Anexo I.- Números pseudo-aleatorios con Random

Un **número aleatorio** es aquel escogido al azar, es decir, donde todos los números tienen la misma probabilidad de ser seleccionados y no hay forma de poder determinar el resultado antes de que se produzca.

Los ordenadores están construidos para ser máquinas deterministas, es a decir, dado un conjunto de entradas, las salidas serán siempre las mismas. Esto supone un problema si queremos generar números aleatorios con un ordenador.

Entonces, ¿cómo podemos generar números aleatorios con un ordenador? la respuesta es simple, no podemos.

Ahora bien, en muchas ocasiones vamos a necesitar generar números aleatorios en nuestros programas (videojuegos, simulaciones, criptografía, etc..) y para ello tenemos dos aproximaciones:

- Usar números **pseudo-aleatorios**, que son números generados mediante un algoritmo donde es muy complicado poder predecir el resultado, puesto que el conjunto de posibles combinaciones es extremadamente grande.
- Tomar un número aleatorio de una **fuente externa**, por ejemplo el ruido atmosférico (random.org).

Para obtener los mejores resultados se tendría que utilizar un fuente externa pero, evidentemente, eso supone un impedimento. En la mayoría de casos será suficiente utilizar números **pseudo-aleatorios**.

La clase Random

La clase **Random** pertenece al paquete java.util y permite generar números pseudo-aleatorios en Java. Hay que tener en cuenta que los ordenadores son máquinas deterministas, y por tanto si proporcionamos los mismos datos de entrada y aplicamos los mismos algoritmos, siempre obtendremos los mismos resultados. Para generar números aleatorios, los ordenadores utilizan una función matemática que proporciona una secuencia de números. Si el dato de entrada a la función, conocido habitualmente como semilla, siempre es el mismo, siempre obtendremos la misma secuencia de números.
Para generar números pseudo-aleatorios con la clase Random seguiremos los siguientes pasos:

1. En primer lugar debemos hacer el import de la clase Random:
   ```
   import java.util.Random;
   ```
2. Crear un objeto de la clase *Random*.
   ```
   Random rnd = new Random();
   ```
   La clase Random por defecto utiliza como semilla para la generación de números pseudo-aleatorios el timestamp actual del sistema en nanosegundos (System.nanoTime()).
   Si queremos utilizar una semilla concreta, podemos pasársela como parámetro en el constructor de

la clase Random. Por ejemplo:

```
Random rnd = new Random(5000);
```

En este caso se utilizaría el número 5000 como semilla para generar la secuencia de números pseudo-aleatorios.

3. Acceder a los métodos de la clase *Random* que necesitamos. Los más interesantes son:
   a) nextInt(): devuelve un número entero positivo o negativo dentro del rango de enteros.
   b) nextInt(n): devuelve un número entero >=0 y menor que n.
   c) nextLong(): devuelve un número entero positivo o negativo dentro del rango de los enteros largos (long).
   d) nextBoolean(): devuelve valor booleano (true o false) aleatorio
   e) nextDouble(): devuelve un número real (con decimales) positivo de tipo double mayor o igual que 0.0 y menor que 1.0.
   f) nextFloat(): devuelve un número real (con decimales) positivo de tipo float mayor o igual que 0.0f y menor que 1.0f

Por ejemplo, para generar 5 números enteros aleatorios en el rango de los enteros y mostrarlos por pantalla:

```
Random rnd = new Random();
for(int i = 1; i <= 5 ;i++) {
    int aleatorio = rnd.nextInt();
    System.out.println(aleatorio);
}
```

Un posible resultado sería:

```
-394395199
1133763686
-424454120
1147379979
-2049113297
```

Para generar 5 números enteros entre 0 y 6:

```
Random rnd = new Random();
for(int i = 1; i <= 5; i++) {
    int aleatorio = rnd.nextInt(7);
    System.out.println(aleatorio);
}
```

Hay que tener en cuenta que el rango superior no está incluído, por eso se le pasa 7 para que vaya de 0 a 6.

A partir de lo anterior, podemos crear una fórmula que nos permita generar números enteros pseudo-aleatorios entre entre cualesquiera dos números (min y max), ambos incluidos:

```
Random rnd = new Random();
int aleatorio = rnd.nextInt(max - min + 1) + min;
```

Figura 1: Generación de números enteros pseudo-aleatorios en un rango determinado

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

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

En el caso que queramos generar números aleatorios de tipos double o float la fórmula sería ligeramete diferente.

```
Random rnd = new Random();
double aleatorio = rnd.nextDouble() * (max - min) + min;
```

Figura 2: Generación de números reales pseudo-aleatorios en un rango determinado

Por ejemplo, para generar 5 números reales pseudo-aleatorios entre 50 y 100:

```
Random rnd = new Random();
for(int i = 1; i <= 5; i++) {
    double aleatorio = rnd.nextDouble() * (100 - 50) + 50;
    System.out.println(aleatorio);
}
```

Desde la versión 17 de Java existe la posibilidad de utilizar los métodos nextInt, nextLong, nextFloat y nextDouble pasándole un rango. Por ejemplo:

```
Random rnd = new Random();
int aleatorioInt = rnd.nextInt(1, 10);
long aleatorioLong = rnd.nextLong(1, 10);
float aleatorioFloat = rnd.nextFloat(1.0f, 10.0f);
double aleatorioDouble = rnd.nextDouble(1.0, 10.0);
```

Dado que la versión 17 de Java es una versión relativamente reciente, se aconseja no utilizar estos métodos, ya que su uso obligaría a tener compatibilidad con dicha versión, y muchos frameworks e incluso Android, tardan en actualizar sus API por lo que tardan en dar soporte a características de Java recientemete añadidas.