<http://blog.sina.com.cn/s/blog_4f925fc30100uvur.html>

|  |
| --- |
| **方法1** (数据类型)(最小值+Math.random()\*(最大值-最小值+1)) 例: (int)(1+Math.random()\*(10-1+1)) 从1到10的int型随数  **方法2** 获得随机数 for (int i=0;i<30;i++) {System.out.println((int)(1+Math.random()\*10));} (int)(1+Math.random()\*10) 通过java.Math包的random方法得到1-10的int随机数 公式是:最小值---最大值（整数）的随机数 （类型）最小值+Math.random()\*最大值  **方法3** Random ra =new Random(); for (int i=0;i<30;i++) {System.out.println(ra.nextInt(10)+1);} 通过java.util包中的Random类的nextInt方法来得到1-10的int随机数    生成0到1之间的任意随机小数：  生成[0,d)区间的随机小数，d为任意正的小数，则只需要将nextDouble方法的返回值乘以d即可。  [n1，n2]  也就是 ra.nextDouble() \* (n2-n1)+n1 |

java产生随机数的几种方式  
一.在j2se里我们可以使用Math.random()方法来产生一个随机数，这个产生的随机数是0-1之间的一个double，我们可以把他乘以一定的数，比如说乘以100，他就是个100以内的随机，这个在j2me中没有。

二.在java.util这个包里面提供了一个Random的类，我们可以新建一个Random的对象来产生随机数，他可以产生随机整数、随机float、随机double，随机long，这个也是我们在j2me的程序里经常用的一个取随机数的方法。

三.在我们的System类中有一个currentTimeMillis()方法，这个方法返回一个从1970年1月1号0点0分0秒到目前的一个毫秒数，返回类型是long，我们可以拿他作为一个随机数，我们可以拿他对一些数取模，就可以把他限制在一个范围之内啦

其实在Random的默认构造方法里也是使用上面第三种方法进行随机数的产生的

对于方法二中的Random类有以下说明：

java.util.Random类有两种方式构建方式：带种子和不带种子

不带种子：  
此种方式将会返回随机的数字，每次运行结果不一样

public class RandomTest {  
public static void main(String[] args) {  
java.util.Random r=new java.util.Random();  
for(int i=0;i<10;i++){  
    System.out.println(r.nextInt());  
}

}  
带种子：  
此种方式，无论程序运行多少次，返回结果都是一样的

public static void main(String[] args) {  
java.util.Random r=new java.util.Random(10);  
for(int i=0;i<10;i++){  
    System.out.println(r.nextInt());  
}  
}

两种方式的差别在于

(1) 首先请打开Java Doc，我们会看到Random类的说明：

此类的实例用于生成伪随机数流，此类使用 48 位的种子，该种子可以使用线性同余公式对其进行修改（请参阅 Donald Knuth 的《The Art of Computer Programming, Volume 2》，第 3.2.1 节）。

如果用相同的种子创建两个 Random 实例，则对每个实例进行相同的方法调用序列，它们将生成并返回相同的数字序列。为了保证实现这种特性，我们为类Random指定了特定的算法。为了 Java 代码的完全可移植性，Java 实现必须让类 Random 使用此处所示的所有算法。但是允许 Random 类的子类使用其他算法，只要其符合所有方法的常规协定即可。

Java Doc对Random类已经解释得非常明白，我们的测试也验证了这一点。

(2) 如果没有提供种子数，Random实例的种子数将是当前时间的毫秒数，可以通过System.currentTimeMillis()来获得当前时间的毫秒数。打开JDK的源代码，我们可以非常明确地看到这一点。

public Random() { this(System.currentTimeMillis()); }

另外：

random对象的nextInt(),nextInt(int n)方法的说明：

int nextInt()  
    返回下一个伪随机数，它是此随机数生成器的序列中均匀分布的 int 值。  
int nextInt(int n)  
    返回一个伪随机数，它是从此随机数生成器的序列中取出的、在 0（包括）和指定值（不包括）之间均匀分布的 int值。

**Java随机数总结**

　　随机数在实际中使用很广泛，比如要随即生成一个固定长度的字符串、数字。或者随即生成一个不定长度的数字、或者进行一个模拟的随机选择等等。Java提供了最基本的工具，可以帮助开发者来实现这一切。

　　一、Java随机数的产生方式

　　在Java中，随机数的概念从广义上将，有三种。

　　1、通过System.currentTimeMillis()来获取一个当前时间毫秒数的long型数字。

　　2、通过Math.random()返回一个0到1之间的double值。

　　3、通过Random类来产生一个随机数，这个是专业的Random工具类，功能强大。

　　二、Random类API说明

　　1、Java API说明

　　Random类的实例用于生成伪随机数流。此类使用 48 位的种子，使用线性同余公式对其进行修改(请参阅 Donald Knuth 的《The Art of Computer Programming， Volume 2》，第 3.2.1 节)。

　　如果用相同的种子创建两个 Random 实例，则对每个实例进行相同的方法调用序列，它们将生成并返回相同的数字序列。为了保证属性的实现，为类 Random 指定了特定的算法。

　　很多应用程序会发现 Math 类中的 random 方法更易于使用。

　　2、方法摘要

　　Random()

　　创建一个新的随机数生成器。

　　Random(long seed)

　　使用单个 long 种子创建一个新随机数生成器： public Random(long seed) { setSeed(seed); } next 方法使用它来保存随机数生成器的状态。

　　protected int next(int bits)

　　生成下一个伪随机数。

　　boolean nextBoolean()

　　返回下一个伪随机数，它是从此随机数生成器的序列中取出的、均匀分布的 boolean 值。

　　void nextBytes(byte[] bytes)

　　生成随机字节并将其置于用户提供的字节数组中。

　　double nextDouble()

　　返回下一个伪随机数，它是从此随机数生成器的序列中取出的、在 0.0 和 1.0之间均匀分布的 double 值。

　　float nextFloat()

　　返回下一个伪随机数，它是从此随机数生成器的序列中取出的、在 0.0 和 1.0 之间均匀分布的 float 值。

　　double nextGaussian()

　　返回下一个伪随机数，它是从此随机数生成器的序列中取出的、呈高斯(“正常地”)分布的 double 值，其平均值是 0.0，标准偏差是 1.0。

　　int nextInt()

　　返回下一个伪随机数，它是此随机数生成器的序列中均匀分布的 int 值。

　　int nextInt(int n)

　　返回一个伪随机数，它是从此随机数生成器的序列中取出的、在 0(包括)和指定值(不包括)之间均匀分布的 int值。

　　long nextLong()

　　返回下一个伪随机数，它是从此随机数生成器的序列中取出的、均匀分布的 long 值。

　　void setSeed(long seed)

　　使用单个 long 种子设置此随机数生成器的种子。

　　三、Random类使用说明

　　1、带种子与不带种子的区别Random类使用的根本是策略分带种子和不带种子的Random的实例。

　　通俗说，两者的区别是：带种子的，每次运行生成的结果都是一样的。

　　不带种子的，每次运行生成的都是随机的，没有规律可言。

　　2、创建不带种子的Random对象

　　Random random = new Random();

　　3、创建不带种子的Random对象有两种方法：

　　1) Random random = new Random(555L);

　　2) Random random = new Random();random.setSeed(555L);

　　四、测试

　　通过一个例子说明上面的用法

　　import java.util.Random;

　　public class TestRandomNum {

　　public static void main(String[] args) {

　　randomTest();

　　testNoSeed();

　　testSeed1();

　　testSeed2();

　　}

　　public static void randomTest() {

　　System.out.println("--------------test()--------------");

　　//返回以毫秒为单位的当前时间。

　　long r1 = System.currentTimeMillis();

　　//返回带正号的 double 值，大于或等于 0.0，小于 1.0。

　　double r2 = Math.random();

　　//通过Random类来获取下一个随机的整数

　　int r3 = new Random().nextInt();

　　System.out.println("r1 = " + r1);

　　System.out.println("r3 = " + r2);

　　System.out.println("r2 = " + r3);

　　}

　　public static void testNoSeed() {

　　System.out.println("--------------testNoSeed()--------------");

　　//创建不带种子的测试Random对象

　　Random random = new Random();

　　for (int i = 0; i < 3; i++) {

　　System.out.println(random.nextInt());

　　}

　　}

　　public static void testSeed1() {

　　System.out.println("--------------testSeed1()--------------");

　　//创建带种子的测试Random对象

　　Random random = new Random(555L);

　　for (int i = 0; i < 3; i++) {

　　System.out.println(random.nextInt());

　　}

　　}

　　public static void testSeed2() {

　　System.out.println("--------------testSeed2()--------------");

　　//创建带种子的测试Random对象

　　Random random = new Random();

　　random.setSeed(555L);

　　for (int i = 0; i < 3; i++) {

　　System.out.println(random.nextInt());

　　}

　　}

　　}

　　运行结果：

　　--------------test()--------------

　　r1 = 1227108626582

　　r3 = 0.5324887850155043

　　r2 = -368083737

　　--------------testNoSeed()--------------

　　809503475

　　1585541532

　　-645134204

　　--------------testSeed1()--------------

　　-1367481220

　　292886146

　　-1462441651

　　--------------testSeed2()--------------

　　-1367481220

　　292886146

　　-1462441651

　　Process finished with exit code 0

　　通过testSeed1()与testSeed2()方法的结果可以看到，两个打印结果相同，因为他们种子相同，再运行一次，结果还是一样的，这就是带种子随机数的特性。

　　而不带种子的，每次运行结果都是随机的。

　　五、综合应用

　　下面通过最近写的一个随机数工具类来展示用法：

　　import java.util.Random;

　　public class RandomUtils {

　　public static final String allChar = "0123456789abcdefghijklmnopqrstuvwxyzABCDEFGHIJKLMNOPQRSTUVWXYZ";

　　public static final String letterChar = "abcdefghijklmnopqrstuvwxyzABCDEFGHIJKLMNOPQRSTUVWXYZ";

　　public static final String numberChar = "0123456789";

　　public static String generateString(int length) {

　　StringBuffer sb = new StringBuffer();

　　Random random = new Random();

　　for (int i = 0; i < length; i++) {

　　sb.append(allChar.charAt(random.nextInt(allChar.length())));

　　}

　　return sb.toString();

　　}

　　public static String generateMixString(int length) {

　　StringBuffer sb = new StringBuffer();

　　Random random = new Random();

　　for (int i = 0; i < length; i++) {

　　sb.append(allChar.charAt(random.nextInt(letterChar.length())));

　　}

　　return sb.toString();

　　}

　　public static String generateLowerString(int length) {

　　return generateMixString(length).toLowerCase();

　　}

　　public static String generateUpperString(int length) {

　　return generateMixString(length).toUpperCase();

　　}

　　public static String generateZeroString(int length) {

　　StringBuffer sb = new StringBuffer();

　　for (int i = 0; i < length; i++) {

　　sb.append('0');

　　}

　　return sb.toString();

　　}

　　public static String toFixdLengthString(long num, int fixdlenth) {

　　StringBuffer sb = new StringBuffer();

　　String strNum = String.valueOf(num);

　　if (fixdlenth - strNum.length() >= 0) {

　　sb.append(generateZeroString(fixdlenth - strNum.length()));

　　} else {

　　throw new RuntimeException("将数字" + num + "转化为长度为" + fixdlenth + "的字符串发生异常!");

　　}

　　sb.append(strNum);

　　return sb.toString();

　　}

　　public static String toFixdLengthString(int num, int fixdlenth) {

　　StringBuffer sb = new StringBuffer();

　　String strNum = String.valueOf(num);

　　if (fixdlenth - strNum.length() >= 0) {

　　sb.append(generateZeroString(fixdlenth - strNum.length()));

　　} else {

　　throw new RuntimeException("将数字" + num + "转化为长度为" + fixdlenth + "的字符串发生异常!");

　　}

　　sb.append(strNum);

　　return sb.toString();

　　}

　　public static void main(String[] args) {

　　System.out.println(generateString(15));

　　System.out.println(generateMixString(15));

　　System.out.println(generateLowerString(15));

　　System.out.println(generateUpperString(15));

　　System.out.println(generateZeroString(15));

　　System.out.println(toFixdLengthString(123, 15));

　　System.out.println(toFixdLengthString(123L, 15));

　　}

　　}

　　运行结果：

　　vWMBPiNbzfGCpHG

　　23hyraHdJkKPwMv

　　tigowetbwkm1nde

　　BPZ1KNEJPHB115N

　　000000000000000

　　000000000000123

　　000000000000123

　　Process finished with exit code 0

　　六、总结

　　1、随机数很常用，在Java有三种产生方式，以Random随机数的使用最为复杂。

　　2、Random类对象有是否带种子之分，带种子的只要种子相同，多次运行，生成随机数的结果总是那样。

　　3、带种子随机数的带种子的对象创建方式有两种，效果一样。但是带种子的随机数用处似乎不大。

　　4、Random的功能涵盖了Math.random()的功能。

　　5、可以通过随机数去做实现随机字符串等复杂的随机数据。

　　6、不要研究不重复的随机数，意义不大。

在Java 中我们可以使用java.util.Random类来产生一个随机数发生器。它有两种形式的构造函数，分别是Random()和Random(long seed)。Random()使用当前时间即System.currentTimeMillis()作为发生器的种子，Random(long seed)使用指定的seed作为发生器的种子。  
  
        随机数发生器(Random)对象产生以后，通过调用不同的method：nextInt()、nextLong()、nextFloat()、nextDouble()等获得不同类型随机数。  
  
       1>生成随机数  
           Random random = new Random();  
           Random random = new Random(100);//指定种子数100  
           random调用不同的方法，获得随机数。  
           如果2个Random对象使用相同的种子（比如都是100），并且以相同的顺序调用相同的函数，那它们返回值完全相同。如下面代码中两个Random对象的输出完全相同  
          import java.util.\*;  
          class TestRandom {  
                public static void main(String[] args) {  
                     Random random1 = new Random(100);  
                     System.out.println(random1.nextInt());  
                     System.out.println(random1.nextFloat());  
                     System.out.println(random1.nextBoolean());  
                     Random random2 = new Random(100);  
                     System.out.println(random2.nextInt());  
                     System.out.println(random2.nextFloat());  
                     System.out.println(random2.nextBoolean());  
                }  
            }  
  
        2>指定范围内的随机数  
             随机数控制在某个范围内,使用模数运算符%  
            import java.util.\*;  
                 class TestRandom {  
                      public static void main(String[] args) {  
                           Random random = new Random();  
                           for(int i = 0; i < 10;i++) {  
                              System.out.println(Math.abs(random.nextInt()));  
                           }  
                      }  
                 }  
             获得的随机数有正有负的，用Math.abs使获取数据范围为非负数  
  
       3>获取指定范围内的不重复随机数  
            import java.util.\*;  
            class TestRandom {  
                  public static void main(String[] args) {  
                       int[] intRet = new int[6];  
                       int intRd = 0; //存放随机数  
                       int count = 0; //记录生成的随机数个数  
                       int flag = 0; //是否已经生成过标志  
                       while(count<6){  
                            Random rdm = new Random(System.currentTimeMillis());  
                            intRd = Math.abs(rdm.nextInt())2+1;  
                            for(int i=0;i<count;i++){  
                                if(intRet[i]==intRd){  
                                    flag = 1;  
                                    break;  
                                }else{  
                                    flag = 0;  
                                }  
                            }  
                            if(flag==0){  
                                intRet[count] = intRd;  
                                count++;  
                            }  
                   }  
                  for(int t=0;t<6;t++){  
                      System.out.println(t+"->"+intRet[t]);  
                  }  
               }  
            }  
Java中的随机数是否可以重复？Java中产生的随机数能否可以用来产生数据库主键？带着这个问题，我们做了一系列测试。  
1.测试一： 使用不带参数的Random()构造函数  
  
public class RandomTest {  
  
public static void main(String[] args) {  
     java.util.Random r=new java.util.Random();  
     for(int i=0;i<10;i++){  
         System.out.println(r.nextInt());  
     }  
}  
}  
程序运行结果：  
-1761145445  
-1070533012  
216216989  
-910884656  
-1408725314  
-1091802870  
1681403823  
-1099867456  
347034376  
-1277853157  
  
再次运行该程序：  
-169416241  
220377062  
-1140589550  
-1364404766  
-1088116756  
2134626361  
-546049728  
1132916742  
-1522319721  
1787867608  
  
    从上面的测试我们可以看出，使用不带参数的Random()构造函数产生的随机数不会重复。那么，什么情况下Java会产生重复的随机数呢？且看下面的测试。  
  
2. 测试二：为Random设置种子数  
  
public class RandomTest\_Repeat {  
     
    public static void main(String[] args) {  
       java.util.Random r=new java.util.Random(10);  
       for(int i=0;i<10;i++){  
           System.out.println(r.nextInt());  
       }  
   }  
}  
  
无论程序运行多少次，其结果总是：  
-1157793070  
1913984760  
1107254586  
1773446580  
254270492  
-1408064384  
1048475594  
1581279777  
-778209333  
1532292428  
  
甚至在不同的机器上测试，测试结果也不会改变！  
  
3.原因分析：  
(1) 首先请打开Java Doc，我们会看到Random类的说明：  
此类的实例用于生成伪随机数流，此类使用 48 位的种子，该种子可以使用线性同余公式对其进行修改（请参阅 Donald Knuth 的《The Art of Computer Programming, Volume 2》，第 3.2.1 节）。  
如果用相同的种子创建两个 Random 实例，则对每个实例进行相同的方法调用序列，它们将生成并返回相同的数字序列。为了保证实现这种特性，我们为类Random指定了特定的算法。为了 Java 代码的完全可移植性，Java 实现必须让类 Random 使用此处所示的所有算法。但是允许 Random 类的子类使用其他算法，只要其符合所有方法的常规协定即可。  
Java Doc对Random类已经解释得非常明白，我们的测试也验证了这一点。  
  
(2) 如果没有提供种子数，Random实例的种子数将是当前时间的毫秒数，可以通过System.currentTimeMillis()来获得当前时间的毫秒数。打开JDK的源代码，我们可以非常明确地看到这一点。  
     
     public Random() { this(System.currentTimeMillis()); }  
  
4. 结论：  
   通过上面的测试和分析，我们会对Random类有较为深刻的理解。同时，我觉得，通过阅读Java Doc的API文档，可以很好地提高我们的Java编程能力，做到“知其然”；一旦遇到费解的问题，不妨打开Java的源代码，这样我们就能做到“知其所以然”。

--------------------------------------------------------------------------------------------------

java中一般有两种随机数，一个是Math中random()方法，一个是Random类。

一、Math.random()

随即生成0<x<1的小数。

实例：如何写，生成随机生成出0~100中的其中一个数呢？

    Math.random()返回的只是从0到1之间的小数，如果要50到100，就先放大50倍，即0到50之间，这里还是小数，如果要整数，就强制转换int，然后再加上50即为50~100.  
最终代码：(int)(Math.random()\*50) + 50

二、Random类

Random random = new Random();//默认构造方法

Random random = new Random(1000);//指定种子数字

在进行随机时，随机算法的起源数字称为种子数(seed)，在种子数的基础上进行一定的变换，从而产生需要的随机数字。

相同种子数的Random对象，相同次数生成的随机数字是完全相同的。也就是说，两个种子数相同的Random对象，第一次生成的随机数字完全相同，第二次生成的随机数字也完全相同。

2 、Random类中的常用方法

Random 类中的方法比较简单，每个方法的功能也很容易理解。需要说明的是，Random类中各方法生成的随机数字都是均匀分布的，也就是说区间内部的数字生成的几率是均等的。下面对这些方法做一下基本的介绍：

a 、public boolean nextBoolean()

该方法的作用是生成一个随机的boolean值，生成true和false的值几率相等，也就是都是50%的几率。

b 、public double nextDouble()

该方法的作用是生成一个随机的double值，数值介于[0,1.0)之间，这里中括号代表包含区间端点，小括号代表不包含区间端点，也就是0到1之间的随机小数，包含0而不包含1.0。

c 、public int nextInt()

该方法的作用是生成一个随机的int值，该值介于int的区间，也就是-2的31次方到2的31次方-1之间。

如果需要生成指定区间的int值，则需要进行一定的数学变换，具体可以参看下面的使用示例中的代码。

d 、public int nextInt(int n)

该方法的作用是生成一个随机的int值，该值介于[0,n)的区间，也就是0到n之间的随机int值，包含0而不包含n。

如果想生成指定区间的int值，也需要进行一定的数学变换，具体可以参看下面的使用示例中的代码。

e 、public void setSeed(long seed)

该方法的作用是重新设置Random对象中的种子数。设置完种子数以后的Random对象和相同种子数使用new关键字创建出的Random对象相同。

3 、Random类使用示例

使用Random类，一般是生成指定区间的随机数字，下面就一一介绍如何生成对应区间的随机数字。以下生成随机数的代码均使用以下Random对象r进行生成：

Random r = new Random();

a 、生成[0,1.0)区间的小数

 double d1 = r.nextDouble();  
直接使用nextDouble方法获得。

b、生成[0,5.0)区间的小数

double d2 = r.nextDouble() \* 5;

因为nextDouble方法生成的数字区间是[0,1.0)，将该区间扩大5倍即是要求的区间。

同理，生成[0,d)区间的随机小数，d为任意正的小数，则只需要将nextDouble方法的返回值乘以d即可。

c、生成[1,2.5)区间的小数  [n1，n2]

       double d3 = r.nextDouble() \* 1.5 + 1;【也就是 r.nextDouble() \* (n2-n1)+n1】

生成[1,2.5)区间的随机小数，则只需要首先生成[0,1.5)区间的随机数字，然后将生成的随机数区间加1即可。

同理，生成任意非从0开始的小数区间[d1,d2)范围的随机数字(其中d1不等于0)，则只需要首先生成[0,d2-d1)区间的随机数字，然后将生成的随机数字区间加上d1即可。

d、生成任意整数

int n1 = r.nextInt();

直接使用nextInt方法即可。

e、生成[0,10)区间的整数

int n2 = r.nextInt(10);

n2 = Math.abs(r.nextInt() % 10);

以上两行代码均可生成[0,10)区间的整数。

第一种实现使用Random类中的nextInt(int n)方法直接实现。

第二种实现中，首先调用nextInt()方法生成一个任意的int数字，该数字和10取余以后生成的数字区间为(-10,10)，因为按照数学上的规定余数的绝对值小于除数，然后再对该区间求绝对值，则得到的区间就是[0,10)了。

同理，生成任意[0,n)区间的随机整数，都可以使用如下代码：

int n2 = r.nextInt(n);

n2 = Math.abs(r.nextInt() % n);

f、生成[0,10]区间的整数

int n3 = r.nextInt(11);

n3 = Math.abs(r.nextInt() % 11);

相对于整数区间，[0,10]区间和[0,11)区间等价，所以即生成[0,11)区间的整数。

g、生成[-3,15)区间的整数

int n4 = r.nextInt(18) - 3;   【也就是 r.nextInt() \* (n2-n1)+n1】 n1是个负数

n4 = Math.abs(r.nextInt() % 18) - 3;

生成非从0开始区间的随机整数，可以参看上面非从0开始的小数区间实现原理的说明。