GENERAR NÚMEROS ALEATORIOS EN JAVA

Para generar números aleatorios en Java tenemos dos opciones. Por un lado podemos usar *Math.random()*, por otro la clase *java.util.Random*. La primera es de uso más sencillo y rápido. La segunda nos da más opciones.

1) **Math.random( )**

La llamada a *Math.random()* devuelve un número aleatorio entre 0.0 y 1.0, excluido este último valor, es decir, puede devolver 0.346442, 0.2344234, 0.98345,....

En muchas de nuestras aplicaciones no nos servirá este rango de valores. Por ejemplo, si queremos simular **una tirada de dado**, queremos números entre 1 y 6 sin decimales. Debemos hacer algunas unas cuentas para obtener lo deseado.

En primer lugar, miramos cuántos valores queremos. En nuestro caso del dado son 6 valores, del 1 al 6 ambos incluidos. Debemos entonces multiplicar *Math.random()* por 6 (que es el resultado de hacer la siguiente operación (máximo-mínimo)+1, es decir, ((6-1)+1). La sintaxis sería:

```
Math.random()*6    // Esto da valores de 0.0 a 6.0, excluido el 6.0, pero será con decimales y tendríamos que usar printf para mostrar el entero.
```

Como el primer valor que queremos es 1 y no 0, le sumamos 1 al resultado, quedando:

```
Math.random()*6 + 1   // Esto da valores entre 1.0 y 7.0 excluido el 7.0, es decir, del 1 al 6 que es lo que queremos, pero sigue dando decimales.
```

Finalmente, para conseguir un entero en lugar de un real, quitamos los decimales usando la clase y el método *Math.floor()* (floor devuelve el entero anterior de lo de dentro del paréntesis, por ejemplo, si ponemos 5.7 devuelve 5, es decir, devuelve el “suelo” (para abajo) entero más próximo). También lo guardamos en una variable de nombre valorDado:

```
int valorDado = (int)Math.floor(Math.random()*6+1); // Ojo, primero multiplica por 6 el aleatorio y luego suma 1.
```

En general, para conseguir un número entero entre M y N con M menor que N, debemos usar esta fórmula:

```
int valorEntero = (int)Math.floor(Math.random()*(N-M+1)+M);

// Valor entre M y N, ambos incluidos.
```

Si no queremos un valor entero sino *double*, la fórmula es sin el +1 (eso sí, recuerda que el valor N queda excluido y no saldrá nunca)

double valorAleatorio = Math.random()*(N-M)+M;

2) Clase java.util.Random

La clase *java.util.Random* debemos instanciarla, a diferencia del método *Math.random( )* que es llamada sin crear ningún objeto usando el nombre de la clase **Math**. A cambio, tendremos bastantes más posibilidades.

Podemos usar un constructor sin parámetros o bien pasarle una semilla. Si instanciamos varias veces la clase con la misma semilla, tendremos siempre la misma secuencia de números aleatorios.

NOTA: Los números aleatorios en este caso se crean realizando muchas operaciones matemáticas sobre un primer número llamado "semilla". En realidad, se llaman números pseudoaleatorios. Si partimos del mismo primer número llegaremos al mismo número final (el supuestamente aleatorio) por tanto hay que cambiar de semilla si queremos números aleatorios diferentes.

Random r1 = new Random();

Random r2 = new Random(4234);

Random r3 = new Random(4234); // r2 y r3 darán el mismo reultado.

Lo más fácil es usar el constructor sin parámetros, que normalmente dará secuencias distintas en cada instancia. De todas formas, una manera de obtener una semilla que sea distinta cada vez que ejecutemos nuestro programa puede ser obtener el tiempo actual en nanosegundos con *System.nanoTime( )*, que dará números distintos salvo que hagamos la instancia justo en el mismo instante de tiempo (con milisegundos los actuales procesadores darían en un bucle el mismo número):

- **Usando System.nanoTime ( ):**

```
import java.util.Random;

public class Ppal {

	public static void main(String [ ] args) {

		Random num= new Random (System.nanoTime());

		for (int i=0; i<5; i++) {

			int numeroAleatorio= num.nextInt (10);

			System.out.println("Número aleatorio entre 0 y 9: "+numeroAleatorio);

		}

	}

}
```

De esta manera, la semilla cambiará cada vez que ejecutemos el programa.

Con esta clase, una vez instanciada, nuestro problema del dado sería bastante más sencillo, usando el método *nextInt (int n)*, que devuelve un valor entre 0 y n, excluido n:

```
Random r = new Random(); //Creamos un objeto tipo Random

int valorDado = r.nextInt(6)+1;  // Entre 0 y 5, más 1 para obtener el 1 y el 6.
```

* Por último, para generar valores en un rango dado, por ejemplo, entre 10 y 100, basta con sumarle el menor al resultado de generar valores de la resta entre los dos límites, siempre mayor menos menor.

Ejemplo: Entre 10 y 100

```
Random r = new Random();//Creamos un objeto tipo Random

int valorDado = r.nextInt(100-10)+10;
```

Esto generará números entre 0 y 90 pero al resultado le suma 10 entonces, si se obtiene el número 0 le suma 10 y resulta 10 como mínimo y si saca el mayor que sería 90 le suma 10 y queda en 100 en general es:

```
int valorDado = r.nextInt(Max-min)+min;
```

En general, para generar enteros al azar entre dos límites DESDE, HASTA, ambos incluidos, la fórmula es:

rnd.nextInt (HASTA-DESDE+1)+DESDE

Por ejemplo, para generar 5 números aleatorios entre 10 y 20:

```
for (int i = 0; i<5; i++){
    System.out.println (rnd.nextInt (20-10+1)+10);
}
```

También tenemos funciones que nos dan un valor aleatorio siguiendo una curva de Gauss o que nos rellenan un array de bytes de forma aleatoria. Y por supuesto, el *nextDouble ( )* que devuelve un valor aleatorio entre 0.0 y 1.0, excluido este último.

Probad vosotros a generar números entre cualquier intervalo pensando en determinados juegos o aplicaciones donde os harían falta, por ejemplo, dados, cartas, loterías, etc.