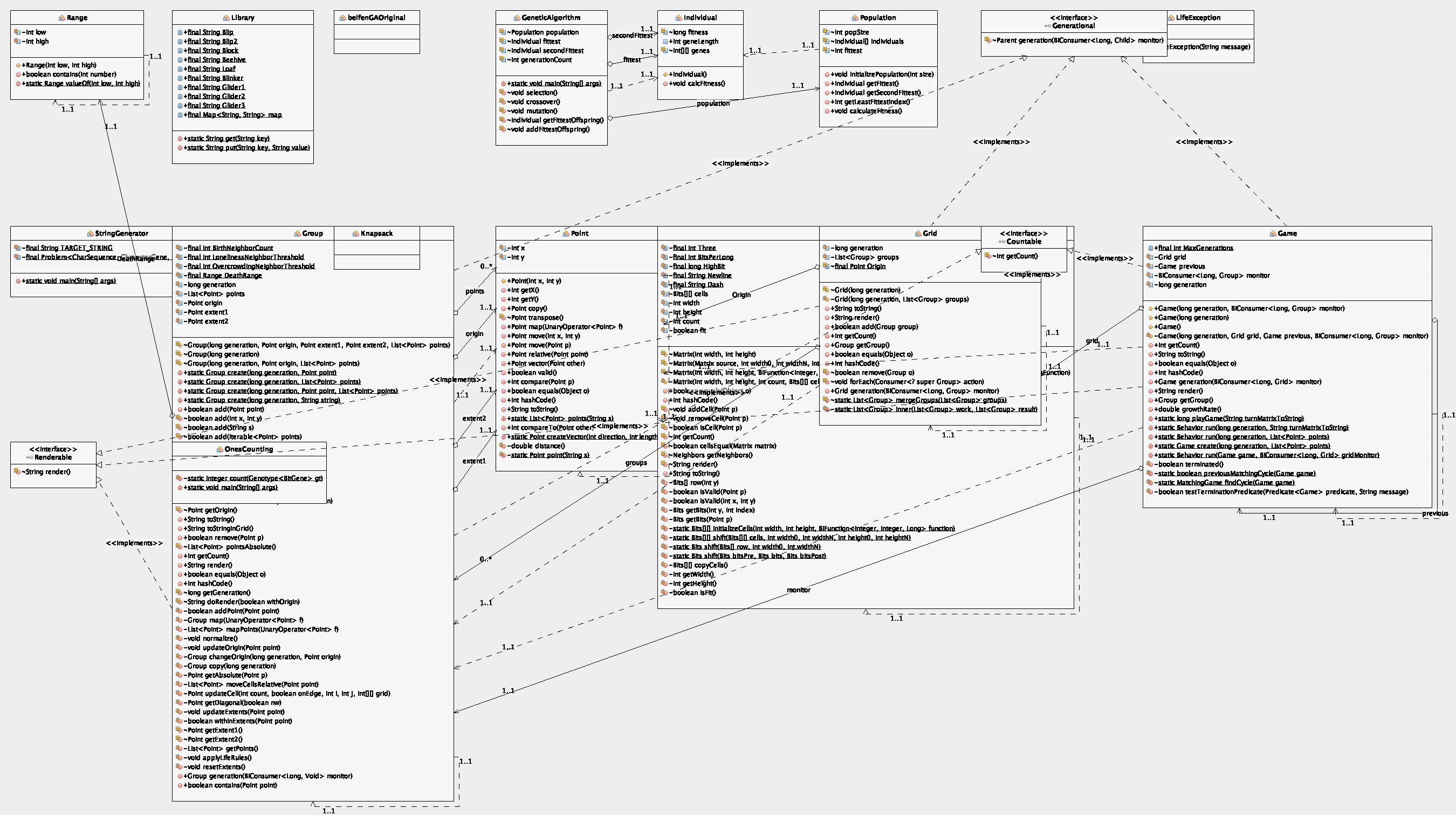
**Project Structure**

Class graph

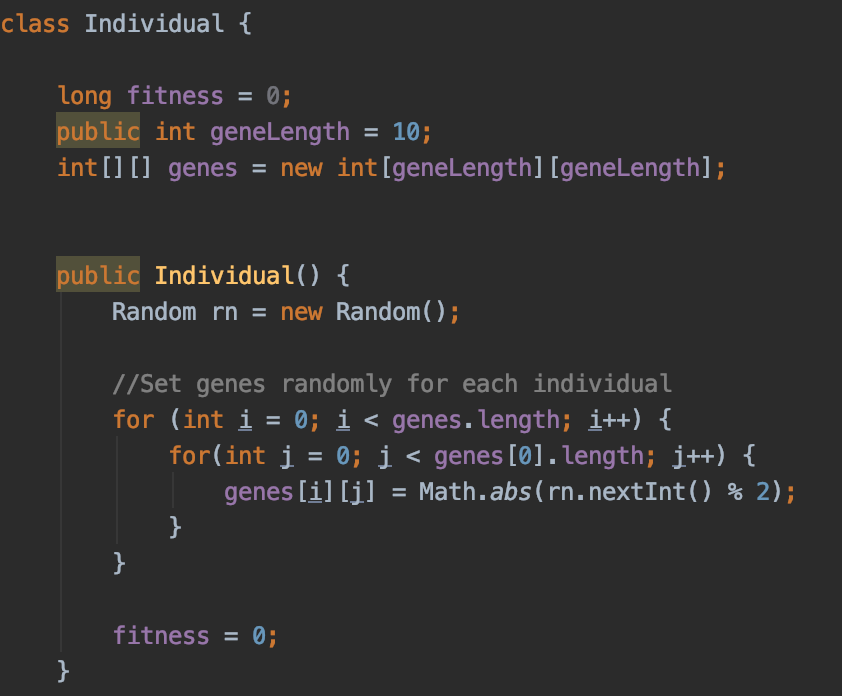


All Generic Algorithms are in the “GeneticAlgorithm” class under “base” package. It does not use extra libraries, the whole algorithm was written by my own in this single class document.

**Algorithm Declaration**

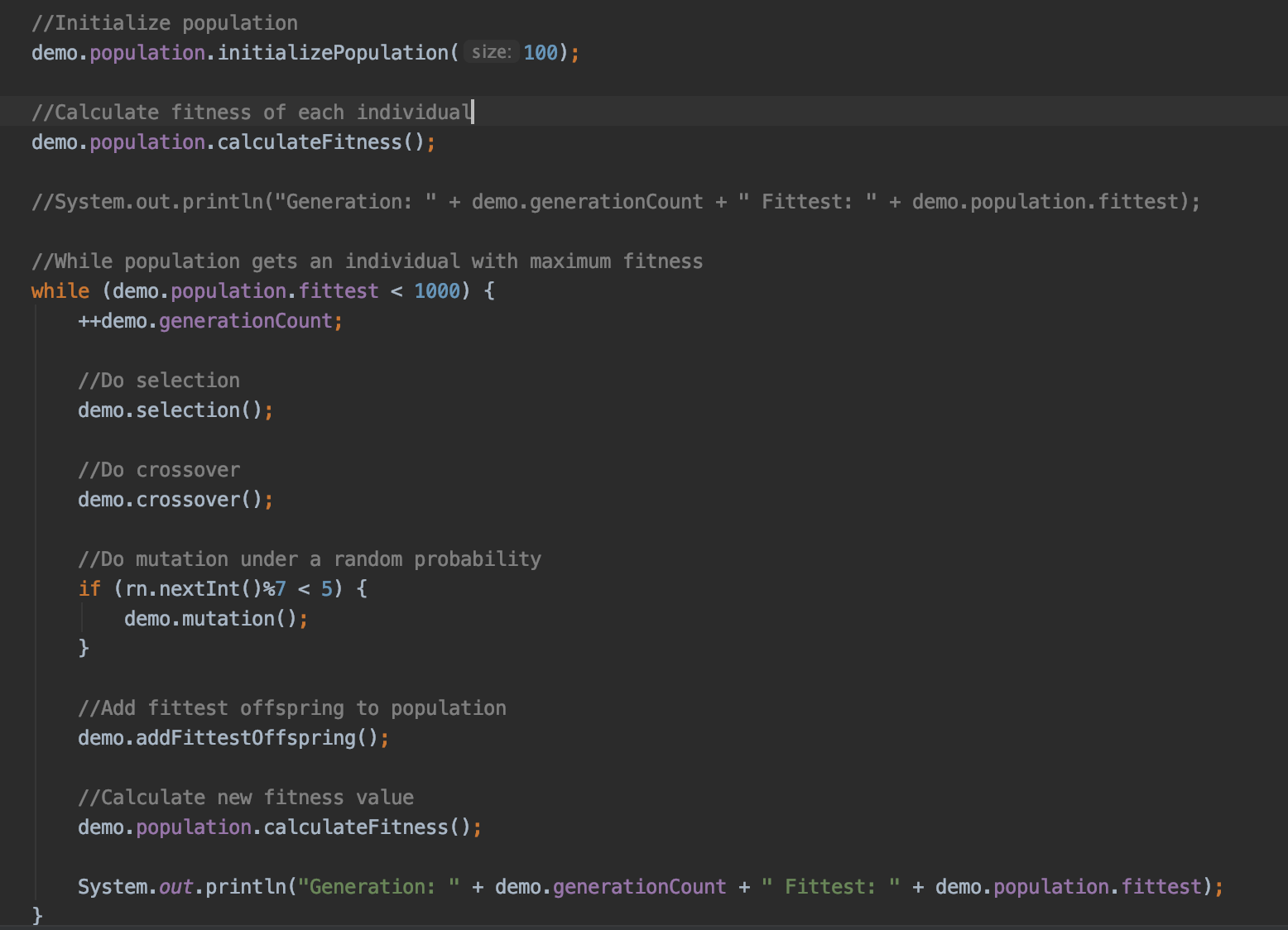
I randomly produced starting individuals(stored the map of life game by using matrix), the population(number of individuals) of 1 generation is set to be 100. But it can be fixed at any time if needed. We do things like ” natural selection” to find a possible “best” solution of a given problem. In this project, the problem is Conway’s Life Game.

A. The structure of a individual



B.The work steps of this algorithm is shown as below:

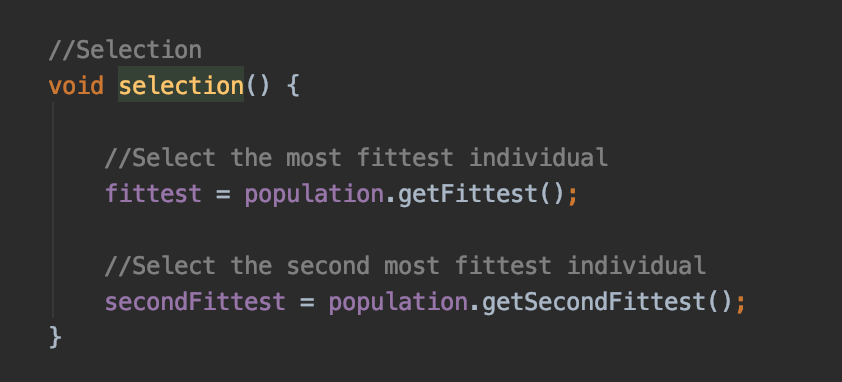
1/ Initialize the population of each generation and calculate fitness(picture of this function showed below) by putting the genes matrix of each individual int the life game process.



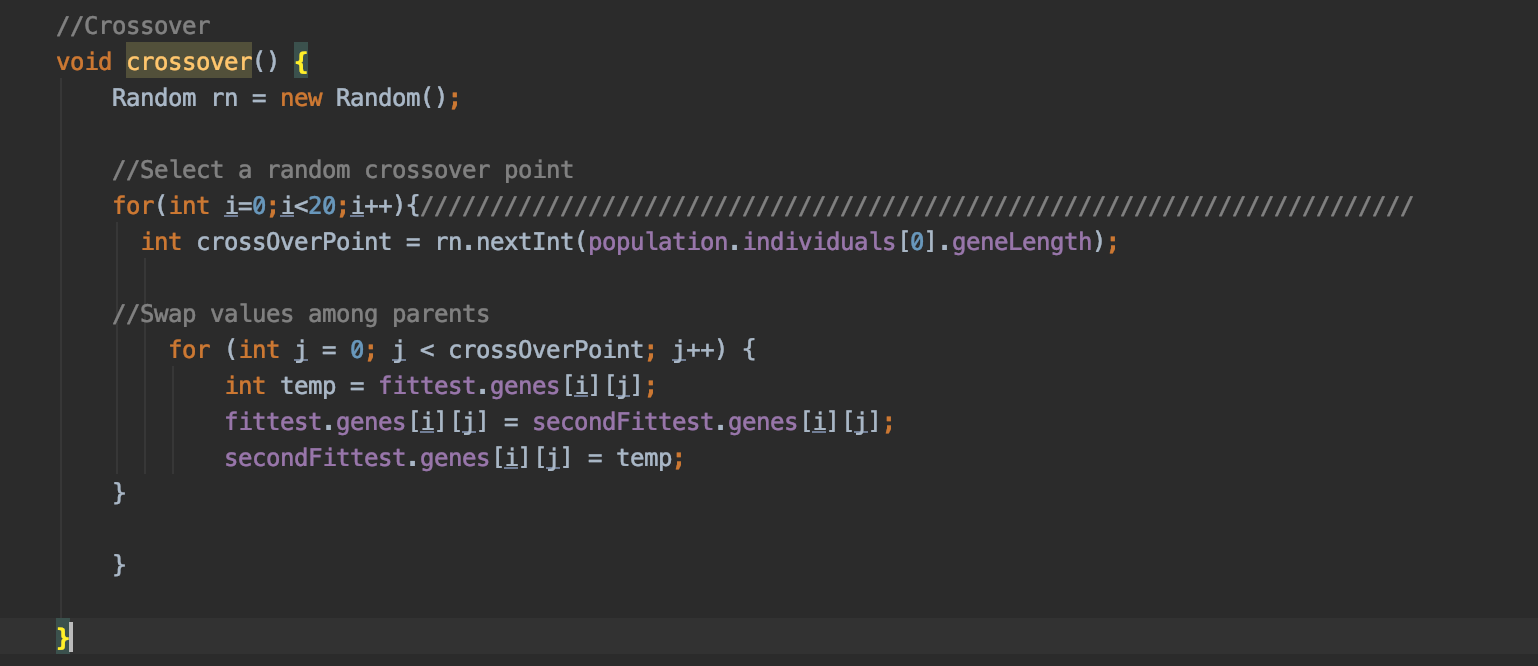
Fitness function:



2/ If we do not get the genes fit enough(grow 1000 steps without extinction), select the fittest two individuals in this generation.



3/Do crossover by using the fittest two individuals’ genes in this generation.



4/ Do mutation and keep going in next generation.





**Experiments of running**

A. (Group number:100 geneLength:10 maxGenerationLimit:1000)

Experiment1 result

Fitness: 1000

Genes:

0110110011

0001001010

0111110010

0010000100

0001010111

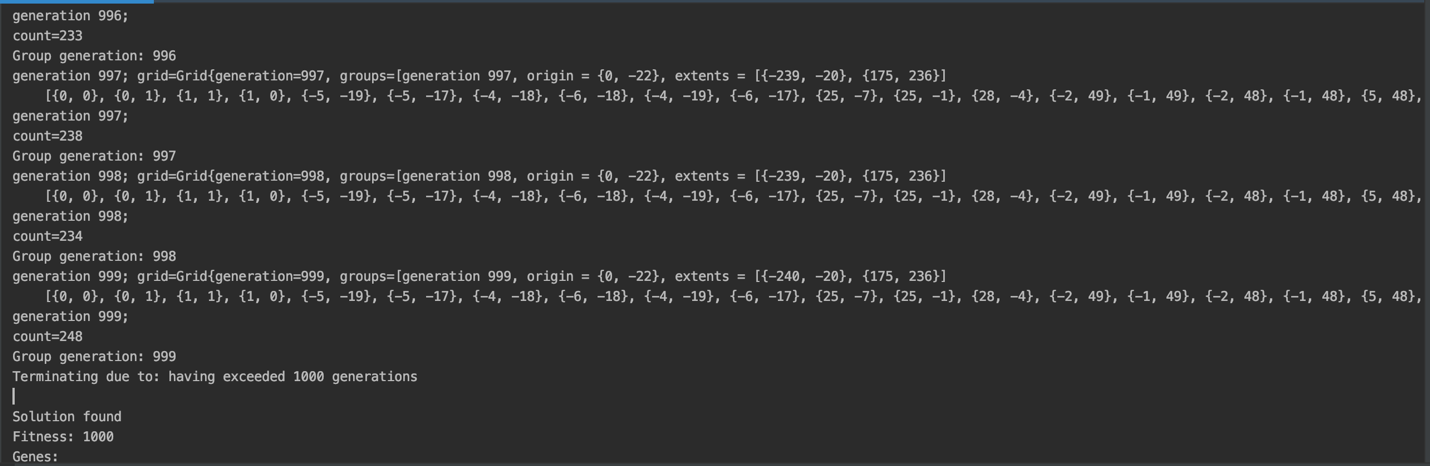
0111111011

0011011011

0001000101

1000010000

1110010010



Experiment2 result

Fitness: 1000

Genes:

0001110001

1110001111

1110001101

0001000100

0110010010

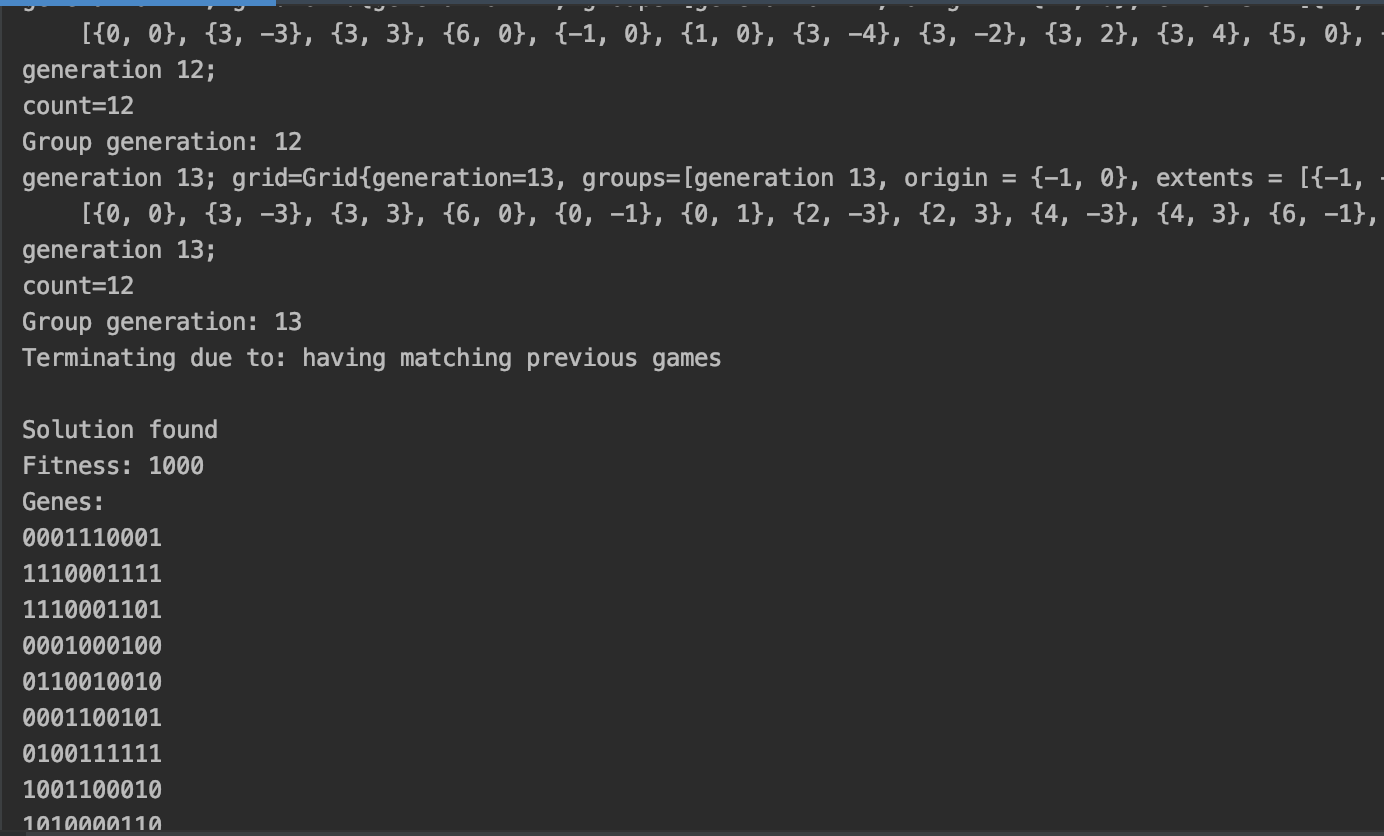
0001100101

0100111111

1001100010

1010000110

0011111000



B. (Group number:100 geneLength:20 maxGenerationLimit:1000)

Experiment1 result

Fitness: 1000

Genes:

11011100010100110111

10101100011110110011

01000001111101100001

01010001110001110101

11101010000111001110

11010111110001011010

11010111111000111100

00111101111000001110

00001110100001011011

10000111111110000111

10101010000010001110

00111001000000011101

10100111010010110000

00010011011010011100

01101001101000110100

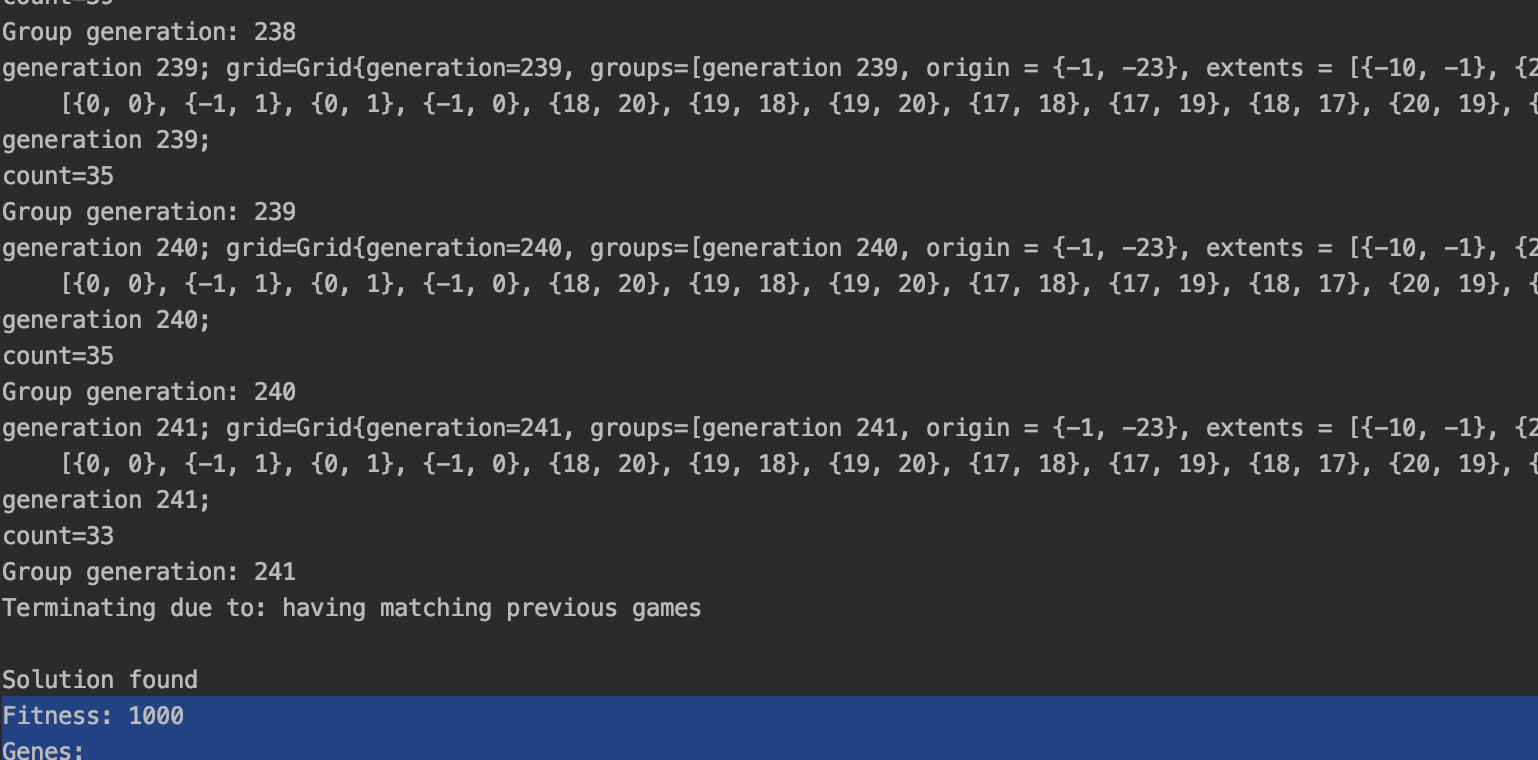
00010100111011110111

10101111111110010101

11010100110000010101

11100111001110000100

00100010111111000111



Experiment2 result

Fitness: 1000

Genes:

11101110111001101101

11100110101011000000

01100111010101111001

11000001110100010010

01100011011000000100

01010001100110101010

10111111110001100101

11111110000101100000

10001100110101110011

10110101000100001111

11100100010100101000

01011100011100111011

11011111110011111110

00000000110000000101

00000111010010101100

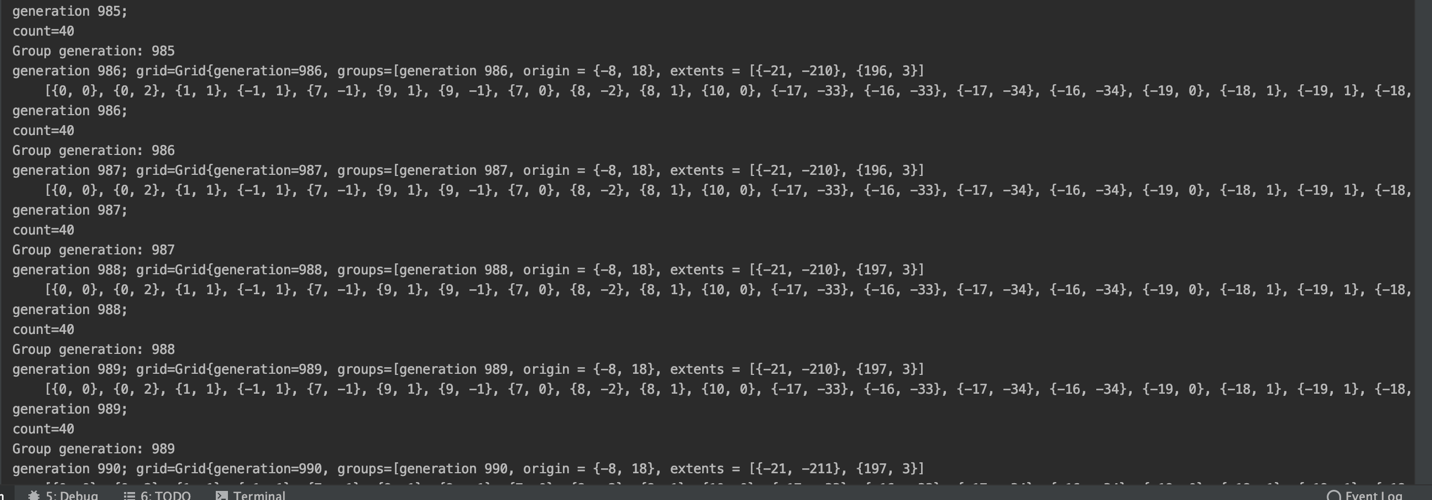
00111010101001111100

10001101100001110101

01110111111100111100

01010011100000101010

00001100100010010011



Unit Test Screen Shots

I failed to pass some of the unit test, there was even some compile errors. But in my point of view, the few test failures have nothing to do with my programming. Because somewhat the compile errors are all related to the teacher’s life game source code.

For example: there is no such function can be used in Group.create( generation, point, points)—only Group.create( generation, point)or Group.create( generation, points) are presented in the source code of professor’s life game. I think there might be something going wrong between the test code and the main code.

