Random

INHERITS FROM Object

CLASS DESCRIPTION

The Random class provides services for random number generation. It uses an instance of a subclass of RandomEngine as a source for sequences of random bits (in byte-sized chunks).

This class is part of Version 2.0 of Random, distributed 1992 May 29.

Contemporary Design Studios
2339 Woodchip Way #1B
Ypsilanti, MI 48197
313-572-1779

Gregor N. Purdy
Software Designer
University of Michigan
gregor@umich.edu

WHILE THE AUTHOR DOES WORK FOR THE UNIVERSITY OF MICHIGAN, AND PROVIDES HIS EMAIL ADDRESS AT WORK, THIS PRODUCT IS IN NO WAY AFFILIATED WITH THE UNIVERSITY OF MICHIGAN. ALL THE DESIGN, CODING, TESTING, AND DOCUMENTATION IS DONE ON THE AUTHOR'S OWN TIME.

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.

Comments, suggestions, and bug reports are welcome. Feel free to drop us an email or letter with your comments.

This work is distributed as FreeWare. Version 1.0 was further restricted under the GNU Public License, version 1. Version 1.1 had the same restrictions as this version.

This version of the Random Classes 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 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 retains the right to make improvements to this work and to change the distribution conditions and terms in future versions, including commercial derivative works.

(iv) You may NOT in any way change the Random Classes and redistribute them, 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.

(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 Classes will still be governed by these rules.

(v) You MAY distribute the source code of the Random Classes with your work, but if you do all the documentation for the Random Classes, including this notice, must be distributed with it. Nobody should ever see the source code to the Random Classes without also seeing these documentation files, so they can use it, too.

INSTANCE VARIABLES

Declared in Random

```
id          engineClass;
ulong       unit;
uchar       bitbuffer;
uchar       *bytebuffer;
uchar       curbit;
ulong       curbyte;
id    engine;
```

engineClass	The id of the class of our random bit source
unit	How many bytes of random bits we get for each request of engine
bitbuffer	One byte worth of bits for use in random booleans
bytebuffer	A buffer our engine will fill when empty from which we grab bytes
curbit	The current bit in bitbuffer ready to be used
curbyte	The current byte bytebuffer ready to be used
engine	The id of our engine instance

METHOD TYPES

Creating and freeing instances	+ alloc - free
Getting the class version	+ version
Initializing a new instance	- initEngineClass: - initEngineInstance:

Getting random numbers - bool
- percent
- rand
- randFunc:
- randMax:

	- randMin:max:
Archiving	- read: - write:

CLASS METHODS

alloc

+ **alloc**

Returns a new uninitialized instance.

version

+ **version**

Returns the version number of this class, currently 3.

INSTANCE METHODS

bool

- (BOOL)**bool**

Returns a random Boolean value.

free

- **free**

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

initEngineClass:

- **initEngineClass:***aClass*

Initializes the Random with a newly allocated instance of the class *aClass* as its engine.

See also: ± **initEngineInstance:**

initEngineInstance:

- **initEngineInstance:***anObject*

Initializes the Random with the *anObject* as its engine. *anObject* must be an instance of a subclass of RandomEngine. It is an error to pass in an object which is not.

See also: ± **initEngineClass:**

percent

- (double)**percent**

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

rand

- (int)**rand**

Returns an int in the range [0, RAND_RANGE]. RAND_RANGE is the maximum unsigned long integer value possible.

randFunc:

- (double)**randFunc:**(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*

Unarchives a Random from *stream*.

See also: - **write:**

write:

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

Archives a Random to *stream*.

See also: - **read:**

DEFINED TYPES

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