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 Higly Random Random±Number Generatorº 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 1.0 of Random, distributed 1991 May 30.

Written by Gregor Purdy, Contemporary Design Studios

Real Job Contact Information (NOT Contemporary Design Studios):
gregor@oit.itd.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. Its usage and distribution, however, are governed by the GNU ªCopyleftº (version 1) described in the file ªCOPYINGº which should be in the same directory as this file, if everything went well with downloading, uncompressing, and untarring this distribution. At your option, you may consider this work governed by a later release of the GNU ªCopyleftº.

This README and the file COPYING must remain with any derivative works.

See the COPYING file included for distribution and usage rights.

INSTANCE VARIABLES

Declared in Random	int	h1, h2, h3;
h1, h2, h3	The current seed values	

METHOD TYPES

Creating and freeing instances	+ alloc - free
Initializing a new instance	- init - initSeeds:::
Seed operations	- newSeeds - setSeeds::: - getSeeds:::
	Getting random numbers - rand - randMax: - randMin:max: - percent
Rolling dice	- rollDie: - roll:die: - rollBest:of:die:
Archiving	- read:

- write:

CLASS METHODS

alloc
> **+ alloc**

Returns a new instance.

INSTANCE METHODS

free
> **- free**

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

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

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

See also: ± **setSeeds:::**

init
> **- init**

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

See also: ± **initSeeds:::**, ± **newSeeds**

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

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

See also: ± **init**, ± **newSeeds**, ± **setSeeds:::**

newSeeds

- newSeeds

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

See also: ± **init**

percent
- (float)percent

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

rand
- (int)rand

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

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..

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:**