Gebze Technical University

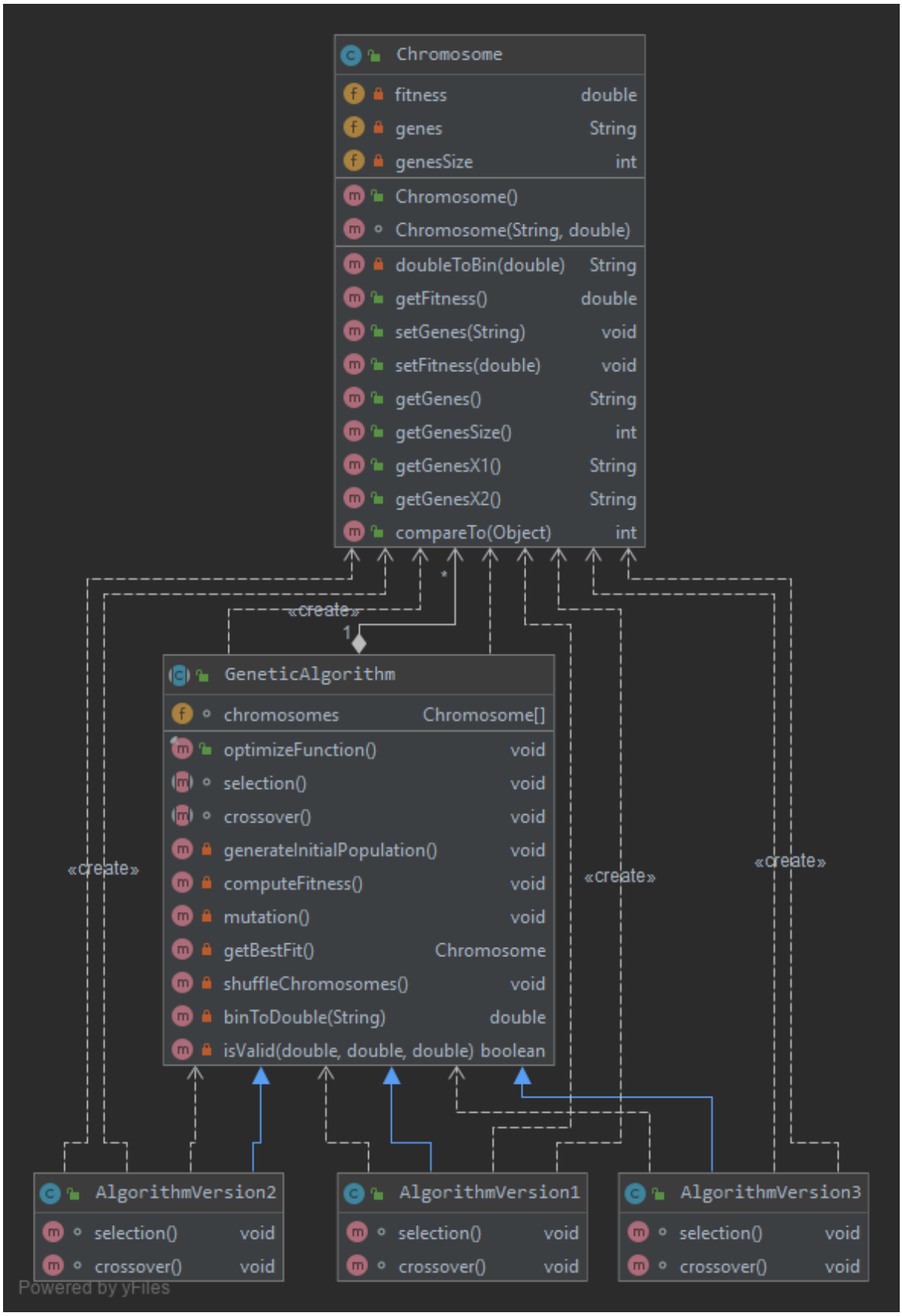
Computer Engineering

Object Oriented Analysis and Design
CSE443 – 2019

HOMEWORK 2 REPORT

Serkan Sorman
151044057

Q1)

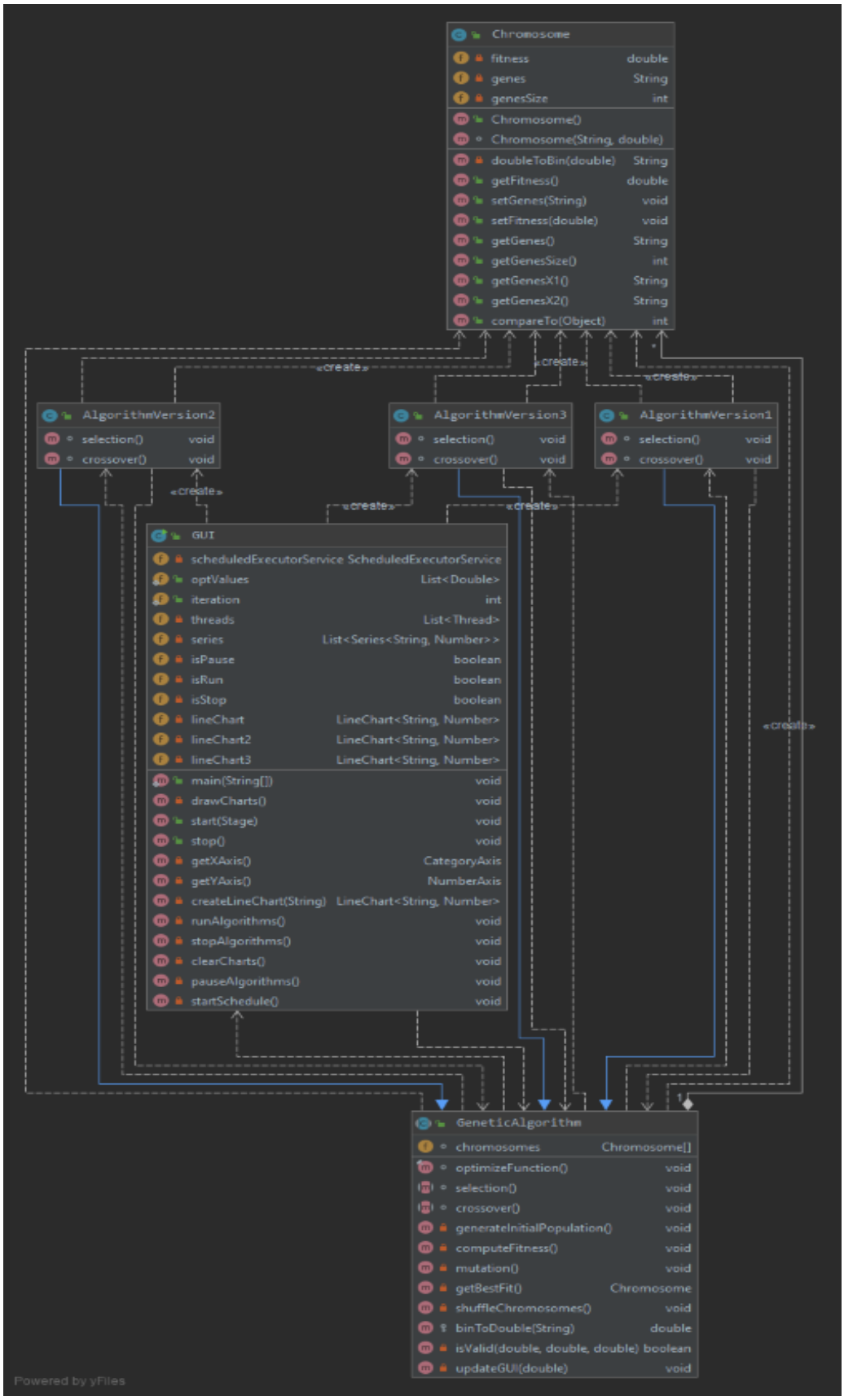

Chromosome
fitness double
genes String
genesSize int
Chromosome()
Chromosome(String, double)
doubleToBin(double) String
getFitness() double
setGenes(String) void
setFitness(double) void
getGenes() String
getGenesSize() int
getGenesX1() String
getGenesX2() String
compareTo(Object) int
«create»
1
*
GeneticAlgorithm
chromosomes Chromosome[]
optimizeFunction() void
selection() void
crossover() void
generateInitialPopulation() void
computeFitness() void
mutation() void
getBestFit() Chromosome
shuffleChromosomes() void
binToDouble(String) double
isValid(double, double, double) boolean
«create»
«create»
«create»
AlgorithmVersion2
selection() void
crossover() void
AlgorithmVersion1
selection() void
crossover() void
AlgorithmVersion3
selection() void
crossover() void
Powered by yFiles

3 farklı genetik algoritmasının implementasyonu için Template design patternı kullanıldı. Genetik algoritmalarının ortak adımları olan generateInitialPopulation(), computeFitness(), mutation() metodları super class içerisinde implement edildi. Algoritmalara göre değişiklik gösteren selection() ve crossOver() metodları ise sub classlar tarafından override edildi. Ayrıca base class içersinde shuffleChrosomes() metodu selection sonrası aynı kromozomların ard arda gelmesini engellemek için implement edilmiştir. Tüm bu metodlar genetik algoritmasının implementasyonu için optimizeFunction() içerisinde kullanılmıştır. OptimizeFunction() metodunda 100 nesil üretilip Populasyon boyutu 100 olarak ayarlanmıştır.

Başlangıç olarak X1 ve X2 değerleri verilen kısıtlamalara göre random olarak üretilip her biri 63 bit olmak üzere toplamda 126 bitden oluşan bir binary string, kromozom genleri olarak tutulmuştur. Bu şekilde kromozomlardan bir populasyon oluşturulup optimize edilecek olan fonksiyondaki sonuçlarına göre fitnessları set edilmiştir. Kısıtlamalara uymayan x1 ve x2 değerlerine sahip olan kromozomların fitnessı 0 olarak set edilmiştir. Selection işlemi her algoritmada belirtilen şekilde yapılmış ardından kromozomların sırasının karışması için kromozom listesi shuffle edilmiştir. Ardından One and Two point cross over seçeneğine göre parent genler ikişer ikişer çaprazlanmıştır. Sonrasında 126 bitten random olarak seçilen bitler değiştirilip mutasyon gerçekleştirilmiştir. Populasyondaki kromozomların fitnessları tekrar hesaplanmış ve bu işlem her nesil için tekrarlanarak fonksiyon için maksimum değerleri veren x1 ve x2 değerleri elde edilmeye çalışılmıştır.

```
########### Version 1 ############
Optimal Value: 126,0390, x1: 2,0453 x2: 2,9546
########### Version 2 ############
Optimal Value: 125,9744, x1: 1,9920 x2: 3,0078
########### Version 3 ############
Optimal Value: 125,8906, x1: 2,1250 x2: 2,8750
```

Q2)


Chromosome
fitness double
genes String
genesSize int
Chromosome()
Chromosome(String, double)
doubleToBin(double) String
getFitness() double
setGenes(String) void
setFitness(double) void
getGenes() String
getGenesSize() int
getGenesX1() String
getGenesX2() String
compareTo(Object) int
«create»
AlgorithmVersion2
selection() void
crossover() void
AlgorithmVersion3
selection() void
crossover() void
AlgorithmVersion1
selection() void
crossover() void
GUI
scheduledExecutorService ScheduledExecutorService
optValues List<Double>
iteration int
threads List<Thread>
series List<Series<String, Number>>
isPause boolean
isRun boolean
isStop boolean
lineChart LineChart<String, Number>
lineChart2 LineChart<String, Number>
lineChart3 LineChart<String, Number>
main(String[]) void
drawCharts() void
start(Stage) void
stop() void
getXAxis() CategoryAxis
getYAxis() NumberAxis
createLineChart(String) LineChart<String, Number>
runAlgorithms() void
stopAlgorithms() void
clearCharts() void
pauseAlgorithms() void
startSchedule() void
GeneticAlgorithm
chromosomes Chromosome[]
optimizeFunction() void
selection() void
crossover() void
generateInitialPopulation() void
computeFitness() void
mutation() void
getBestFit() Chromosome
shuffleChromosomes() void
binToDouble(String) double
isValid(double, double, double) boolean
updateGUI(double) void
Powered by yFiles

Start butonuna basıldığında 3 algoritmanın da aynı anda çalışması amacıyla her bir algoritma için bir

Thread oluşturuldu. Çizgi grafiklerinin gerçek zamanlı olarak gösterilmesi için Scheduled Execute Service kullanıldı. Start butonuna basıldığında algoritmaların eş zamanlı çalışması ve grafiklerin güncellenmesi için Observer Design patternı kullanılabilirdi. Böylece maksimum fitness değerinde bir değişiklik olduğunda tüm chartlar notify edilir ve güncel veriler her chart için ekranda ayrı ayrı gösterilirdi. Bu yaklaşımı uygulamak yerine GUI sınıfı içerisinde 3 elemanlı static bir double listesi tutuldu. optimizeFunction metodunda bulunan döngü içerisinde her iterasyonda bu liste updateGUI() metodu ile güncellendi.

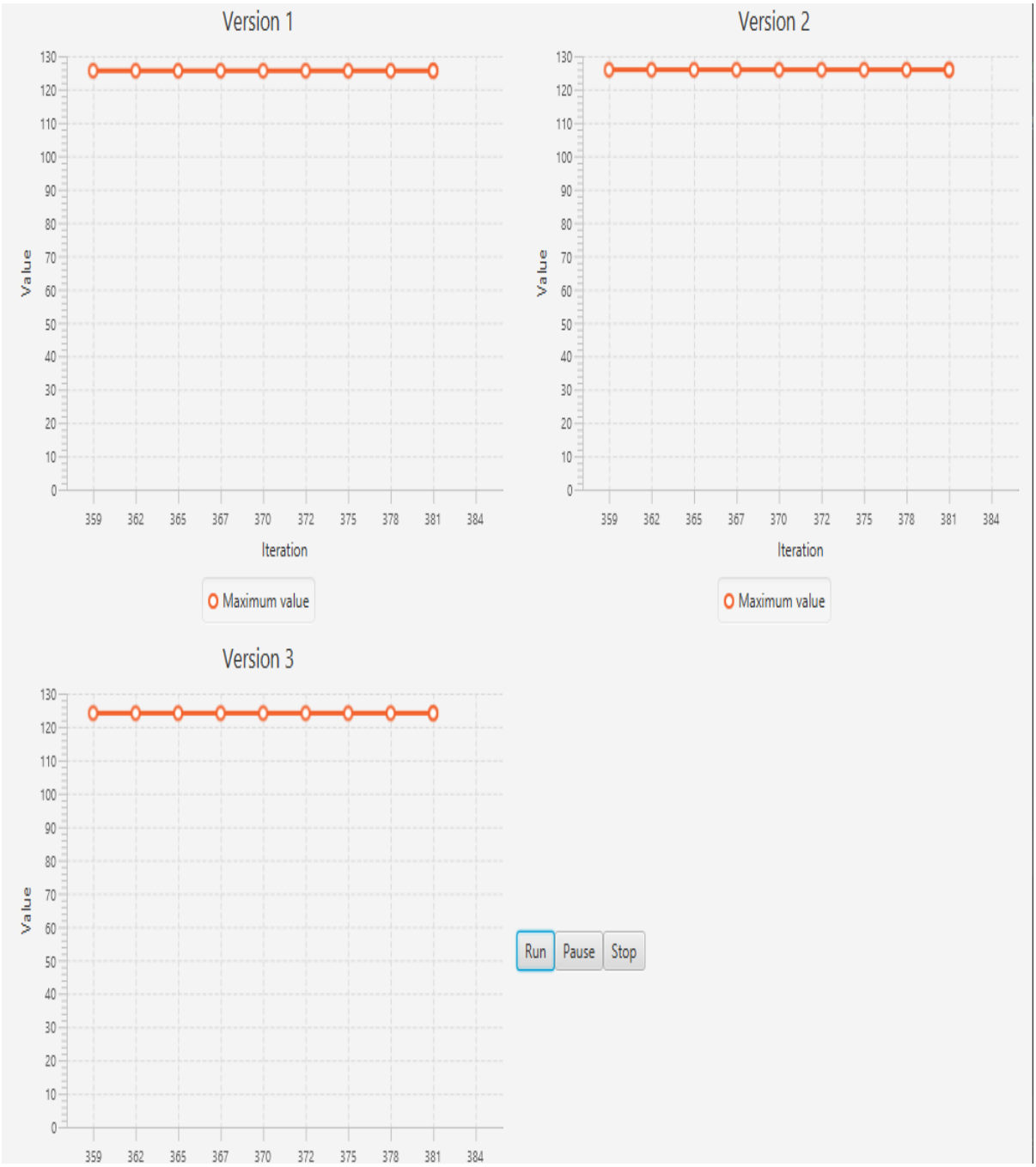
```
private void updateGUI(double maxFit){

    if(this instanceof AlgorithmVersion1){
        GUI.optValues.set(0,maxFit);
    }
    else if(this instanceof AlgorithmVersion2){
        GUI.optValues.set(1,maxFit);
    }
    else if(this instanceof AlgorithmVersion3){
        GUI.optValues.set(2,maxFit);
    }
}
```

Start butonuna basıldığında oluşturulan threadler bir listeye alındı. Pause butonuna basıldığında

Öncelikle GUI' nin durması için Scheduled Execute Service' nin shutdown() metodu çağrıldı. Algoritmaların duraklatılması için ise thread listesindeki threadlerin suspend() metodu çağrıldı.

Stop butonuna basıldığında pausedeki ile aynı şekilde GUI durdurulup tüm threadlerin stop() metodu çağrıldı. Ayrıca tüm chartlardaki datalar temizlendi. Ard arda thread oluşumunu engellemek için start butonuna basıldıktan sonra tekrar basılması engellendi. Aynı şekilde stop butonuna ard arda basılması durumunda uyarı verildi. Iterasyonlar arasındaki farkın daha rahat bir şekilde görülebilmesi için her iterasyonda 1 saniye delay yapıldı.


Version 1
Value
Iteration
Maximum value
Version 2
Value
Iteration
Maximum value
Version 3
Value
Run
Pause
Stop