

Random

INHERITS FROM

Object

CLASS DESCRIPTION

The Random class provides services for random number generation and die rolling. It implements its own random number generator with a cycle length of 8.8 trillion. The algorithm used by the Random class is that given in the article:

^aA Highly Random Random±Number Generator^o by T.A. Elkins
Computer Language, Volume 6, Number 12 (December 1989), Pages 59-65
Published by:
Miller Freeman Publications
500 Howard Street
San Francisco, CA 94105
(415) 397-1881

This is Version 1.1 of Random, distributed 1992 Feb 27.

Written by Gregor Purdy, Contemporary Design Studios

Real Job Contact Information (NOT Contemporary Design Studios):
gregor@umich.edu
University of Michigan / 1600 SEB / 610 E. University / Ann Arbor / MI / 48109

THIS WORK IS DISTRIBUTED AS IS, WITH NO WARANTEE OR GUARANTEE EXPRESSED OR IMPLIED IN ANY RESPECT. THE AUTHOR IS NOT LIABLE FOR ANY DAMAGES WHATSOEVER

DIRECTLY OR INDIRECTLY RELATED TO THE USAGE OF THIS WORK.

That said, I do welcome comments, suggestions, and bug reports. I want to use this in some of my own projects, so I'm very interested in making sure it works correctly. Feel free to drop me an email or letter with your comments.

This work is distributed as FreeWare. Previous versions were further restricted under the GNU Public License, version 1. This version is no longer so restricted.

This version of the Random class may be used by anyone, anywhere, for anything, with the following conditions:

- (i) The usage must not be illegal in any way.
- (ii) You must credit Contemporary Design Studios and Gregor N. Purdy and include the copyright information at the top of this document in your documentation and in your "Info Panel," if any, if your work is ever used by anyone other than yourself.
- (iii) Contemporary Design Studios and Gregor N. Purdy retain the right to make improvements to this work and to change the distribution conditions and terms in future versions.
- (iv) You may NOT in any way change the Random class and redistribute it, even under a new name, but you MAY, and in fact are encouraged to, report bugs and make suggestions for enhancements to Contemporary Design Studios and Gregor N. Purdy.
- (iv) You MAY make subclasses of Random that do whatever you want. Of course, these subclasses will be governed by your own rules, but the Random class will still be governed by these rules.
- (v) You MAY distribute the source code of the Random class with your work, but all the documentation for Random, including this notice, must be distributed with it. Nobody should ever see the source code to the Random class without also seeing these documentation files, so they can use it, too.

INSTANCE VARIABLES

<i>Declared in Random</i>	int	h1, h2, h3;
	int iset;	
	double	gset;
	double	gscale, gorigin;

h1, h2, h3
iset
gset
gscale, gorigin

The current seed values
A flag indicating a stored Gaussian result (from a pair)
The stored Gaussian result, if any
The scale and origin parameters for Gaussian generation

METHOD TYPES

Creating and freeing instances

- + alloc
- free

Getting the class version

- + version

Initializing a new instance

- init
- initSeeds:::

Getting and setting seeds

- newSeeds
- setSeeds:::
- getSeeds:::

Gaussian parameters

- gOrigin
- gScale
- setGOrigin:
- setGScale:

Getting random numbers - bool

- gaussian
- percent
- rand
- randFunc:
- randMax:
- randMin:max:

Rolling dice

- rollDie:
- roll:die:
- rollBest:of:die:

Archiving

- read:

- write:

CLASS METHODS

alloc

+ **alloc**

Returns a new uninitialized instance.

version

+ **version**

Returns the version number of this class, currently 2.

INSTANCE METHODS

bool

- (BOOL)**bool**

Returns a random Boolean value.

free

- **free**

Frees the memory occupied by the Random and returns **nil**.

gaussian

- (double)**gaussian**

Returns a random double gaussian value.

The algorithm used to calculate the Gaussian variables is from the book:

^aNumerical Recipes in C^o by William H. Press, Saul A. Teukolsky, Brian P. Flannery, and William T. Vetterling. Published by Cambridge University Press. ISBN 0-521-35465-X

getSeeds:::

- **getSeeds**:(int *)s1 :(int *)s2 :(int *)s3

Puts the values of the seeds into the integer variables pointed to.

See also: \pm **setSeeds:::**

gOrigin

- (double)**gOrigin**

Returns the origin of the Gaussian variables the instance is generating.

gScale

- (double)**gScale**

Returns the scaling factor of the Gaussian variables the instance is generating.

init

- **init**

Initializes the Random with seeds from the milliseconds count of the system clock (uses **newSeeds**).

See also: \pm **initSeeds:::**, \pm **newSeeds**

initSeeds:::

- **initSeeds**:(int)s1 :(int)s2 :(int)s3

Initializes the Random with the seeds given (uses **setSeeds**).

See also: \pm **init**, \pm **newSeeds**, \pm **setSeeds:::**

newSeeds

- **newSeeds**

Sets the seeds from the milliseconds count of the system clock.

See also: \pm **init**

percent

- (double)**percent**

Returns a double in the range [0.0, 1.0].

rand

- (int)**rand**

Returns an int in the range [0, 32767].

randFunc:

- (double)**rollDie:(ddfunc)func**

Returns a double which is the result of applying the function *func* to a random percentage.

This is useful for transforming the uniform random numbers Random returns into a non-uniform distribution of your choice.

randMax:

- (int)**randMax:(int)max**

Returns an int in the range [0, *max*].

randMin: max:

- (int)**randMin:(int)min max:(int)max**

Returns an int in the range [*min*, *max*].

read:

- **read:(NXTypedStream *)stream**

Reads a Random from *stream*.

See also: - **write:**

rollDie:

- (int)**rollDie:(int)numSides**

Returns an int in the range [1, *numSides*].

roll: die:

- (int)**roll**:(int)*numRolls* **die**:(int)*numSides*

Returns an int in the range [*numRolls*, *numRolls* * *numSides*].

rollBest: of: die:

- (int)**rollBest**:(int)*numWanted* **of**:(int)*numRolls* **die**:(int)*numSides*

Returns the sum of the best *numWanted* rolls..

setGOrigin:

- **setGOrigin**:(double)*anOrigin*

Sets the Random instance to generate subsequent Gaussian variables around the origin *anOrigin*.

setGScale:

- **setGScale**:(double)*aScale*

Sets the Random instance to scale subsequent Gaussian variables by *aScale*.

setSeeds:::

- **setSeeds**:(int)*s1* :(int)*s2* :(int)*s3*

Sets the seeds to the values given.

See also: ± **getSeeds:::**

write:

- **write**:(NXTypedStream *)*stream*

Writes a Random to *stream*.

See also: - **read**:

DEFINED TYPES

```
/* Double Function Returning Double */  
typedef double (*ddfunc) (double);
```