

Task1

```
import java.util.ArrayList;
import java.util.List;
import java.util.Random;
import java.util.concurrent.ThreadLocalRandom;
/**
 * Created by yauheni on 07.09.16.
 */
class Individual implements Comparable<Individual> {
    public static final int GENE_LENGTH = 20;
    private List<Double> genes;
    private double fitnessValue;
    public double getGene(int index) {
        return genes.get(index);
    }
    public void setGene(int index, double gene) {
        genes.set(index, gene);
    }
    public Individual() {
        Random r = new Random();
        genes = new ArrayList<>();
        for (int i = 0; i < GENE_LENGTH; ++i) {
            genes.add(r.nextInt(1000)/ 100.0 - 5);
        }
        fitnessValue = getFitness();
    }
    public double getFitnessValue() {
        return fitnessValue;
    }
    public void setFitnessValue(double fitnessValue) {
        this.fitnessValue = fitnessValue;
    }
    public void generateIndividual() {
    }
    public double getFitness() {
        double temp = 0.0;
        for (int i = 0; i < genes.size(); i++) {
            fitnessValue += genes.get(i) * Math.sin(Math.abs(genes.get(i)));
        }
        return fitnessValue;
    }
    @Override
    public int compareTo(Individual o) {
        return (o.getFitness() < fitnessValue ? 1 : (o.getFitness() == fitnessValue) ? 0 : -1);
    }
}
```

```
class Population {
    public static final int POPULATION_SIZE = 20;
    private List<Individual> individuals;
```

```

    public Population(List<Individual> individuals) {
        this.individuals = individuals;
    }
    public Population() {
    }
    public static Population newPopulation() {
        Population pop = new Population();
        List<Individual> list = new ArrayList<>();
        for (int i = 0; i < 20; i++) {
            list.add(new Individual());
        }
        pop.setIndividuals(list);
        return pop;
    }
    public List<Individual> getIndividuals() {
        return individuals;
    }
    public void setIndividuals(List<Individual> individuals) {
        this.individuals = individuals;
    }
    public Individual getFittestIndividual() {
        try {
            Collections.sort(individuals);
        } catch (Exception e) {
        }
        return individuals.get(0);
    }
    public double averageFitness() {
        double temp = 0.0;
        for (int i = 0; i < 20; i++) {
            temp += individuals.get(i).getFitness();
        }
        return temp/20;
    }
    public double getMax() {
        return getFittestIndividual().getFitness();
    }
}

```

```

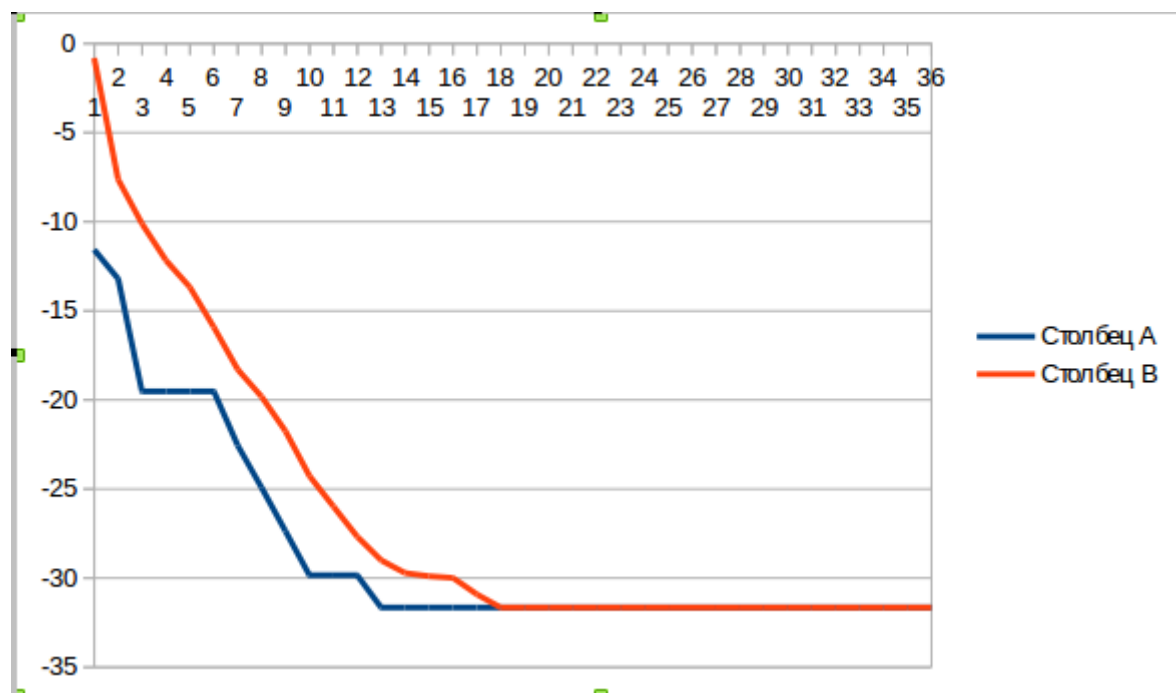
import java.util.ArrayList;
import java.util.List;
import java.util.concurrent.ThreadLocalRandom;
/**
 * Created by yauheni on 07.09.16.
 */
public class Main {
    public static void main(String[] args) throws InterruptedException {
        Population pop = Population.newPopulation();
        List<Double> tempList = new ArrayList<>();
        int count=0;
        while(true) {
            count++;
            if(count == 100)
                break;
            System.out.print(pop.getMax() + " ");
            System.out.println(pop.averageFitness());
            Population newPopulation = new Population();

```

```

List<Individual> newIndividuals = new ArrayList<>();
List<Individual> individuals = pop.getIndividuals();
for (int i = 0; i < 10; i++) {
    int first = ThreadLocalRandom.current().nextInt(0, 20);
    int second = ThreadLocalRandom.current().nextInt(0, 20);
    int razrez = ThreadLocalRandom.current().nextInt(0, 20);
    Individual firstInd = individuals.get(first);
    Individual secondInd = individuals.get(second);
    Individual newFirst = new Individual();
    Individual newSecond = new Individual();
    for (int j = 0; j < 20; j++) {
        if(j < razrez) {
            newFirst.setGene(j, firstInd.getGene(j));
            newSecond.setGene(j, secondInd.getGene(j));
        } else {
            newFirst.setGene(j, secondInd.getGene(j));
            newSecond.setGene(j, firstInd.getGene(j));
        }
    }
    newIndividuals.add(newFirst);
    newIndividuals.add(newSecond);
}
newPopulation.setIndividuals(newIndividuals);
pop = newPopulation;
}
System.out.println("=====");
for (int i = 0; i < tempList.size(); i++) {
    System.out.println(tempList.get(i));
}
}
}

```



red line is average, blue line is fitness.

TASK2

```
import java.util.ArrayList;
```

```
import java.util.List;
```

```
import java.util.Random;
```

```
import java.util.concurrent.ThreadLocalRandom;
```

```
/**
```

```
 * Created by yauheni on 07.09.16.
```

```
 */
```

```
class Individual implements Comparable<Individual> {
```

```
    public static final int GENE_LENGTH = 10;
```

```
    private List<Integer> genes;
```

```
    private double fitnessValue;
```

```
    public int getGene(int index) {  
        return genes.get(index);  
    }
```

```
    public void setGene(int index, int gene) {  
        genes.set(index, gene);  
    }
```

```
    public Individual() throws InterruptedException {  
        Random r = new Random();  
        genes = new ArrayList<>();  
        for (int i = 0; i < GENE_LENGTH; ++i) {  
            //Thread.sleep(50);  
            genes.add(r.nextInt(2));  
        }  
        fitnessValue = getFitness();  
    }
```

```
    public double getFitnessValue() {  
        return fitnessValue;  
    }
```

```
    public void setFitnessValue(int fitnessValue) {  
        this.fitnessValue = fitnessValue;  
    }
```

```
    public void generateIndividual() {  
    }
```

```
    public double getX() {  
        String t = "";  
        for (int i = 1; i < genes.size(); i++) {  
            t += genes.get(i);  
        }  
        if(genes.get(0) == 0) {  
            return (double)Integer.parseInt(t, 2)/ 99;  
        } else {  
            return (double) -Integer.parseInt(t,2)/ 99;  
        }  
    }
```

```
    public double getFitness() {  
        return getX() * Math.sin(Math.abs(getX()));  
    }
```

```
    @Override
```

```
    public int compareTo(Individual o) {  
        return (o.getFitness() < getFitness() ? 1 : (o.getFitness() == getFitness()) ? 0 : -1);  
    }
```

```
}
```

```
import java.util.ArrayList;
```

```

import java.util.List;
import java.util.concurrent.ThreadLocalRandom;
/**
 * Created by yauheni on 07.09.16.
 */
public class Main {
    public static void main(String[] args) throws InterruptedException {
        Population pop = Population.newPopulation();
        List<Double> tempList = new ArrayList<>();
        int count=0;
        while(true) {
            count++;
            if(count == 20)
                break;
//      System.out.println(pop.getFittestIndividual().getX());
//      System.out.print(pop.getFittestIndividual().getFitness() + " ");
//      System.out.println(pop.averageFitness());
//      for (int i = 0; i < 20; i++) {
//          pop.getFittestIndividual();
//          System.out.print(pop.getIndividuals().get(i).getX() + " ");
//          System.out.println(pop.getIndividuals().get(i).getFitness());
//      }
//      System.out.println("=====");
        Population newPopulation = new Population();
        List<Individual> newIndividuals = new ArrayList<>();
        List<Individual> individuals = pop.getIndividuals();
        for (int i = 0; i < 10; i++) {
            int first = ThreadLocalRandom.current().nextInt(0, 10);
            int second = ThreadLocalRandom.current().nextInt(0, 10);
            int razrez = ThreadLocalRandom.current().nextInt(0, 10);
            Individual firstInd = individuals.get(first);
            Individual secondInd = individuals.get(second);
            Individual newFirst = new Individual();
            Individual newSecond = new Individual();
            for (int j = 0; j < 10; j++) {
                if(j < razrez) {
                    newFirst.setGene(j, firstInd.getGene(j));
                    newSecond.setGene(j, secondInd.getGene(j));
                } else {
                    newFirst.setGene(j, secondInd.getGene(j));
                    newSecond.setGene(j, firstInd.getGene(j));
                }
            }
            newIndividuals.add(newFirst);
            newIndividuals.add(newSecond);
        }
        newPopulation.setIndividuals(newIndividuals);
        pop = newPopulation;
    }
    System.out.println("=====");
}
}

```