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.

- Math.random()
- Clase java.util.Random

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 unos ajustes 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 para obtener un número decimal entre 0 y 6, pero sin llegar a dar el 6 nunca:

```
Math.random()*6   // Esto da valores de 0.0 a 5,999, no llega al 6.0
```

Y después sumar 1 para que genere el valor uno como valor más pequeño.

```
Math.random()*6 + 1

// Esto da valores entre 1.0 y 7.0 excluido el 7.0
```

Por lo tanto, la regla nos quedaría que si queremos generar valores entre 1 y un límite dado:

```
Math.random()*limite+1;
```

Donde límite será el número más alto del rango. Los números genéricos que generaremos irán entre el 1 y el límite.

Finalmente, para conseguir un entero, quitamos los decimales usando la clase *Math.floor()* (Devuelve el máximo entero menor o igual a un número)

```
int valorDado = Math.floor(Math.random()*6+1);
```

O también haciendo un casting a un entero:

```
int numeroAleatorio = (int)(Math.random()*6+1);
```

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

```
 int valorEntero = (int)(Math.random()*(M-N+1)+N);
// Valor entre N y M, ambos incluidos.
```

Clase java.util.Random

La clase *java.util.Random* debemos instanciarla, a diferencia del método anterior, *Math.random()*. 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.

```
Random r1 = new Random();
Random r2 = new Random(4234);
Random r3 = new Random(4234); // r2 y r3 darán la misma secuencia.
```

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 milisegundos con *System.currentTimeMillis()*, que dará números distintos salvo que hagamos la instancia justo en el mismo instante de tiempo. **Nota**: la implementación interna de Java depende de la versión de Java.

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

```
Random r = new Random();
int valorDado = r.nextInt(6)+1;  // Entre 0 y 5, más 1. Es decir, 1-6
```

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 decimal entre 0.0 y 1.0, excluido este último.

La clase *Random* proporciona un generador de números aleatorios que es más flexible que el método estático *random* de la clase *Math*.

Para crear una secuencia de números aleatorios tenemos que seguir los siguientes pasos:

1. Proporcionar a nuestro programa información acerca de la clase *Random.* Al principio del programa escribiremos la siguiente sentencia.

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

2. Crear un objeto de la clase *Random*
3. Llamar a una de las funciones miembro que generan un número aleatorio
4. Usar el número aleatorio.

Constructores

La clase dispone de dos constructores, el primero crea un generador de números aleatorios cuya semilla es inicializada en base al instante de tiempo actual.

```
Random rnd = new Random();
```

El segundo, inicializa la semilla con un número del tipo **long**.

```
Random rnd = new Random(3816L);
```

El sufijo L no es necesario, ya que, aunque 3816 es un número **int** por defecto, es promocionado automáticamente a **long**.

Aunque no podemos predecir qué números se generarán con una semilla particular, podemos, sin embargo, duplicar una serie de números aleatorios usando la misma semilla. Es decir, cada vez que creamos un objeto de la clase *Random* con la misma semilla obtendremos la misma secuencia de números aleatorios. Esto no es útil en el caso de loterías, pero puede ser útil en el caso de juegos, exámenes basados en una secuencia de preguntas aleatorias, las mismas para cada uno de los estudiantes, simulaciones que se repitan de la misma forma una y otra vez, etc.

Funciones miembro

Podemos cambiar la semilla de los números aleatorios en cualquier momento, llamando a la función miembro *setSeed*.

```
rnd.setSeed(3816);
```

Función miembro	Descripción	Rango
rnd.nextInt()	Número aleatorio entero de tipo int	2^{-32} y 2^{32}
rnd.nextLong()	Número aleatorio entero de tipo long	2^{-64} y 2^{64}
rnd.nextFloat()	Número aleatorio real de tipo float	[0,1[
rnd.nextDouble()	Número aleatorio real de tipo double	[0,1[

Podemos generar números aleatorios de cuatro formas diferentes:

```
rnd.nextInt();
```

Genera un número aleatorio entero de tipo **int**

```
rnd.nextLong();
```

Genera un número aleatorio entero de tipo **long**

```
rnd.nextFloat();
```

Genera un número aleatorio de tipo **float** entre 0.0 y 1.0, aunque siempre menor que 1.0

```
rnd.nextDouble();
```

Genera un número aleatorio de tipo **double** entre 0.0 y 1.0, aunque siempre menor que 1.0

Casi siempre usaremos esta última versión. Por ejemplo, para generar una secuencia de 10 números aleatorios entre 0.0 y 1.0 escribimos

```
for (int i = 0; i < 10; i++) {
    System.out.println(rnd.nextDouble());
}
```

Para crear una secuencia de 10 números aleatorios enteros comprendidos entre 0 y 9 ambos incluidos escribimos:

```
int x;
for (int i = 0; i < 10; i++) {
        x = (int)(rnd.nextDouble() * 10.0);
        System.out.println(x);
}
```

(int) transforma un número decimal **double** en entero **int** eliminando la parte decimal.

Secuencias de números aleatorios

En la siguiente porción de código, se imprimen dos secuencias de cinco números aleatorios uniformemente distribuidos entre [0, 1), separando los números de cada una de las secuencias por un carácter tabulador.

```
System.out.println("Primera secuencia");
for (int i = 0; i < 5; i++) {
    System.out.print("\t"+rnd.nextDouble());
}
System.out.println("");

System.out.println("Segunda secuencia");
for (int i = 0; i < 5; i++) {
    System.out.print("\t"+rnd.nextDouble());
}
System.out.println("");
```

Comprobaremos que los números que aparecen en las dos secuencias son distintos.

En la siguiente porción de código, se imprimen dos secuencias iguales de números aleatorios uniformemente distribuidos entre [0, 1). Se establece la semilla de los números aleatorios con la función miembro *setSeed*.

```
rnd.setSeed(3816);
System.out.println("Primera secuencia");
for (int i = 0; i < 5; i++) {
    System.out.print("\t"+rnd.nextDouble());
}
System.out.println("");

rnd.setSeed(3816);
System.out.println("Segunda secuencia");
for (int i = 0; i < 5; i++) {
    System.out.print("\t"+rnd.nextDouble());
}
System.out.println("");
```

Otra forma:

```
ThreadLocalRandom.current().nextInt(min, max);
```