**INFO6205: Program Structure and Algorithms Section 5**

**Title: Super Gene Mapper**

**By**

**Sai Swagath (001417567)**

**Goutham Reddy(001449422)**

**Ajay Mohandas(001426741)**

**Git Hub URL:** <https://github.com/saiswagath/INFO6250-GeneticAlgorithm-Team521>

Contents

[Introduction 2](#_Toc6573698)

[Problem Statement 2](#_Toc6573699)

[Summary 2](#_Toc6573700)

[Algorithm 3](#_Toc6573701)

[Extra Features 4](#_Toc6573702)

[Flowchart 4](#_Toc6573703)

[Screenshot 4](#_Toc6573704)

[Merits of our application 6](#_Toc6573705)

[Future Scope 6](#_Toc6573706)

[References 6](#_Toc6573707)

# Introduction

Over the last decades, molecular cloning has transformed biological sciences. The field has attracted many scientist and implementing the same in numerous fields. The use of recombinant DNA, genome therapy, gene editing has successfully started to enter the field of cellular engineering. Due to the emergence of technology and research done we are able to find cure for few of the diseases but it takes years for any ground-breaking result to develop by a human. What if humans had help from computers who would analyse and provide us the solutions to finding cure. What if we build an algorithm that when given boundary values and restriction will calculate and predict the most immune cell from millions, which could be multiplied in order to develop the arsenal to fight against common diseases permanently. Our application is one such sample on how we can apply genetic algorithm to find cure for any disease out of million possible blood samples.

# Problem Statement

Find a human genome from a series of population having the super gene to fight against cancer

# Summary

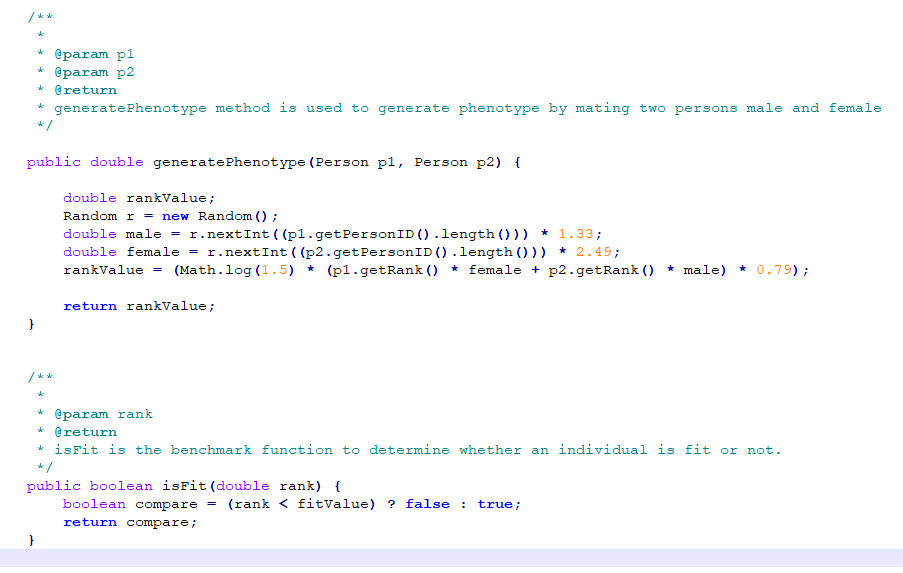
Genetic Algorithm has three main components such as genotype, phenotype and fitness value. There are two ways where a genetic algorithm stops - one when we find a value equivalent to fitness value provided in fitness function, two when we forcefully stop the algorithm.

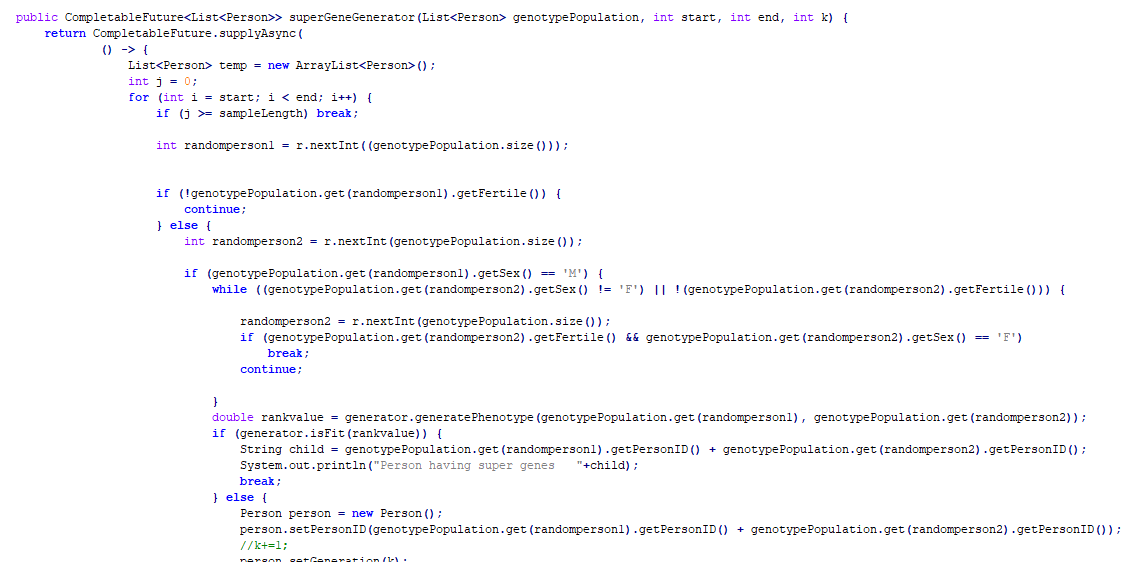
Our application applies the above algorithm to find the super genes present in a person’s blood by comparing the person’s generated fitness value to a threshold value which is required to cure a disease.

Following are the factors considered in the project:-

1. A “Person” class has attributes like personID, sex, fertile, rank and details to which generation the person belongs to
2. Person Id will help is find the trace the generation path of a person and give us details as to from which two parent the current generation child was born right from original population
3. Our application has two conditions for two persons gene’s to be mated that is two person should be of opposite sex and both the person has to be fertile to produce next generation
4. If any of the condition does not satisfy then that person is not considered for mating
5. **Genotype** is initial series of population that population we will be generating having rank value set based on the formula
6. **Phenotype** is a function applied on genotype to produce a rank value
7. **Fitness function**  helps us find the person having super gene above the threshold limit
8. **CompletableFuture** helps us in dividing huge population into half and process the superGeneGenerator method in parallel and check if we find super genes in first half or second half of the population
9. **SuperGeneGenerator** method taken parameters like the person’s list, start value, end value and generation number. It check if there’s a person having gene’s rank value greater than fitness value. If there are no one present, then a second series of population is generated which is stored in arraylist. People generated in this class is now compared with the fitness value and if no matched then next generation is formed.
10. This goes on till 10th generation of our original population or till we find a person having super gene

Below are the few screenshots of our calculations:





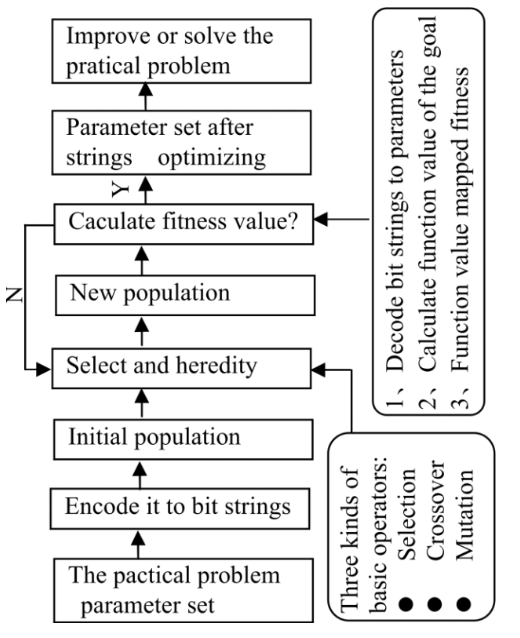
# Algorithm

1. Generate a series of random population and set its attributes as mentioned in the class Person
2. Compute phenotype from these samples and compare with the fitness value
4. If rank computed falls in range of fitness value then algorithm stops
5. Else pick two **fertile** **opposite** **sex** sample and combine the genes to create their child
6. Store new cell with key by appending person ID from two samples mated to know the hierarchy of child
7. Repeat step 2, 3 and 4 till the rank computed falls in range of fitness value
8. Stop if we reach 10th generation of original population

# Extra Features

1. Used “**CompletableFuture**” and “**ForJoinPool**” for parallel processing, so that our comparison can be done much faster than normal way
2. Our population list is divided into half and processed together to find super human genome
3. The total number of comparison done for each set of population is 0.5\* N \* Log(N), an equation we came up from Union-Find assignment
4. We also tested our application with total 8 test cases, thus following a test driven development

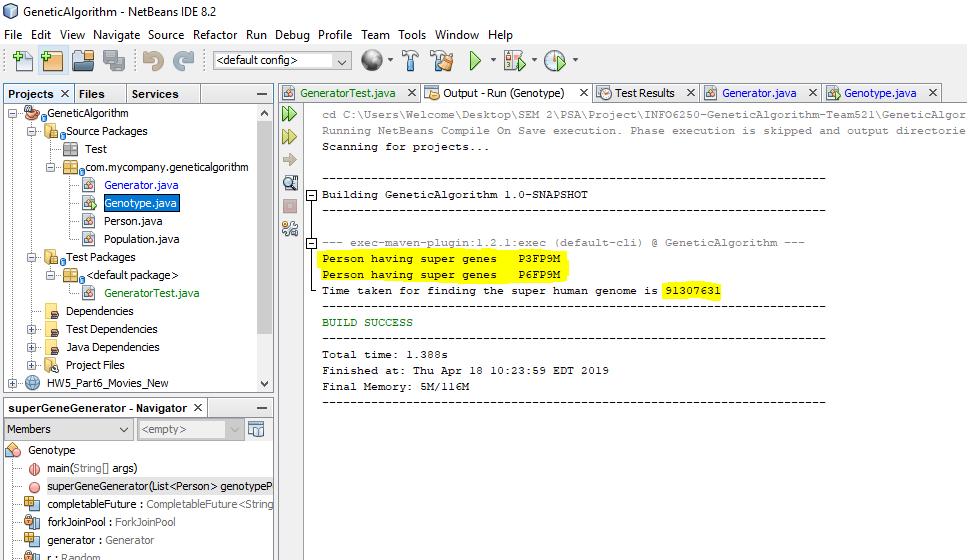
# Flowchart



# Screenshot

Main class

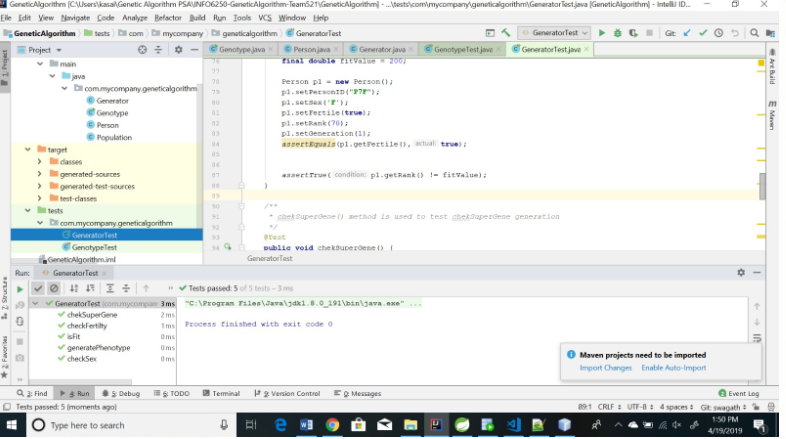
1. When we find a person having super gene

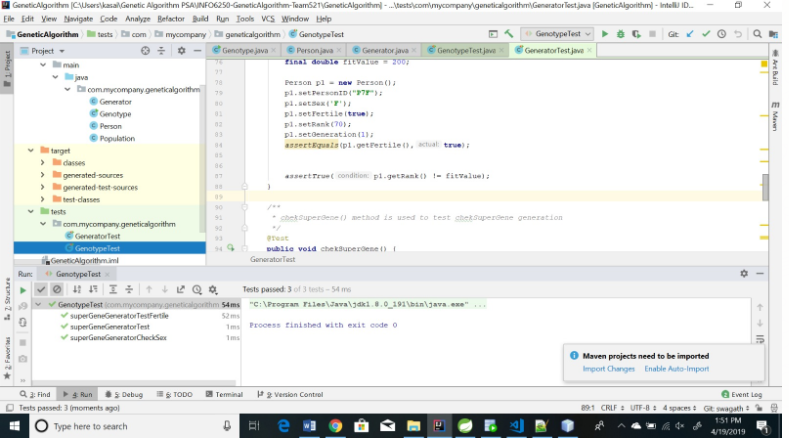


1. When the population does not have a super gene



Test File





# Merits of our application

1. Have used List interface as much as possible so that in future if we want to change the data type of Person then we can do it easily
2. The number of comparison done is equivalent to 0.5\*N\*Log(N)
3. Application helps us find genes having the power to fight against a disease

# Future Scope

Scope of the application could be widened to find super genes for any type of disease once we get a threshold limit required to kill the virus or resist them

# References

1. <http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.93.3979&rep=rep1&type=pdf>
2. <https://www.mathworks.com/discovery/genetic-algorithm.html>
3. <https://www.mathworks.com/help/gads/genetic-algorithm.html>