1 exercise – Andrey Cheslow

Conditions

1. Create 100 random binary-chromosomes each with 1000 genes.
2. Fitness is the number of “1” in one chromosome – the more the better.
3. Select 2 chromosomes by roulette method
4. Create a child chromosome by a uniform-crossover.
5. Repeat from 2. to 5. 100 times and create the next generation.
6. Repeat 6. until the fitness value does not change any more.

Source Code

```
package siit_1;
import java.util.Random;
/**
 *
 * @author Andrey Cheslow
 */
public class SIIT_1 {
   public static void main(String[] args) {
      int generation[][]=new int[100][1000];
      int next_generation[][]=new int[100][1000];
      int number_of_choices[] =new int [100];
      Random rnd=new Random();
      //make first generation
      for(int i=0;i<100;i++){
         for(int j=0;j<1000;j++){

            generation[i][j]=(rnd.nextInt(2));
         }
      }
      double probability[] = new double[100];
      int fitness[] = new int[100];
      int sum_fit=0;
      while (fitness[0]<1000){
         for(int i=0;i<100;i++){
            fitness[i]=0;
         }
         for(int i=0;i<100;i++){
            number_of_choices[i]=0;
         }
         //calculating fitness
         for(int i=0;i<100;i++){
            for(int j=0;j<1000;j++){
               if (generation[i][j]==1) fitness[i]++;
            }
         }
         //sort generation according to ascending
         sort(fitness,generation);
         sum_fit=0;
         //calculating probability
         for(int j=0;j<100;j++){
               sum_fit+=fitness[j];
```

```
            }
        for(int j=0;j<100;j++){
              probability[j]=(double)(fitness[j])/sum_fit;
            }
        System.out.println(fitness[0]+" "+sum_fit/100);

        make_new_gen(probability,generation,next_generation,number_of_choices);
        for(int i=0;i<100;i++){
          System.out.println(number_of_choices[i]);
        }
        generation=next_generation;
      }
    }
    public static void sort(int[] fitness,int[][] generation){
      int temp=0;

      for(int i=0;i<99;i++){
        for(int j=i+1;j<100;j++){
          if (fitness[j]>fitness[i]){
            temp=fitness[i];
            fitness[i]=fitness[j];
            fitness[j]=temp;
            int temps[]=generation[i];
              generation[i] = generation[j];
              generation[j] = temps;
          }
        }
      }
    }
    public static void make_new_gen(double[] probability,int[][] generation,int[][] next_generation, int[]
number_of_choices){
      Random rnd= new Random();
      for(int i=0;i<100;i++){
        //roulette method
        double p1=(double)(rnd.nextInt(100))/100;
        double p2=(double)(rnd.nextInt(100))/100;

        int nump1=0,nump2=0;
        int k=0;
        while(p1>0){

          if((p1-probability[k])>0){k++;p1-=probability[k];}
          else {nump1=k;p1-=probability[k];}
        }
         k=0;
        while(p2>0){

          if((p2-probability[k])>0){k++;p2-=probability[k];}
          else {nump2=k;p2-=probability[k];}
        }
        //truncate method
        /*int nump1=rnd.nextInt(50);
        int nump2=rnd.nextInt(50);*/
        number_of_choices[nump1]++;
```

```
    number_of_choices[nump2]++;
    for(int f=0;f<1000;f++){
      int orrr=rnd.nextInt(2);
      //uniform crossover
      if(orrr==0)
      next_generation[i][f]=generation[nump1][f];
      else next_generation[i][f]=generation[nump2][f];
    }
}}}
```

number of generation / number of chromosome	First	20-th	last
1	2	7	4
2	1	1	1
3	1	1	1
4	4	2	2
5	1	2	2
6	1	4	2
7	2	2	2
8	1	3	1
9	0	5	5
10	1	2	0
11	3	2	2
12	3	0	1
13	2	1	2
14	3	3	2
15	1	2	3
16	4	2	0
17	0	0	6
18	0	1	1
19	1	1	1
20	3	2	0
21	3	8	2
22	5	2	2
23	3	4	1
24	1	1	1
25	6	3	3
26	1	4	3
27	2	1	1
28	0	1	2
29	3	2	3
30	1	0	4
31	2	0	3
32	1	5	1
33	2	2	2
34	2	3	6

35	3	0	2
36	1	1	2
37	2	2	3
38	2	2	1
39	3	2	0
40	2	2	1
41	1	1	0
42	2	3	0
43	0	1	2
44	4	2	4
45	1	1	0
46	4	3	1
47	4	0	0
48	3	1	0
49	5	4	1
50	3	0	4
51	2	3	3
52	1	0	3
53	2	5	4
54	1	1	4
55	3	2	1
56	1	1	3
57	0	0	2
58	3	1	0
59	0	2	1
60	3	3	2
61	1	1	1
62	4	1	6
63	0	3	3
64	3	2	1
65	3	1	1
66	6	2	1
67	2	5	1
68	1	3	4
69	3	1	4
70	1	2	0
71	1	0	2
72	4	5	1
73	3	2	2
74	0	3	2
75	1	0	1
76	0	2	1
77	4	2	2
78	3	2	2
79	1	3	2
80	3	1	5
81	1	0	4

82	2	0	1
83	2	2	3
84	0	0	0
85	1	2	2
86	1	2	4
87	3	4	1
88	2	1	4
89	1	2	4
90	2	2	1
91	1	0	0
92	1	6	3
93	1	2	1
94	4	5	5
95	3	3	1
96	2	2	1
97	1	1	1
98	0	1	3
99	4	0	3
100	2	2	0