COMO GENERAR NÚMEROS ALEATORIOS EN JAVA

Java · 8 octubre, 2013 · 0 · Iván Salas

Hay ocasiones en las que es necesario usar números aleatorios y en Java tenemos 2 formas sencillas de obtenerlo o bien con Math.random o sino con java.util.Random que nos ofrece algunas posibilidades más aunque su uso es igual de sencillo.

MATH.RANDOM()

Si se opta por la usar la función random de la clase Math el proceso es muy sencillo ya que esta función nos devuelve un número aleatorio entre 0 y 1 (de tipo double) y solo es necesario multiplicarlo por el número que queramos que sea el número más alto posible y/o sumarle (o restarle) algún número para que el limite inferior sea otro distinto de 0.

Esta función es estatica por lo que no hay que instanciar un objeto de la clase Math.

```
int numero = (int)(Math.random()*10+1);
```

En este caso el número generado estará entre 1 y 10, por que primero lo multiplicamos por 10 con lo que tendríamos un número entre 0 y 9 y al sumarle 1 a este número tendremos finalmente un número entre 1 y 10.

De este ejemplo sacamos que el número generado estará entre 0 y el número por el que multiplicamos sin incluirle y para modificar el limite inferior hay que sumar o restar para obtener el limite inferior deseado, por ejemplo si se quieren obtener números entre 25 y 75 multiplicariamos por 51 (y obtendriamos números entre 0 y 25) y le sumariamos 25 y ya tenemos un número aleatorio entre los márgenes que queríamos. La formula para obtener un número entre dos números cual quiera es (X-Y+1)+Y.

```
int numero = (int)(Math.random()*(X-Y+1)+Y;
int numero = (int)(Math.random()*(75-25+1)+25);
```

Y para terminar este caso un ejemplo típico de números aleatorios es la lotería y con esta función y un bucle podemos generar una posible combinación, vamos a generar una combinación para el euromillones y como en la lotería los números no se repiten hay que controlarlo por que al ser números aleatorios podrían repetirse.

```
int numero;
ArrayList numeros = new ArrayList();
ArrayList estrellas = new ArrayList();
// Genera 5 numeros entre 1 y 50
for (int i = 1; i <= 5; i++) {
    numero = (int) (Math.random() * 50 + 1);
    if (numeros.contains(numero)) {
        i--;
    } else {
        numeros.add(numero);
    }
}
System.out.println("La convinacion del Euromillones es:");
for (Integer n : numeros) {
    System.out.println(n + "");
}
// Genera 2 numeros entre 1 y 11
for (int i = 1; i <= 2; i++) {
    numero = (int) (Math.random() * 11 + 1);
    if (estrellas.contains(numero)) {
        i--;
    } else {
        estrellas.add(numero);
    }
}
System.out.println("\nY las estrellas son: ");
for (Integer n : estrellas) {
    System.out.println(n);
}
```

JAVA.UTIL.RANDOM

La dinámica para generar los números aleatorios es igual a la del método anterior pero ofrece un mayor abanico de posibilidades en lo referente al tipo de los números generados (boolean, integer, double, float, Long y arrays de bytes), en la forma en la que se generan los números (distribución gaussiana) y también se permite pasarle una semilla para generar los números a partir de esta aunque hay que tener en cuenta que si es un número fijo se generaran los números siguiendo siempre la misma secuencia por lo que si se crean 2 instancias 2 con la misma semilla ambas producirán la misma secuencia. Si no se indica ninguna semilla por defecto se usa System.currentTimeMillis();.

```
Random numAleatorio = new Random();
// Constructor pasadole una semilla
Random numAleatorio1 = new Random(234);
// Numero entero entre 25 y 75
int n = numAleatorio.nextInt(75-25+1) + 25;
// Asigna una nueva semilla
numAleatorio.setSeed(1235);
// Numero decimal entre 0.0 y 1.0
double d = numAleatorio.nextDouble();
```

En este caso si que es necesario crear una instancia de la clase Random y luego el número aleatorio se genera con los métodos nextXXX() con el objeto de la clase Random previamente instanciado.

Y para probar el uso de java.util.Random vamos a ver un sencillo ejemplo para simular que estamos tirando una moneda al aire:

```
Random numAleatorio = new Random();
// Genera un boolean de forma aleatoria
boolean moneda = numAleatorio.nextBoolean();
// Si es verdadero cara y sino cruz
if (moneda) {
    System.out.println("Cara");
} else {
    System.out.println("Cruz");
}
```

Con esto ya estamos listos para generar los números aleatorios que necesitemos para cualquier cosa que se nos ocurra y si vamos a generar unos números aleatorios y no queremos que se puedan predecir los resultados futuros (o que sea más difícil) podemos usar la clase SecureRandom que a efectos de uso es igual que Random aunque internamente es distinto y no usa la hora del sistema para generar los números aleatorios sino otras formas más complejas.

POST RELACIONADOS

COMO USAR LOG4J EN APLICACIONES WEB
28 abril, 2013 · 1

LIMITAR EL NÚMERO DE DECIMALES DE UN DOUBLE O UN FLOAT EN JAVA
9 noviembre, 2017 · 0

COMO COMPARAR STRING EN JAVA
28 octubre, 2013 · 3

EJEMPLO DE USO DE HASHMAP EN JAVA
15 febrero, 2013 · 17

Recent | Popular

PROGRAMANDO O INTENTÁNDOLO

Home
Java
JavaScript
HTML
CSS
SQL
PHP
Android
TypeScript
WordPress
Kotlin

COMO DESPLEGAR UNA APLICACIÓN SPRING BOOT + ANGULAR EN UN SERVIDOR CLOUD VPS
4 julio, 2019
Angular | Cent
Iván Salas
Hacer pruebas en local esta bien pero tarde o temprano necesitamos desplegar nuestras aplicaciones en un servidor real y lo mejor es hacerlo cuanto antes para no encontramos con sorpresas desagradables por culpa de que ...
Read More

SPRING BOOT PROFILES
3 julio, 2019
Maven | Sprin
Iván Salas
Las aplicaciones raramente se utilizan en único entorno, como mínimo lo normal es tener entornos para desarrollo, qa, preproducción, producción, ... y muy probablemente cada entorno va a tener sus particularidades como por ejemplo la ...
Read More

SPRING BOOT: INTERNACIONALIZACIÓN (I18N)
13 junio, 201
Spring | Sprin
Iván Salas
La internacionalización es un requisito básico para la mayoría de las aplicaciones y con es Spring Boot se puede conseguir de una forma muy sencilla. Vamos a ver cómo convertir la
Read More

FUNCIONES FLECHA O ARROW FUNCTIONS EN JAVASCRIPT
12 mayo, 20
JavaScript
Iván Salas
0
Las funciones flecha o arrow funtions son funciones anónimas con una sintaxis más compacta y que aparte de la diferencia en la sintaxis también tienen algunas peculiaridades como que no vinculan su propio this o ...
Read More

SPRING MVC VALIDATION GROUPS
17 abril, 201
Java | Spring
Iván Salas
Spring MVC Validation Groups Con los validations groups o grupos de validación se puede conseguir que un objeto se pueda validar de formas distintas dependiendo del lugar desde el que se haga. Esto es muy ...
Read More

Previous