Project3

3

Generated by Doxygen 1.8.15

1 Hierarchical Index **1**

1.1 Class Hierarchy 1

2 Class Index **3**

2.1 Class List 3

3 File Index **5**

3.1 File List 5

4 Class Documentation **7**

4.1 DifferenialEvolution Class Reference 7

4.1.1 Detailed Description 7

4.2 FunctionClass Class Reference 7

4.2.1 Detailed Description 7

4.3 FunctionClass.Functions Enum Reference 8

4.3.1 Detailed Description 8

4.3.2 Member Function Documentation 8

4.3.2.1 function() 8

4.3.3 Member Data Documentation 8

4.3.3.1 function 8

4.4 GeneticAlgorithm Class Reference 9

4.4.1 Detailed Description 9

4.5 Main Class Reference 9

4.5.1 Detailed Description 9

4.5.2 Member Function Documentation 9

4.5.2.1 main() 9

4.6 Matrix Class Reference 10

4.6.1 Detailed Description 10

4.6.2 Constructor & Destructor Documentation 10

4.6.2.1 Matrix() 10

4.7 Project3.MTRandom Class Reference 11

4.7.1 Detailed Description 11

4.7.2 Constructor & Destructor Documentation 12

4.7.2.1 MTRandom() `[1/5]` 12

4.7.2.2 MTRandom() `[2/5]` 12

4.7.2.3 MTRandom() `[3/5]` 12

4.7.2.4 MTRandom() `[4/5]` 13

4.7.2.5 MTRandom() `[5/5]` 13

4.7.3 Member Function Documentation 14

4.7.3.1 next() 14

4.7.3.2 pack() 14

4.7.3.3 setSeed() `[1/3]` 15

4.7.3.4 setSeed() `[2/3]` 15

4.7.3.5 setSeed() [3/3] 16
4.8 Population Class Reference 16
4.8.1 Detailed Description 17
4.8.2 Member Function Documentation 17
4.8.2.1 sortByCostAscending() 17
4.9 Support Class Reference 17
4.9.1 Detailed Description 17

5 File Documentation **19**
5.1 D:/study/CS471/Lu_Project3/src/Project3/FunctionClass.java File Reference 19
5.1.1 Detailed Description 19
5.2 D:/study/CS471/Lu_Project3/src/Project3/Main.java File Reference 19
5.2.1 Detailed Description 20
5.3 D:/study/CS471/Lu_Project3/src/Project3/Matrix.java File Reference 20
5.3.1 Detailed Description 20
5.4 D:/study/CS471/Lu_Project3/src/Project3/Support.java File Reference 20
5.4.1 Detailed Description 21

Chapter 1

Hierarchical Index

1.1 Class Hierarchy

This inheritance list is sorted roughly, but not completely, alphabetically:

DifferenialEvolution 7
FunctionClass 7
FunctionClass.Functions 8
GeneticAlgorithm 9
Main 9
Matrix 10
Population 16
Support 17
Random
    Project3.MTRandom 11

Chapter 2

Class Index

2.1 Class List

Here are the classes, structs, unions and interfaces with brief descriptions:

DifferenialEvolution 7
FunctionClass 7
FunctionClass.Functions 8
GeneticAlgorithm 9
Main 9
Matrix 10
Project3.MTRandom 11
Population 16
Support 17

Chapter 3

File Index

3.1 File List

Here is a list of all documented files with brief descriptions:

D:/study/CS471/Lu_Project3/src/Project3/FunctionClass.java
File for functions 19
D:/study/CS471/Lu_Project3/src/Project3/Main.java
Main file of CS471 project2 19
D:/study/CS471/Lu_Project3/src/Project3/Matrix.java
File for matrix modification 20
D:/study/CS471/Lu_Project3/src/Project3/Support.java
File for support tool for project2 20

Chapter 4

Class Documentation

4.1 DifferenialEvolution Class Reference

Public Member Functions

- **DifferenialEvolution** (Population p, BufferedReader br)
- double **DE** (Population p, BufferedWriter bw)

4.1.1 Detailed Description

Author

Junyu Lu

The documentation for this class was generated from the following file:

- D:/study/CS471/Lu_Project3/src/Project3/DifferenialEvolution.java

4.2 FunctionClass Class Reference

Classes

- enum Functions

4.2.1 Detailed Description

Function class with enum of 18 functions and the function method. Each enum has a function related to its function name.

Author

Junyu Lu

The documentation for this class was generated from the following file:

- D:/study/CS471/Lu_Project3/src/Project3/FunctionClass.java

4.3 FunctionClass.Functions Enum Reference

Public Member Functions

- abstract double function (double[ ] x, int d)

Public Attributes

- function

4.3.1 Detailed Description

An enum type. 18 enums named with its related function.

4.3.2 Member Function Documentation

4.3.2.1 function()

```
abstract double FunctionClass.Functions.function (
              double [] x,
              int d )  [abstract]
```

the abstract

Parameters

x	input vector
d	dimension of the input vector

Returns

output of the function as a double value

4.3.3 Member Data Documentation

4.3.3.1 function

```
FunctionClass.Functions.function
```

Enum value schwefel1. Enum value deJong2. Enum value rosenbrcksSaddle3. Enum value rastrigin4. Enum value griewangk5. Enum value sinEnvelopSinWave6. Enum value stretchedVSinWave7. Enum value ackleysOne8.

Enum value ackleysTwo9. Enum value eggHolder10. Enum value rana11. Enum value pathological12. Enum value michalewicz13. Enum value masterCosineWave14. Enum value quartic15. Enum value levy16. Enum value step17. Enum value alpine18.

The documentation for this enum was generated from the following file:

- D:/study/CS471/Lu_Project3/src/Project3/FunctionClass.java

4.4 GeneticAlgorithm Class Reference

Public Member Functions

- **GeneticAlgorithm** (Population p, BufferedReader br)
- double **GA** (Population p, BufferedWriter bw)
- void **reduce** (Population Pop, Population newPop, int EliteSN)
- void **select** (Population Pop)
- int **selectParent** (Population Pop)

4.4.1 Detailed Description

Author

Junyu Lu

The documentation for this class was generated from the following file:

- D:/study/CS471/Lu_Project3/src/Project3/GeneticAlgorithm.java

4.5 Main Class Reference

Static Public Member Functions

- static void main (String[ ] args)

4.5.1 Detailed Description

Author

Junyu Lu

4.5.2 Member Function Documentation

4.5.2.1 main()

```
static void Main.main (
              String [] args )   [static]
```

Main method going into run()

Parameters

args	the command line arguments

The documentation for this class was generated from the following file:

- D:/study/CS471/Lu_Project3/src/Project3/Main.java

4.6 Matrix Class Reference

Public Member Functions

- Matrix ()
  create an empty matrixs with 0 rows and 0 columns
- Matrix (int columns, int rows)
  create an empty matrixs with given rows and columns
- void **resizeMatrix** (int columns, int rows)
- void **fillMatrix** (double low, double high)
- void **printMatrix** ()
- double [ ] **getArray** (int col)
- void **setArray** (int col, double[ ] x)

4.6.1 Detailed Description

Author

Junyu Lu

4.6.2 Constructor & Destructor Documentation

4.6.2.1 Matrix()

```
Matrix.Matrix ( )
```

create an empty matrixs with 0 rows and 0 columns

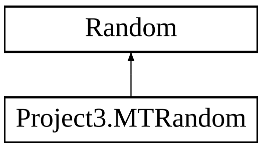
create an empty matrix

The documentation for this class was generated from the following file:

- D:/study/CS471/Lu_Project3/src/Project3/Matrix.java

4.7 Project3.MTRandom Class Reference

Inheritance diagram for Project3.MTRandom:



Public Member Functions

- MTRandom ()
- MTRandom (boolean compatible)
- MTRandom (long seed)
- MTRandom (byte[ ] buf)
- MTRandom (int[ ] buf)
- final synchronized void setSeed (long seed)
- final void setSeed (byte[ ] buf)
- final synchronized void setSeed (int[ ] buf)

Static Public Member Functions

- static int [ ] pack (byte[ ] buf)

Protected Member Functions

- final synchronized int next (int bits)

4.7.1 Detailed Description

Version

1.0

Author

David Beaumont, Copyright 2005

A Java implementation of the MT19937 (Mersenne Twister) pseudo random number generator algorithm based upon the original C code by Makoto Matsumoto and Takuji Nishimura (see `http://www.math.sci.hiroshima-u.ac.jp/~m-mat/MT/emt.html` for more information.

As a subclass of java.util.Random this class provides a single canonical method next() for generating bits in the pseudo random number sequence. Anyone using this class should invoke the public inherited methods (nextInt(), nextFloat etc.) to obtain values as normal. This class should provide a drop-in replacement for the standard implementation of java.util.Random with the additional advantage of having a far longer period and the ability to use a far larger seed value.

This is **not** a cryptographically strong source of randomness and should **not** be used for cryptographic systems or in any other situation where true random numbers are required.

This software is licensed under the `CC-GNU LGPL`.

4.7.2 Constructor & Destructor Documentation

4.7.2.1 MTRandom() [1/5]

```
Project3.MTRandom.MTRandom ( )
```

The default constructor for an instance of MTRandom. Since the no-argument constructor of java.util.Random does not seem to call setSeed anymore (since JDK7), we need to do it manually in this constructor. For legacy purposes, the seed remains initialized by a call to System.currentTimeMillis().

Author

Jonathan Passerat-Palmbach

4.7.2.2 MTRandom() [2/5]

```
Project3.MTRandom.MTRandom (
             boolean compatible )
```

This version of the constructor can be used to implement identical behaviour to the original C code version of this algorithm including exactly replicating the case where the seed value had not been set prior to calling genrand_int32.

If the compatibility flag is set to true, then the algorithm will be seeded with the same default value as was used in the original C code. Furthermore the setSeed() method, which must take a 64 bit long value, will be limited to using only the lower 32 bits of the seed to facilitate seamless migration of existing C code into Java where identical behaviour is required.

Whilst useful for ensuring backwards compatibility, it is advised that this feature not be used unless specifically required, due to the reduction in strength of the seed value.

Parameters

compatible	Compatibility flag for replicating original behaviour.

4.7.2.3 MTRandom() [3/5]

```
Project3.MTRandom.MTRandom (
             long seed )
```

This version of the constructor simply initialises the class with the given 64 bit seed value. For a better random number sequence this seed value should contain as much entropy as possible.

```
This constructor was modified due to be compliant to the JDK7's implementation
of java.util.Random as explained in MTRandom()
```

Parameters

seed	The seed value with which to initialise this class.

See also

MTRandom()

Author

Jonathan Passerat-Palmbach

4.7.2.4 MTRandom() [4/5]

```
Project3.MTRandom.MTRandom (
              byte [] buf )
```

This version of the constructor initialises the class with the given byte array. All the data will be used to initialise this instance.

Parameters

buf	The non-empty byte array of seed information.

Exceptions

NullPointerException	if the buffer is null.
IllegalArgumentException	if the buffer has zero length.

4.7.2.5 MTRandom() [5/5]

```
Project3.MTRandom.MTRandom (
              int [] buf )
```

This version of the constructor initialises the class with the given integer array. All the data will be used to initialise this instance.

Parameters

buf	The non-empty integer array of seed information.

Exceptions

NullPointerException	if the buffer is null.

Exceptions

IllegalArgumentException	if the buffer has zero length.

4.7.3 Member Function Documentation

4.7.3.1 next()

```
final synchronized int Project3.MTRandom.next (
            int bits )  [protected]
```

This method forms the basis for generating a pseudo random number sequence from this class. If given a value of 32, this method behaves identically to the genrand_int32 function in the original C code and ensures that using the standard nextInt() function (inherited from Random) we are able to replicate behaviour exactly.

Note that where the number of bits requested is not equal to 32 then bits will simply be masked out from the top of the returned integer value. That is to say that:

```
mt.setSeed(12345);
int foo = mt.nextInt(16) + (mt.nextInt(16) << 16);
```

will not give the same result as

```
mt.setSeed(12345);
int foo = mt.nextInt(32);
```

Parameters

bits	The number of significant bits desired in the output.

Returns

The next value in the pseudo random sequence with the specified number of bits in the lower part of the integer.

4.7.3.2 pack()

```
static int [] Project3.MTRandom.pack (
            byte [] buf )  [static]
```

This simply utility method can be used in cases where a byte array of seed data is to be used to repeatedly re-seed the random number sequence. By packing the byte array into an integer array first, using this method, and then invoking setSeed() with that; it removes the need to re-pack the byte array each time setSeed() is called.

If the length of the byte array is not a multiple of 4 then it is implicitly padded with zeros as necessary. For example:

```
byte[] { 0x01, 0x02, 0x03, 0x04, 0x05, 0x06 }
```

becomes

```
int[]  { 0x04030201, 0x00000605 }
```

Note that this method will not complain if the given byte array is empty and will produce an empty integer array, but the setSeed() method will throw an exception if the empty integer array is passed to it.

Parameters

buf	The non-null byte array to be packed.

Returns

A non-null integer array of the packed bytes.

Exceptions

NullPointerException	if the given byte array is null.

4.7.3.3 setSeed() [1/3]

```
final synchronized void Project3.MTRandom.setSeed (
            long seed )
```

This method resets the state of this instance using the 64 bits of seed data provided. Note that if the same seed data is passed to two different instances of MTRandom (both of which share the same compatibility state) then the sequence of numbers generated by both instances will be identical.

If this instance was initialised in 'compatibility' mode then this method will only use the lower 32 bits of any seed value passed in and will match the behaviour of the original C code exactly with respect to state initialisation.

Parameters

seed	The 64 bit value used to initialise the random number generator state.

4.7.3.4 setSeed() [2/3]

```
final void Project3.MTRandom.setSeed (
            byte [] buf )
```

This method resets the state of this instance using the byte array of seed data provided. Note that calling this method is equivalent to calling "setSeed(pack(buf))" and in particular will result in a new integer array being generated during the call. If you wish to retain this seed data to allow the pseudo random sequence to be restarted then it would be more efficient to use the "pack()" method to convert it into an integer array first and then use that to re-seed the instance. The behaviour of the class will be the same in both cases but it will be more efficient.

Parameters

buf	The non-empty byte array of seed information.

Exceptions

NullPointerException	if the buffer is null.
IllegalArgumentException	if the buffer has zero length.

4.7.3.5 setSeed() [3/3]

```
final synchronized void Project3.MTRandom.setSeed (
            int [] buf )
```

This method resets the state of this instance using the integer array of seed data provided. This is the canonical way of resetting the pseudo random number sequence.

Parameters

buf	The non-empty integer array of seed information.

Exceptions

NullPointerException	if the buffer is null.
IllegalArgumentException	if the buffer has zero length.

The documentation for this class was generated from the following file:

- D:/study/CS471/Lu_Project3/src/Project3/MTRandom.java

4.8 Population Class Reference

Public Member Functions

- **Population** (BufferedReader br) throws IOException
- **Population** (int s, int d)
- void **randomInit** (double boundLow, double boundHigh)
- void **evaluate** (int fType)
- void **copy** (Population p)
- double [ ] **getVector** (int col)
- void **setVector** (int col, double[ ] in)
- void sortByCostAscending ()

4.8.1 Detailed Description

Author

Junyu Lu

4.8.2 Member Function Documentation

4.8.2.1 sortByCostAscending()

```
void Population.sortByCostAscending ( )
```

Use hand writing selection sort to sort the cost and the matrix

The documentation for this class was generated from the following file:

- D:/study/CS471/Lu_Project3/src/Project3/Population.java

4.9 Support Class Reference

Public Member Functions

- Support () throws IOException

  consturctor

Static Public Member Functions

- static double scale (double in, double low, double high)

  This method is a support for MT, input a random double from 0 ot 1, return the correspoding random value from low to high.

4.9.1 Detailed Description

Author

Junyu Lu

The documentation for this class was generated from the following file:

- D:/study/CS471/Lu_Project3/src/Project3/Support.java

Chapter 5

File Documentation

5.1 D:/study/CS471/Lu_Project3/src/Project3/FunctionClass.java File Reference

a file for functions

Classes

- class FunctionClass
- enum FunctionClass.Functions

5.1.1 Detailed Description

a file for functions

Author

Junyu Lu

Date

2019/4/21

Version

1

5.2 D:/study/CS471/Lu_Project3/src/Project3/Main.java File Reference

main file of CS471 project2

Classes

- class Main

5.2.1 Detailed Description

main file of CS471 project2

Author

Junyu Lu

Date

2019/4/21

Version

1

5.3 D:/study/CS471/Lu_Project3/src/Project3/Matrix.java File Reference

a file for matrix modification

Classes

- class Matrix

5.3.1 Detailed Description

a file for matrix modification

Author

Junyu Lu

Date

2019/4/21

Version

3

5.4 D:/study/CS471/Lu_Project3/src/Project3/Support.java File Reference

a file for support tool for project2

Classes

- class Support

5.4.1 Detailed Description

a file for support tool for project2

Author

Junyu Lu

Date

2019/4/21

Version

1