APRENDERAPROGRAMAR.COM

GENERACIÓN DE NÚMEROS ALEATORIOS EN JAVA. CLASE RANDOM. EJEMPLOS Y EJERCICIOS RESUELTOS. (CU00908C)

Sección: Cursos
Categoría: Lenguaje de programación Java nivel avanzado I

Fecha revisión: 2039

Resumen: Entrega nº8 del curso "Lenguaje de programación Java Nivel Avanzado I".

Autor: Walter Sagástegui y José Luis Cuenca

GENERACIÓN DE NÚMEROS ALEATORIOS

La clase *Random* proporciona un generador de números aleatorios. Es más flexible que la función *random* de la clase *Math*. Vamos a ver un repaso de uso de la clase Random con ejemplos y ejercicios resueltos. La generación de números aleatorios adquiere gran relevancia para un programador, pudiendo tener distintas aplicaciones como:

- Construcción preliminar de programas, en los que a falta de datos definitivos, introducimos datos aleatorios.
- Simulación de procesos aleatorios (número resultante de tirar un dado, elección de un color por parte de una persona, número premiado en un sorteo de lotería, cantidad de personas que entran a un supermercado en una hora...)
- Verificación de programas, y en particular lo que en verificación de algoritmos se conoce como verificación aleatoria (probar el programa con distintos supuestos aleatorios).
- Otras aplicaciones.

Conviene recordar que "aleatorio" no puede confundirse con "cualquier cosa", "descontrol", "incierto", "impredecible", etc. Usaremos el vocablo “aleatorio”, más en el sentido de “no predeterminado” que el de “no predecible”, ya que en general, vamos a definir qué tipo de resultado queremos obtener y en qué rango de valores debe estar. Vamos a imaginar que Java genera números aleatorios, como si fuera un robot lanzador de dardos muy preciso (*robot rnd*). De este modo, cuando se le dice que comience a tirar dardos en distintas posiciones, repite siempre los lugares. Por ejemplo, si la diana está marcada con números, cada vez que le decimos que tire, genera la misma secuencia: 7, 5, 6, 3, etc. ¿Cómo conseguir convertir este proceso predefinido en aleatorio? Simplemente, poniendo a girar la diana (mayordomo randomize), en este caso, a una velocidad que depende del segundo del día en que nos encontremos. Así pues, el proceso lo dividimos al decirle al mayordomo que ponga a girar la diana y al indicarle al robot que dispare. Bueno, un poco simple, pero ¿para qué complicarnos? Veamos los pasos sintácticos a emplear para crear una secuencia de números aleatorios:

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:**

La clase Random dispone de dos constructores, para crear un objeto. El primer constructor es:

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

Este constructor crea un generador de números aleatorios cuya semilla es inicializada automáticamente, en base al tiempo actual. Esto conlleva que en cada ejecución la semilla cambie, es decir, que la secuencia de números aleatorios que se genera en cada ejecución siempre será diferente.

El segundo constructor es:

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

Este constructor nos permite inicializar la semilla manualmente con un número entero cualquiera. Si este número es el mismo en cada ejecución, la secuencia de números aleatorios que se genera en cada ejecución será igual.

3. **Llamar a una de las funciones miembro que generan un número aleatorio:**

Hay cuatro *funciones miembro* diferentes que generan números aleatorios:

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[

Lo más habitual, para generar un número aleatorio, es usar la *función miembro* **rnd.nextDouble()**.El valor devuelto es de tipo double. Un aspecto importante a tener en cuenta, es que el valor devuelto se encuentra en el rango mayor o igual a cero y menor a 1. Es decir, el número devuelto puede ser cero pero no puede ser uno.

En el siguiente ejemplo mostraremos todo lo expresado hasta este momento:

```
/* Ejemplo uso clase Random() – aprenderaprogramar.com */

import java.util.Random;

public class Programa {

    public static void main(String arg[ ]) {

        Random rnd = new Random();
        System.out.println("Número aleatorio real entre [0,1[ : "+rnd.nextDouble());

    }

}
```

En el caso de necesitar números aleatorios enteros en un rango determinado, podemos trasladarnos a un intervalo distinto, simplemente multiplicando, aplicando la siguiente fórmula general:

```
(int) (rnd.nextDouble() * cantidad_números_rango + término_inicial_rango)
```

donde (int) al inicio, transforma un número decimal *double* en entero *int*, eliminando la parte decimal.

Por ejemplo, si deseamos números enteros comprendidos entre el rango [0,99], incluidos los extremos, la fórmula quedaría de la siguiente manera:

(int)(rnd.nextDouble() * 100 + 0); → (int)(rnd.nextDouble() * 100);

100 es la cantidad de números enteros en el rango [0,99] y 0 es el término inicial del rango.

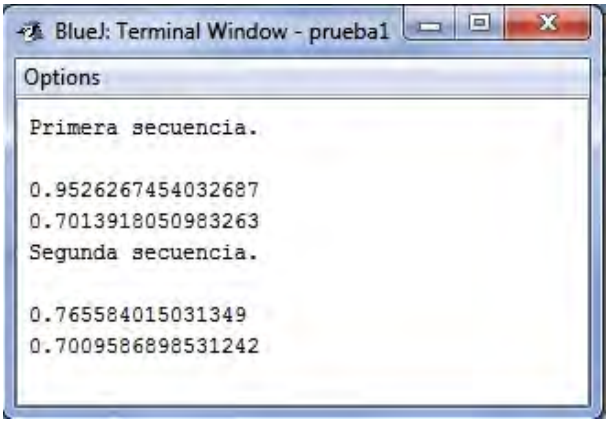
En el caso de querer números aleatorios enteros comprendidos entre [1,6], que son los lados de un dado, la fórmula quedaría así.

(int)(rnd.nextDouble() * 6 + 1);

donde 6 es la cantidad de números enteros en el rango [1,6] y 1 es el término inicial del rango.

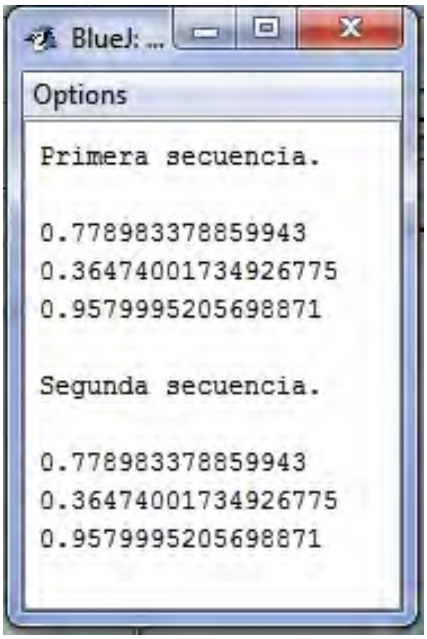
A continuación, veremos dos programas completos , como ejemplo para explicar el cambio de semilla en la generación de números aleatorios. Compílalos y comprueba sus resultados:

```
/* Ejemplo uso clase Random() – aprenderaprogramar.com */
import java.util.Random;

public class Programa {
    public static void main(String arg[ ]) {
          Random rnd = new Random();

          System.out.println("Primera secuencia."+"\n");
          System.out.println(rnd.nextDouble());
          System.out.println(rnd.nextDouble());

          System.out.println("Segunda secuencia."+"\n");
          System.out.println(+rnd.nextDouble());
          System.out.println(rnd.nextDouble());
    }
}
```



En este primer ejemplo, mostramos dos secuencias de tres números aleatorios, los cuales están en el rango [0,1[, incluido el cero pero excluido el uno. Vemos que los números aleatorios en las dos secuencias, son distintos, ya que en realidad, es una sola secuencia, que tiene como punto de partida la misma semilla. A continuación, lo compararemos con el siguiente ejemplo, donde utilizaremos la función ***"setSeed(semilla_nueva)"***, para el cambio de semilla, función resaltada con un comentario flecha:

```
/* Ejemplo uso clase Random() – aprenderaprogramar.com */

import java.util.Random;

public class Programa {

      public static void main(String arg[ ]) {
           Random rnd = new Random();

           rnd.setSeed(3816);  //←
           System.out.println("Primera secuencia."+"\n");
           System.out.println(rnd.nextDouble());
           System.out.println(rnd.nextDouble());
           System.out.println(rnd.nextDouble());

           System.out.println("");

           rnd.setSeed(3816);  //←
           System.out.println("Segunda secuencia."+"\n");
           System.out.println(rnd.nextDouble());
           System.out.println(rnd.nextDouble());
           System.out.println(rnd.nextDouble());
      }
}
```



Observamos que las dos secuencias están formadas por los mismos números aleatorios, ya que a ambas secuencias, les precede la función de cambio de semilla ***"rnd.setSeed(3816)"***. En dicha instrucción, las dos funciones aparecen inicializadas a propósito con la misma semilla "3816", pudiendo haberse elegido cualquier otro número, pero con la peculiaridad de que la semilla tiene que ser igual en ambas funciones para producir dos secuencias con los mismos números aleatorios.

EJERCICIO

Crea un programa que cumpla lo indicado a continuación. El programa permitirá jugar a "adivinar un número entero" tantas veces como el usuario desee. En cada ocasión el programa pedirá al usuario el número inicial y el número final de la serie de números entre los que se encontrará el número a adivinar. También preguntará el número de intentos permitidos. En cada intento el número a adivinar será diferente y estará dentro del rango definido por el usuario.

Trás los intentos el programa nos dirá cuantas veces hemos acertado y cuántas veces hemos fallado, los números que eran solución y nos preguntará si queremos volver a jugar.

Ejemplo de ejecución:

```
Introduce el numero inicial del rango: 40
Introduce el numero final del rango: 60
Introduce el numero de intentos: 2
¿Qué numero estoy pensando? : 44
No has acertado
¿Qué numero estoy pensando? : 42
No has acertado
Has acertado 0 veces y has fallado 2 veces. Eran solución: 49, 41.
Quieres probar otra vez ? (S/N): S
Introduce el numero inicial del rango: 4
Introduce el numero final del rango: 6
Introduce el numero de intentos: 2
¿Qué numero estoy pensando? : 4
No has acertado
¿Qué numero estoy pensando? : 5
Has acertado!!
Has acertado 1 veces y has fallado 1 veces. Eran solución: 6, 5
Quieres probar otra vez ? (S/N): N
Adiós
```

Para comprobar si tu solución es correcta puedes consultar en los foros aprenderaprogramar.com.

Próxima entrega: CU00909C

Acceso al curso completo en aprenderaprogramar.com -- > Cursos, o en la dirección siguiente:
http://aprenderaprogramar.com/index.php?option=com_content&view=category&id=58&Itemid=180