Did Kiryl Chepeleu, AS-35

Task: All One Problem
I define 50 dogs with 1000 binary genes. I tryed a evolution with one-point-crossover with mutation rate being 1/1000.

Source code in Java:

```
package com.siit.lab1;
import com.siit.lab1.core.Dog;
import com.siit.lab1.core.DogManager;
public class App
{
    public static void main( String[] args )
    {
        int maxSteps = Integer.MAX_VALUE;
        int dogs = 50;
        int gensPerDog = 1000;
        DogManager dogManager = new DogManager(dogs,gensPerDog);
        dogManager.evolution(maxSteps);
    }
}
```

```
package com.siit.lab1.core;
import java.util.Random;
public class Dog implements Comparable{
    private boolean gens[];
    private int numberOfGens;
    public static final Random random = new Random();
    public Dog(final int numberOfGens){
        if(numberOfGens < 1){
            this.numberOfGens = 1000;
        }else {
            this.numberOfGens = numberOfGens;
        }
        gens = new boolean[numberOfGens];
        for(int i=0;i<gens.length;i++){
            gens[i] = random.nextBoolean();
        }
    }
    private Dog(){

    }
    public int fitness(){
        int res=0;
        for(boolean i:gens){
            if(i){
                res++;
            }
        }
        return res;
    }
```

```
    @Override
    public String toString(){
        StringBuilder stringBuilder = new StringBuilder();
        for(boolean i:gens){
            stringBuilder.append(i?1:0);
        }
        return stringBuilder.toString();
    }
    public Dog getCopy(){
        Dog dog = new Dog();
        dog.gens = this.gens.clone();
        dog.numberOfGens = this.numberOfGens;
        return dog;
    }
    public static void crossOver(Dog mother,Dog father){
        if(mother.numberOfGens!=father.numberOfGens){
            return;
        }
        final int border = random.nextInt(mother.numberOfGens);
        for(int i = 0; i < border; i++){
            boolean tmp = father.gens[i];
            father.gens[i] = mother.gens[i];
            mother.gens[i] = tmp;
        }
        for(int i = 0; i < mother.gens.length; i++){
            if(random.nextInt(1000)<1){
                father.gens[i] = !father.gens[i];
            }
            if(random.nextInt(1000)<1){
                mother.gens[i] = !mother.gens[i];
            }
        }
    }
    @Override
    public int compareTo(Object o) {
        if(o instanceof Dog){
            return ((Dog)o).fitness() - this.fitness();
        }else {
            return 0;
        }
    }
}

package com.siit.lab1.core;
import java.lang.reflect.Array;
import java.util.Arrays;

public class DogManager {
    private Dog dogs[] = null;
    private final int numberOfDogs;
    public DogManager(int numberOfDogs,int gensPerDog){
        if(numberOfDogs < 1){
```

```
            this.numberOfDogs = 50;
        }else {
            this.numberOfDogs = numberOfDogs;
        }
        dogs = new Dog[numberOfDogs];
        for(int i = 0;i<numberOfDogs;i++){
            dogs[i]= new Dog(gensPerDog);
        }
    }
    public void evolution(int steps){
        Dog children[] = dogs.clone();
        double lastAvgFitness=-1;
        int lastFitnessCount=0;

        for(int step = 0;step < steps && lastFitnessCount < 10;step++) {
            Arrays.sort(dogs);
            int half = numberOfDogs / 2;
            for (int i = 0; i < half; i++) {
                int i1 = Dog.random.nextInt(half);
                int i2;
                do{
                    i2 = Dog.random.nextInt(half);
                }while (i1==i2);
                Dog d1 = dogs[i1].getCopy();
                Dog d2 = dogs[i2].getCopy();
                Dog.crossOver(d1, d2);
                children[i * 2] = d1;
                children[i * 2 + 1] = d2;
            }
            Arrays.sort(children);
            Dog tmp[] = children;
            children = dogs;
            dogs = tmp;
            double avgFitness = 0;
            for (int i = 0; i < dogs.length; i++) {
                avgFitness+=dogs[i].fitness();
            }
            if( Math.abs(lastAvgFitness - avgFitness) <0.00001 ){
                lastFitnessCount ++;
            }else{
                lastFitnessCount = 0;
            }
            lastAvgFitness = avgFitness;
            System.out.println(step+"\t"+avgFitness/dogs.length+"\t"+ dogs[0].fitness());
        }
    }
}
```

Plot graphic average fitness and best fitness