APPLICATION 2: Symmetric Key Cryptography

Initial Comments

With this application, the user is able to generate cipher texts used for symmetric key encryption. This is based off of the ICIGA (Improved Cryptography Inspired by Genetic Algorithms) block cipher system. The purpose is to create cipher texts while randomly generating keys to determine more secure protocols.

Essentially, the user provides three things: key length, block size, and plaintext and the output is a ciphertext.

Mutable object: ciphertext

CipherEvolverApp

* The client of the TanzaniteEngine Framework
* A Facade to the underlying picture creation/representation framework

CipherComponent (abstract) **These are blocks**

* Implements Mutatable
  + able to build itself randomly, mutate, swap subtrees (crossover), etc
* Utilizes Composite Pattern to construct itself

CipherLeaf (abstract)

* Represents different abstract ways to construct a block
  + Hexadecimal, Octal, Decimal, Binary, etc.

CipherComposite

* Composite pattern realization, holds up to N CipherComponents

CipherDecorator (abstract)

* Capable of applying a certain encryption to modify the existing text string.
  + AES, DES3, SHA-256

Settings (abstract)

* Holds information about how to decorate a subtree
* This exact information is beyond scope of application design (very complex), but will approximately represented using a byte array
* Utilizes Bridge Pattern to separate application of settings from the Decorator
* Settings are applied to CipherComponents (**Blocks**)

CipherComponentFactory (abstract) (**Blocks**)

* Used to create CipherComponents (**Blocks)**
* Allows for random key generation, useful for evolution

CipherDecrypter (abstract)

* Clients will not be able to read blocks (string of bits). Converting the bits and returning a common String type is necessary via exporting

CipherGatherer (abstract)

* Initial populations for evolution can be selected using various kinds of CipherGatherers

**Subclasses (self-explanatory)**

CipherLeaf (abstract)

* BinaryLeaf (abstract)
* OctalLeaf (abstract)
* HexadecimalLeaf (abstract)
* etc.

CipherDecorator (abstract)

* AESDecorator
* DES3Decorator
* SHA256Decorator
* MD5Decorator
* etc.

Settings (abstract)

* AESSettings
* DES3Settings
* SHA256Settings
* MD5Settings
* etc.

Usage

Fitness Testing function:

F = n + (€ /m)

Where F = Fitness Function.

n = Total number of symbols used in key formation.

m = Percentage of maximum appeared symbol.

€ = Ideal Percentage of each symbol.