HAMZA YOĞURTCUOĞLU

OBJECT ORIENTED ANALYSIS AND DESIGN
171044086 - HW2

Why Template Method Design Pattern?

Template method design pattern is to define an algorithm as skeleton of operations and leave the details to be implemented by the child classes. The overall structure and sequence of the algorithm is preserved by the parent class.
In template method pattern,we have a present structure method called template method which consists of steps. This steps can be abstract method which will be implemented by its subclasses. This behavioral design pattern is one of the easiest to understand and implement. This design pattern is used popularly in framework development. This helps to avoid code duplication also.
For example: We have 4 repeating function ; Selection, Crossover, Mutation, Compute fitness.

How did I implement them?

1)Selection : We have to implement 3 versions of this algorithm which are roulette wheel selection,rank selection and tournament selection. So we can abstract function in abstract base class in order to implement concrete classes.

Version-1 (roulette wheel selection) : Each individual has fitness. I thought that each individual takes slice from cake. Then, select a random number (0 to Sum of the fitnesses). If the I have chosen random as the parent of which individual's fitnes. Then second parent like selected first parent.

Version-2 (rank selection): first ranks the population and then every chromosome receives fitness from this ranking. The worst will have fitness 1, second worst 2 etc. Then again, taking a random number (1 to sum of total rank). I take parent which if random number is in interval of individual.

Version-3 (tournament selection): I create 2 tournament. Then I selected 2 individuals in each tournament. I take like parent which is fittest.

2)Crossover : Each version has different of crossover method. That can be 2 or 1. So Subclass can implement it.

1 point crossover : Parent1 and Parent2 are swapping x1 chromosomes for create a child.

2 point crossover : Parent1 and Parent2 are swapping x1 and x2 chromosomes for create a child.

3)Mutation : Our algorithm don't use different mutation method. They are all same so this method that will be final method in abstract class. So, Subclass can use it.

Example : I take random number for x1 and x2. But I paid attention for following constrained ;

$x1 + x2 \leqslant 5$; $0 \leqslant x1 \leqslant 5$; $0 \leqslant x2 \leqslant 5$

4)Compute fitness : Each individuals has x1 and x2. So just I put the fitness function;

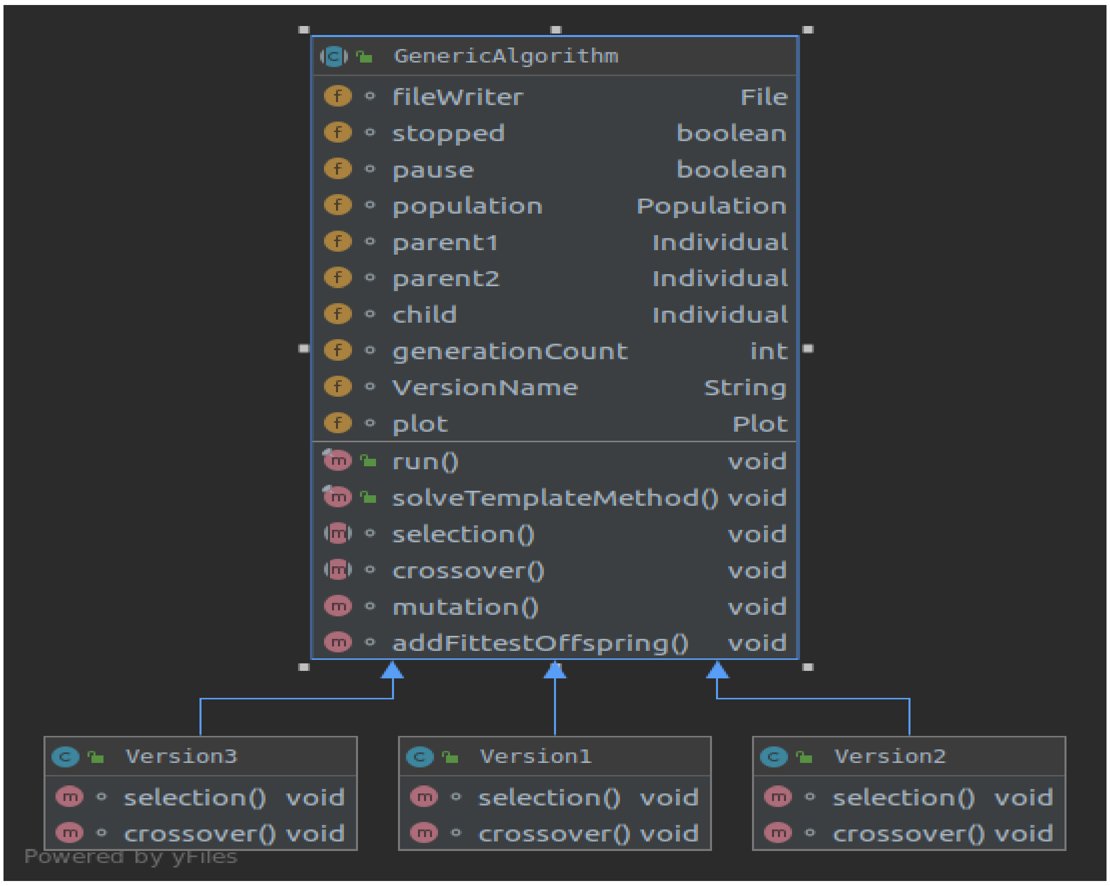
$$f(x_1, x_2) = 20x_1x_2 + 16x_2 - 2x_1^2 - x_2^2 - (x_1 + x_2)^2$$

Template Design Pattern - Class Diagram:

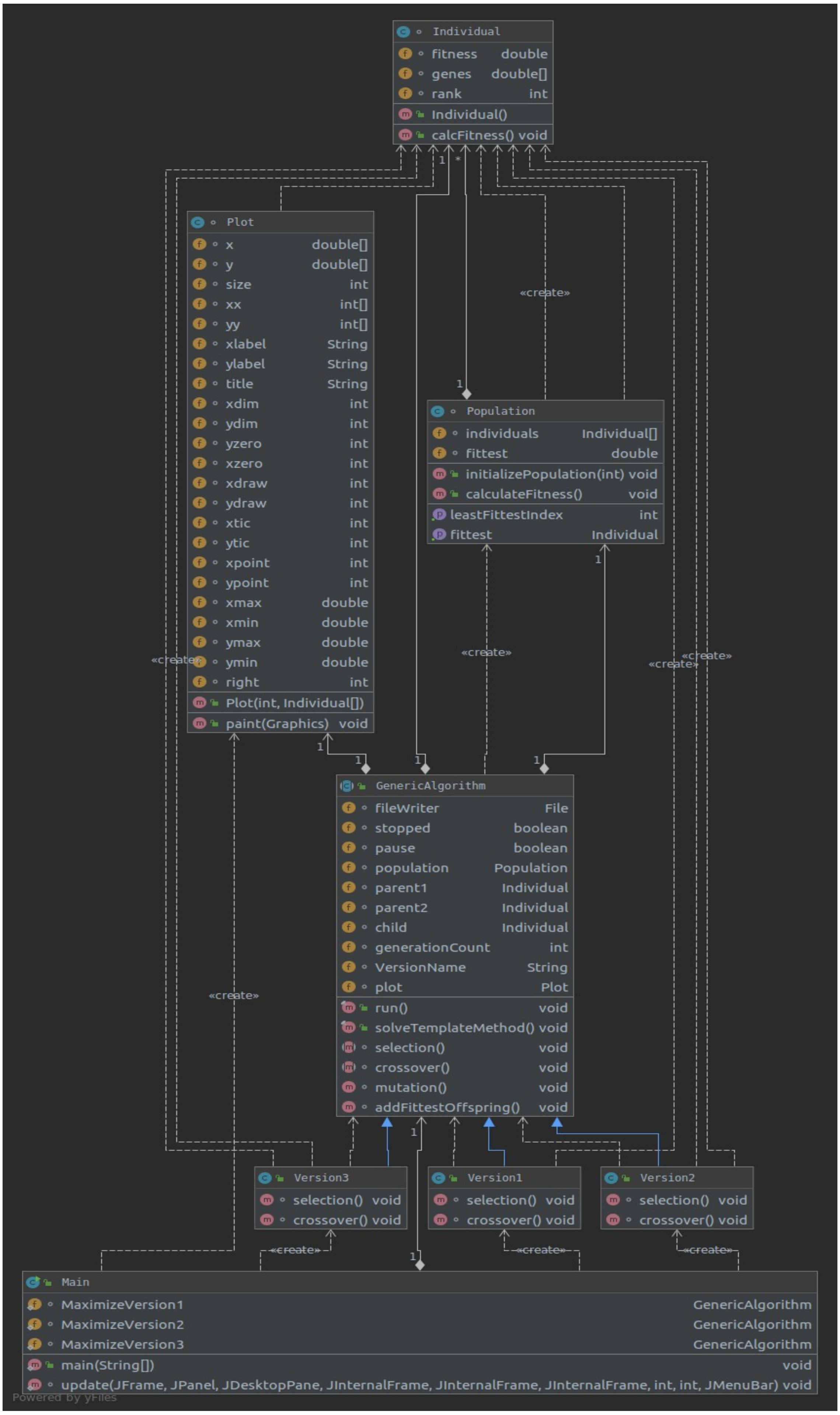
Abstract class (GenericAlgorithm)

final method : solveTemplateMethod , mutation, calculateFitness

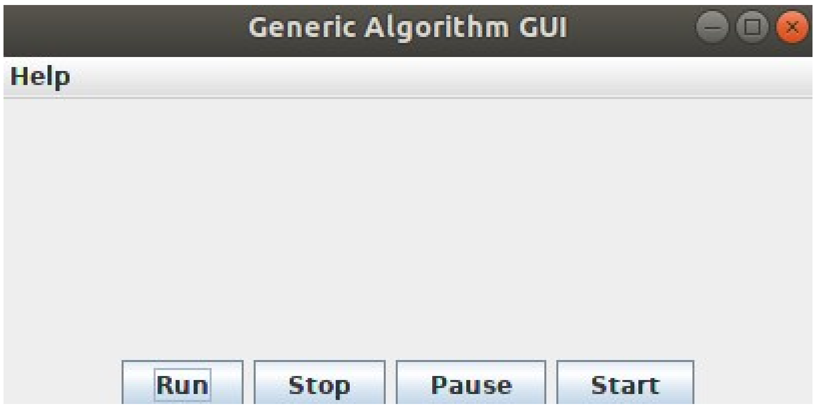
overridden method : Selection and crossover



Project Total Class Diagram :

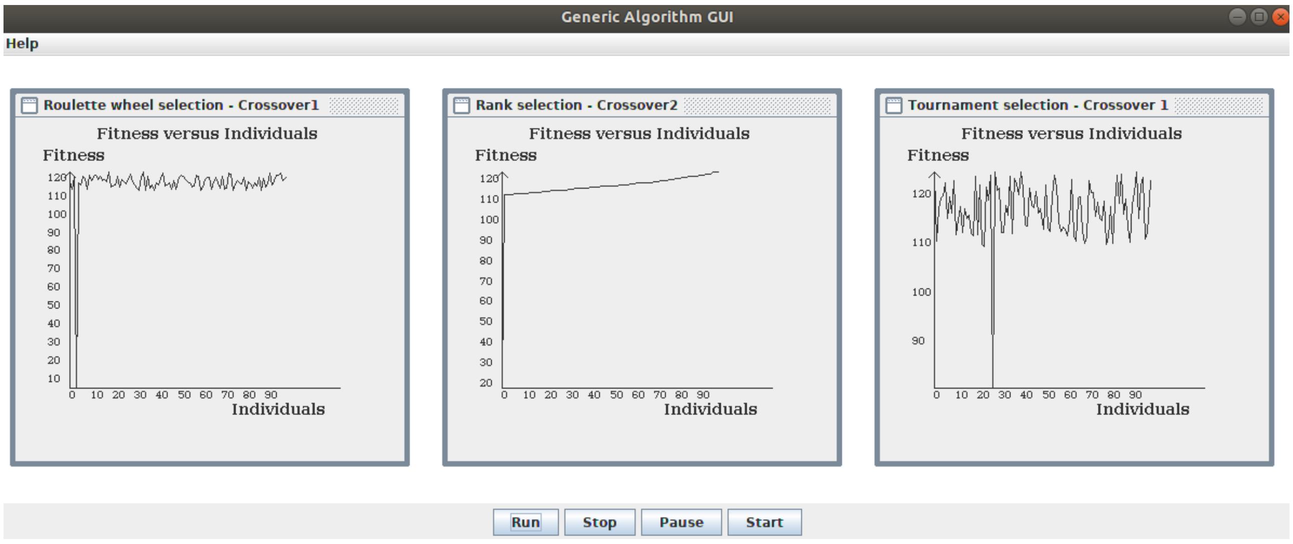


Graphical User Interface : I create 3 threads in order to run 3 versions.
Note: You can see that is last finished version in **LastFinishedOutput.txt**



Note: You can check Help-button to understand what run,stop,pause and start button are.

When you press start button. 3 version of algorithms are started.


Generic Algorithm GUI
Help
Roulette wheel selection - Crossover1
Fitness versus Individuals
Fitness
Individuals
Rank selection - Crossover2
Fitness versus Individuals
Fitness
Individuals
Tournament selection - Crossover 1
Fitness versus Individuals
Fitness
Individuals
Run
Stop
Pause
Start

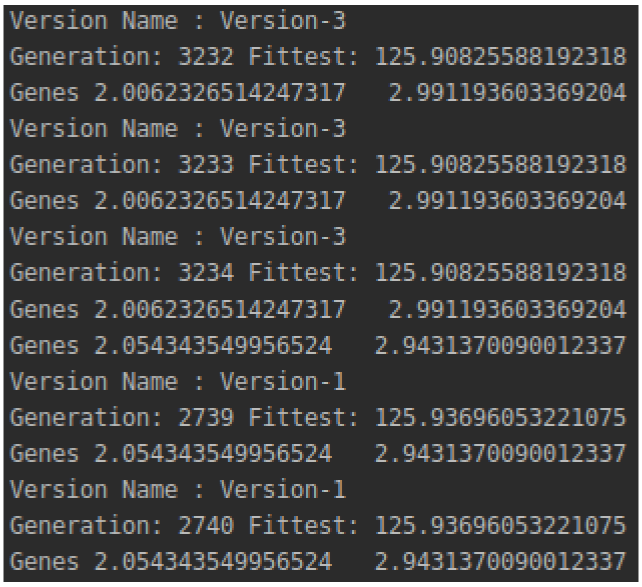
You will see three 2D-plots. When they are finished until population has converged (126).

If any version is not 126 fittest. You can stop or pause. If you paused the running algorithms. You can resume them.

You can check on the console.

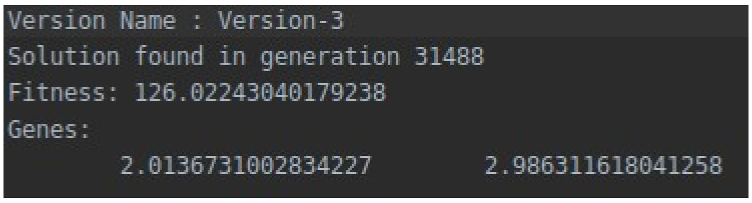
Note: x_1 x_2

Genes 2.05… 2.94…



```
Version Name : Version-3
Generation: 3232 Fittest: 125.90825588192318
Genes 2.0062326514247317   2.991193603369204
Version Name : Version-3
Generation: 3233 Fittest: 125.90825588192318
Genes 2.0062326514247317   2.991193603369204
Version Name : Version-3
Generation: 3234 Fittest: 125.90825588192318
Genes 2.0062326514247317   2.991193603369204
Genes 2.054343549956524   2.9431370090012337
Version Name : Version-1
Generation: 2739 Fittest: 125.93696053221075
Genes 2.054343549956524   2.9431370090012337
Version Name : Version-1
Generation: 2740 Fittest: 125.93696053221075
Genes 2.054343549956524   2.9431370090012337
```

Last finished version see on **LastFinishedOutput.txt :**



```
Version Name : Version-3
Solution found in generation 31488
Fitness: 126.02243040179238
Genes:
        2.0136731002834227        2.986311618041258
```

Generally Last finished version is version-2