was
- 49.695% 6 of hoger
- 50.305% 5 of lager
```

Er is dus een lichte voorkeur voor de onderste helft van de mogelijke getallen. Hoe hoger de n (aantal tests), hoe dichter de percentages tegen de 50 aan zullen liggen. Bij een n van 10000000, zien we pas verschil bij het derde getal achter de komma. Daarom kunnen we zeggen dat de verdeling tussen hoge en lage getallen gelijk is wanneer we veel testen.

Test 2: Gemiddelde

Om een goed beeld te krijgen hoe precies de verdeling is, heb ik een test onwikkeld, dat het gemiddelde van de gekozen nummers kan bepalen.

```
import java.util.Random;Arrayimport
java.math.BigInteger;Array Arraypublic class HogerLager
{Array public static void main ( String[] args )
{Array  System.out.println (
"+-----------------------------------------+"
);Array  System.out.println ( "+ Zijn random getallen gelijk
verdeeld?     +" );Array  System.out.println (
"+-----------------------------------------+"
);Array  System.out.println ( );Array  System.out.println (
"+--------------+" );Array Array  double z = 0.0;Array  int n
= 100;Array  int n2 = 10000000;Array Array  for ( int j = 0; j
< n; j++ ) {Array   System.out.print ( (j+1) + " ");Array   if
( j < 9 )Array    System.out.print ( " " );Array   if ( j < 99
)Array    System.out.print ( " " );Array   int total =
0;Array   Random r = new Random ( );Array   for ( int i = 0; i
< n2; i++) {Array    int x = r.nextInt ( 10 ) +
1;Array    total += x;Array    if ( i % ( n2 / 10 ) == 0
)Array     System.out.print (
"*");Array   }Array   System.out.println ( " |" );Array   z +=
total / (double)n2;Array  }Array  System.out.println (
"+--------------+" );Array  System.out.println ( "Gemiddelde:
" + (z/(double)n) );Array }Array}
```

En het resultaat:

```
Gemiddelde: 5.499953607999998
</code
Wederom een lichte voorkeur voor de lagere getallen. Een
lichte teleurstelling, omdat ik verwacht had na 1.000.000.000
getallen wel op precies 5.5 zou zitten. Maar alsnog is het
natuurlijk een hele goede score.</p>Array
```

Test 3: Getalverdeling ArrayEn dan nu de laatste test. Ik ga de computer vragen om een getal onder de 10 en kijk of de verdeling over de getallen van 1 t/m 10 gelijk is.ArrayArray

```
import java.util.Random;Array Arraypublic
class GetalVerdeling {Array public static
void main ( String[] args ) {Array  int x[] =
new int[10];Array  Random r = new Random (
);Array  int n = 100000000;Array Array  for (
int i = 0; i < n; i++ ) {Array   int y =
r.nextInt ( 10
);Array   x[y]++;Array  }Array Array  for (
int i = 0; i < 10; i++)
{Array   System.out.println ( (i+1) + ": " +
( x[i] / (double) n ) * 100 + "%"
);Array  }Array }Array}
```

En natuurlijk het resultaat:

```
1: 9.996597999999999%
2: 10.00047%
3: 9.998962%
4: 10.000429%
5: 10.000429%
6: 10.000235%
7: 10.000811%
8: 10.000245000000001%
9: 10.003157999999999%
10: 9.998663%
```

Na 100 miljoen getallen is de verdeling nog steeds niet gelijk. Opvallend is, dat 1 en 10 de laagste scores hebben gehaald. Dit berust echter op toeval, want de vier tests

die ik daarna heb uitgevoerd, vertoonden dit niet.

Conclusie

Computers kunnen niet denken, maar zijn wel waanzinnig goed in het genereren van random getallen. Hoe meer tests je uitvoert, hoe preciezer het resultaat wordt. Uit de laatste test kunnen we concluderen dat alle getallen even vaak voorkomen. Wanneer je een RaadHetGetal-spelletje speelt en alleen 8 is nog niet geweest, dan is de kans groot dat dit de volgende zal zijn.
De random-functie in Java is goed te gebruiken voor een party-mode in een mp3-speler. Deze zal at random mp3'tjes selecteren en op de langere termijn zal ieder nummer gespeeld worden, zonder dat er een patroon te vinden is. Wel zal de MP3-speler altijd een voorkeur hebben voor een bepaald nummer. Uit alle tests bleek, dat pas na een miljoen getallen, het verschil te verwaarlozen is. Doe je Test 3 met n=10, dan zul je merken dat sommige nummers helemaal niet zijn voorgekomen en andere wel 3 keer. Toch is het algoritme erop gebaseerd om het gelijk op te laten gaan. Zeker voor bijvoorbeeld kaartenschudmachines kan dit een risico vormen. Maar bij handmatig schudden loop je weer tegen andere problemen aan, zoals clusters van kaarten.
Echt random bestaat dus niet bij computers, maar het is al een stuk beter dan wat mensen presteren.