[](http://1.bp.blogspot.com/_iwFQ7Phn1bc/RhGtboxvKqI/AAAAAAAAALk/RlMy1uAXLbw/s1600-h/new+logo.jpg)

**FACULTY OF COMPUTER SYSTEMS & SOFTWARE ENGINEERING**

**BCS2313**

**ARTIFICIAL INTELLIGENCE TECHNIQUES**

Assignment 3

Prepared By:

**NABILA ZAINAL**

**CB14036**

**Lecturer Name :**

Mr. Abdul Sahli Fakharudin

**ANSWER 1**

package org.jenetics.example;

import static java.lang.Math.PI;

import static java.lang.Math.cos;

import static java.lang.Math.sin;

import static org.jenetics.engine.EvolutionResult.toBestPhenotype;

import static org.jenetics.engine.limit.bySteadyFitness;

import org.jenetics.DoubleGene;

//import org.jenetics.MeanAlterer;

import org.jenetics.Mutator;

import org.jenetics.Optimize;

import org.jenetics.Phenotype;

import org.jenetics.SinglePointCrossover;

import org.jenetics.engine.Engine;

import org.jenetics.engine.EvolutionStatistics;

import org.jenetics.engine.codecs;

import org.jenetics.util.DoubleRange;

import org.jenetics.MultiPointCrossover; //2-point crossover

import org.jenetics.RouletteWheelSelector;

public class Assignment3 {

// The fitness function.

private static double fitness(final double x) {

//Bella Lab6 GA

return (6\*(Math.pow(x, 2))) - (3\*x); //function is f(x)=6x2-3x

}

public static void main(final String[] args) {

final Engine<DoubleGene, Double> engine = Engine

// Create a new builder with the given fitness

// function and chromosome.

.builder(

Assignment3::fitness,

codecs.ofScalar(DoubleRange.of(-5.0, 5.)))

.populationSize(10) //10 chromosomes

.optimize(Optimize.MAXIMUM)

.survivorsSelector(new RouletteWheelSelector<>()) //randomly select

.alterers( // alterers (singlepointcrossover, multipointcrossover, mutation, etc)

//new MultiPointCrossover<>(0.1),

new SinglePointCrossover<>(0.5), //1-point crossover

new Mutator<>(0.1)) //mutation; letak nilai rendah (1%)

// value of mutation and crossover can be changed acc. to yr desires

//new MeanAlterer<>(0.6))

// Build an evolution engine with the

// defined parameters.

.build();

// Create evolution statistics consumer.

final EvolutionStatistics<Double, ?>

statistics = EvolutionStatistics.ofNumber();

final Phenotype<DoubleGene, Double> best = engine.stream()

// Truncate the evolution stream after 7 "steady"

// generations.

//.limit(bySteadyFitness(7))

// The evolution will stop after maximal 100

// generations.

.limit(100)

// Update the evaluation statistics after

// each generation

.peek(statistics)

// Collect (reduce) the evolution stream to

// its best phenotype.

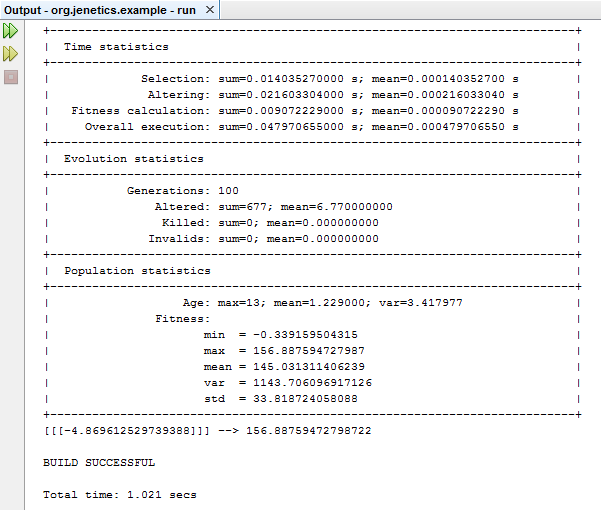
.collect(toBestPhenotype());

System.out.println(statistics);

System.out.println(best);

}

}



**ANSWER 2**

package org.jenetics.example;

import static java.lang.Math.PI;

import static java.lang.Math.cos;

import static java.lang.Math.sin;

import static org.jenetics.engine.EvolutionResult.toBestPhenotype;

import static org.jenetics.engine.limit.bySteadyFitness;

import org.jenetics.DoubleGene;

//import org.jenetics.MeanAlterer;

import org.jenetics.Mutator;

import org.jenetics.Optimize;

import org.jenetics.Phenotype;

import org.jenetics.SinglePointCrossover;

import org.jenetics.engine.Engine;

import org.jenetics.engine.EvolutionStatistics;

import org.jenetics.engine.codecs;

import org.jenetics.util.DoubleRange;

import org.jenetics.MultiPointCrossover; //2-point crossover

import org.jenetics.RouletteWheelSelector;

public class Assignment32 {

// The fitness function.

private static double fitness(final double x) {

//Bella Lab6 GA

return (7\*(Math.pow(x, 2))) + (5\*x); //function is f(x)=7x2+5x

}

public static void main(final String[] args) {

final Engine<DoubleGene, Double> engine = Engine

// Create a new builder with the given fitness

// function and chromosome.

.builder(

Assignment32::fitness,

codecs.ofScalar(DoubleRange.of(-5.0, 5.0)))

.populationSize(10) //10 chromosomes

.optimize(Optimize.MINIMUM)

.survivorsSelector(new RouletteWheelSelector<>()) //randomly select

.alterers( // alterers (singlepointcrossover, multipointcrossover, mutation, etc)

new MultiPointCrossover<>(0.6),

//new SinglePointCrossover<>(0.5), //1-point crossover

new Mutator<>(0.05)) //mutation; letak nilai rendah (1%)

// value of mutation and crossover can be changed acc. to yr desires

//new MeanAlterer<>(0.6))

// Build an evolution engine with the

// defined parameters.

.build();

// Create evolution statistics consumer.

final EvolutionStatistics<Double, ?>

statistics = EvolutionStatistics.ofNumber();

final Phenotype<DoubleGene, Double> best = engine.stream()

// Truncate the evolution stream after 7 "steady"

// generations.

//.limit(bySteadyFitness(7))

// The evolution will stop after maximal 100

// generations.

.limit(100)

// Update the evaluation statistics after

// each generation

.peek(statistics)

// Collect (reduce) the evolution stream to

// its best phenotype.

.collect(toBestPhenotype());

System.out.println(statistics);

System.out.println(best);

}

}

