

```

package com.badmanners;

import java.util.Arrays;
import java.util.Random;

public class Main
{
    static Random rnd = new Random();
    static class Point2D
    {
        double x, y;
        public Point2D()
        {
            x = rnd.nextDouble() * 2 - 1;
            y = rnd.nextDouble() * 2 - 1;
        }
    }
    static class Point10D
    {
        double x[] = new double[10];
        public Point10D()
        {
            for(int i = 0; i < x.length; i++)
                x[i] = rnd.nextDouble() * 2 - 1;
        }
    }
    static class Matrix
    {
        double[][] matrix = new double[30][30];
        public Matrix() {}
        public Matrix(Point10D[] points30D)
        {
            double max = 0;
            for(int i = 0; i < 30; i++)
                for(int j = 0; j < 30; j++)
                {
                    matrix[i][j] = calculateDistance10(points30D[i],
points30D[j]);
                    if(matrix[i][j] > max)
                        max = matrix[i][j];
                }
            for(int i = 0; i < 30; i++)
                for(int j = 0; j < 30; j++)
                    matrix[i][j] /= max;
        }
        public Matrix(Point2D[] points2D)
        {
            double max = 0;
            for(int i = 0; i < 30; i++)
                for(int j = 0; j < 30; j++)
                {
                    matrix[i][j] = calculateDistance2(points2D[i], points2D[j]);
                    if(matrix[i][j] > max)
                        max = matrix[i][j];
                }
            for(int i = 0; i < 30; i++)
                for(int j = 0; j < 30; j++)
                    matrix[i][j] /= max;
        }
        double getFitness()
        {
            double cnt = 0;
            for(int i = 0; i < 30; i++)
                for(int j = 0; j < 30; j++)

```

```

        cnt += matrix[i][j];
    }
    return cnt;
}
}
static class Chromosome
{
    double fitness;
    Point2D[] points2D = new Point2D[30];
    public Chromosome()
    {
        for(int i = 0; i < 30; i++)
            points2D[i] = new Point2D();
    }
    double calculateFitness(Matrix distance30)
    {
        Matrix distance2 = new Matrix(points2D);
        fitness = calculateDifference(distance30, distance2).getFitness();
        return fitness;
    }
}
static class Generation
{
    Chromosome[] chromosomes = new Chromosome[100];
    public Generation()
    {
        for(int i = 0; i < 100; i++)
            chromosomes[i] = new Chromosome();
    }
    double calculateFitness(Matrix distance30)
    {
        for(int i = 0; i < 100; i++)
            chromosomes[i].calculateFitness(distance30);
        Arrays.sort(chromosomes, (o1, o2) ->
        {
            if(o1.fitness > o2.fitness)
                return 1;
            else if(o1.fitness < o2.fitness)
                return -1;
            else
                return 0;
        });
        return chromosomes[0].fitness;
    }
}

static double calculateDistance10(Point10D first, Point10D second)
{
    double cnt = 0.0;
    for(int i = 0; i < 10; i++)
        cnt += (first.x[i] - second.x[i]) * (first.x[i] - second.x[i]);
    return Math.sqrt(cnt);
}
static double calculateDistance2(Point2D first, Point2D second)
{
    return Math.sqrt((first.x - second.x) * (first.x - second.x) + (first.y
- second.y) * (first.y - second.y));
}
static Matrix calculateDifference(Matrix first, Matrix second)
{
    Matrix temp = new Matrix();
    for(int i = 0; i < 30; i++)
        for(int j = 0; j < 30; j++)
            temp.matrix[i][j] = Math.abs(first.matrix[i][j] -
second.matrix[i][j]);
}

```

```

        return temp;
    }
    static Chromosome crossover(Chromosome first, Chromosome second)
    {
        Chromosome chromosome = new Chromosome();
        for(int i = 0; i < chromosome.points2D.length; i++)
            if(rnd.nextInt(60) == 0)
                chromosome.points2D[i] = new Point2D();
            else
                chromosome.points2D[i] = rnd.nextInt(2) == 0 ? first.points2D[i]
: second.points2D[i];
        return chromosome;
    }
    static Generation createNewGeneration(Generation prevGeneration)
    {
        Generation generation = new Generation();
        for(int i = 0; i < generation.chromosomes.length; i++)
            generation.chromosomes[i] =
crossover(prevGeneration.chromosomes[rnd.nextInt(50)],
prevGeneration.chromosomes[rnd.nextInt(50)]);
        return generation;
    }

    public static void main(String[] args)
    {
        Point10D[] point10Ds = new Point10D[30];
        for(int i = 0; i < 30; i++)
            point10Ds[i] = new Point10D();
        Matrix distance30 = new Matrix(point10Ds);

        Generation prevGeneration = new Generation(), generation;
        prevGeneration.calculateFitness(distance30);

        for(int i = 0; ; i++)
        {
            generation = createNewGeneration(prevGeneration);
            double fitness = generation.calculateFitness(distance30);
            System.out.println(i + ": " + fitness);
            if(fitness <= 0.1)
                break;
            prevGeneration = generation;
        }

        System.out.println();
    }
}

```