```
1     package com.fwilhelm.JImport;
2     import java.io.*;
3     import java.util.*;
4
      ¡¥¥¥¥¥¥¥¥
5   ÕÖ×public class JImporter
      £¤¦¦¦¦¦¦¦
6     §{
7     §
8     §/**Variablen für das Öffnen und Lesen der Datei*/
9     § í private FileInputStream file;
10    § í private BufferedInputStream bInputStream;
11    § í private DataInputStream input;
12    §
13    § í private FileWriter fileWriter;
14    § í private PrintWriter fileOut;
15    §
16    § í private String fileInput;
17    § í private String fileOutput;
18    §
19    § í private StringTokenizer st;
20    §
21    § í private SumData sumdata = new SumData();
22    § í private Vector v = new Vector(1,1);
23    §
24    §/**Daten einer Meßreihe*/
25    § í public Data daten = new Data();
26    §/**Gemittelte Daten aus n Messungen*/
27    § í public Data gemitteltеDaten = new Data();
28    §
29    §/**
30    §/*Consturctor
31    §*/
      § ¬¹¹¹¹¹¹¹¹¹
32    §Þßàpublic JImporter(String inputFile, String outputFile)
      § ªË¹¹¹¹¹¹¹¹
33    §  §{
34    §  ¨¹¹ fileInput = inputFile;
35    §  ¨¹¹ fileOutput = outputFile;
36    §  ¨¹¹ this.init();
37    §  ©}
38    §
39    §/*Abkürzung für System.out.println()*/
      § ¬¹¹¹¹¹¹¹¹¹
40    §Þßàprivate void p(String text)
      § ªË¹¹¹¹¹¹¹¹
41    §  §{
42    §  ¨¹¹ System.out.println(text);
43    §  ©}
44    §
45    §/*Hier wird der FileReader initialisiert*/
      § ¬¹¹¹¹¹¹¹¹¹
46    §Þßàprivate void init()
      § ªË¹¹¹¹¹¹¹¹
47    §  §{
48    §  ¨¹¹´try
49    §  §  §{
50    §  §  ¨¹¹ file = new FileInputStream(fileInput);
51    §  §  ¨¹¹ bInputStream = new BufferedInputStream(file);
52    §  §  ¨¹¹ input = new DataInputStream(bInputStream);
53    §  §  §}
      §  §  § ¬¹¹¹¹¹¹¹¹¹
54    §  §  §ðîìcatch(Exception e)
      §  §  § ªË¹¹¹¹¹¹¹¹
55    §  §  §  §{
56    §  §  §  ¨¹¹ p("Fehler in init:JImporter \n" + e);
57    §  §  ©  ©}
58    §  ©}
59    §
60    §/*Gibt die nächste Zeile aus; das File, das beim
61    §/*Erzeugen des Objektes geöffnet wurde*/
      § ¬¹¹¹¹¹¹¹¹¹
62    §Þßàprivate String nextLine()
      § ªË¹¹¹¹¹¹¹¹
63    §  §{
64    §  ¨¹¹´try
65    §  §  §{
66    §  §  ¨¹í String s = new String();
67    §  §  ¨¹³´if((s = input.readLine())!= null)
68    §Â¹Ä  § 6¾¹¹ return s;
69    §  §  § ¶´else
70    §  §  § ¸§{
71    §  §  § ¸¨¹¹ input.close();
72    §Â¹Ä  § ¸¾¹¹ return null;
73    §  §  § È }
74    §  §  §}
      §  §  § ¬¹¹¹¹¹¹¹¹¹
75    §  §  §ðîìcatch(Exception e)
      §  §  § ªË¹¹¹¹¹¹¹¹
76    §  §  §  §{
77    §  §  §  ¨¹¹ p("" + e);
```

```
78    §Â¹Ä  §  ¨¹¹ return null;
79    §  §  ©  ©}
80    §  §
81    §  ©}
82    §
83    §
84    §/**
85    §/*Einlesen einer Messung; da auch die Anzahl der Generationen
86    §/*eingelesen werden paßt sie sich automatisch an!
87    §*/
      § ¬¹¹¹¹¹¹¹¹¹
88    §Þßà public boolean nextMeasure()
      § ªË¹¹¹¹¹¹¹¹
89    §  §{
90    §  ¨¹¹Í String line;
91    §  ¨¹¹Í String token = "";
92    §  ¨¹¹´try
93    §  §  §{
94    §  §  ¨¹¹ line = nextLine(); //Erste Zeile enthält keine Daten
95    §  §  §
96    §  §  ¨¹³´if(line == null)  //Wenn die erste Zeile einer Messung null
97    §Â¹Ä  § ¶¾¹¹ return false;  //ist, gibt es nichts mehr zum Einlesen
98    §  §  §
99    §  §  ¨¹¹ line = nextLine();
100   §  §  §/*zweite Zeile enthält Daten*/
101   §  §  ¨¹¹ st = new StringTokenizer(line);
102   §  §  ¨¹¹ token = st.nextToken(); //Enthält String "Generationen:"
103   §  §  ¨¹¹ token = st.nextToken(); //Enthält den Wert Generationen
104   §  §  ¨¹¹ daten.generation = Integer.parseInt(token);
105   §  §  §
106   §  §  ¨¹¹ token = st.nextToken(); //Enthält den String "Population:"
107   §  §  ¨¹¹ token = st.nextToken(); //Enthält den Wert population
108   §  §  ¨¹¹ daten.population = Integer.parseInt(token);
109   §  §  §
110   §  §  ¨¹¹ token = st.nextToken(); //Enthält den String "Crossoverw.:"
111   §  §  ¨¹¹ token = st.nextToken(); //Enthält den Wert Crossoverw.
112   §  §  ¨¹¹ daten.crossover = Double.parseDouble(token);
113   §  §  §
114   §  §  ¨¹¹ token = st.nextToken(); //Enthält den String "Mutationw.:"
115   §  §  ¨¹¹ token = st.nextToken(); //Enthält den Wert Mutation
116   §  §  ¨¹¹ daten.mutation = Double.parseDouble(token);
117   §  §  §
118   §  §  ¨¹¹ line = nextLine(); //Enthält keine Daten
119   §  §  ¨¹¹ line = nextLine(); //Enthält keine Daten
120   §  §  §
121   §  §  §/*Einlesen der Kosten und der Fitness der einzelnen Generationen*/
122   §  §  ¨¹¹ daten.kostsFitness = new double[daten.generation][2];
123   §  §  ¨¹¹±for(int i = 0; i < daten.generation; i++)
124   §  §  §  5{
125   §  §  §  7¹¹ line = nextLine(); //Zeile der 0-ten Generation
126   §  §  §  7¹¹ st = new StringTokenizer(line);
127   §  §  §  7¹¹ token = st.nextToken(); //Enthält die aktuelle Generation
128   §  §  §  5
129   §  §  §  7¹¹±while(token.lastIndexOf(']') == -1) //Auslesen der aktuellen Generation
130   §  §  §  5  5{
131   §  §  §  5  7¹¹ token = token + st.nextToken();
132   §  §  §  5  °}
133   §  §  §  7¹¹ daten.bestIndividuum = token;
134   §  §  §  5
135   §  §  §  7¹¹ token = st.nextToken(); //Enthält den Wert Kosten
136   §  §  §  7¹¹ daten.kostsFitness[i][0] = Double.parseDouble(token);
137   §  §  §  5
138   §  §  §  7¹¹ token = st.nextToken(); //Enthält den Wert Fitness
139   §  §  §  7¹¹ daten.kostsFitness[i][1] = Double.parseDouble(token);
140   §  §  §  °}
141   §  §  §
142   §  §  ¨¹¹ line = nextLine(); //Enthält keine Daten
143   §  §  ¨¹¹ line = nextLine();
144   §  §  §
145   §  §  ¨¹¹ st = new StringTokenizer(line);
146   §  §  ¨¹¹ token = st.nextToken(); //Enthält den String "Beste"
147   §  §  ¨¹¹ token = st.nextToken(); //Enthält den String "Fitness:"
148   §  §  ¨¹¹ token = st.nextToken(); //Enthält den Wert Beste Fitness
149   §  §  ¨¹¹ daten.bestFitness = Double.parseDouble(token);
150   §  §  §
151   §  §  ¨¹¹ line = nextLine(); //Letzte Zeile einer Messung einlesen
152   §  §  §                     //Enhält keine Daten
153   §Â¹Ä  ¨¹¹ return true;
154   §  §  §}
      §  §  § ¬¹¹¹¹¹¹¹¹¹
155   §  §  §ðîì catch(Exception e)
      §  §  § ªË¹¹¹¹¹¹¹¹
156   §  §  §  §{
157   §  §  §  ¨¹¹ p("Fehler in nextMeasure bzw. keine Daten mehr! \n" + e);
158   §  §  §  ¨¹¹ p("nextMeasure()");
159   §Â¹Ä  §  ¨¹¹ return false;
160   §  §  ©  ©}
161   §  §
162   §  ©}  //nextMeasure()
163   §
164   §/**
```

```
§/*Faßt jeweils n Messungen mit gleiche
§/*Parametern zu einem Eintrag zusammen
§*/
§ ¬¹¹¹¹¹¹¹¹¹
§Þßàpublic boolean summarizeMeasure(int n)
§ ªË¹¹¹¹¹¹¹¹
§  §{
§  ¨¹¹Í Data[] sDaten = new Data[n];
§  ¨¹¹Í double sumKosten = 0.0;
§  ¨¹¹Í double sumFitness = 0.0;
§  ¨¹¹Í double varianzKosten = 0.0;
§  ¨¹¹Í double varianzFitness = 0.0;
§  §
§  ¨¹¹Í double bestFitness = 0.0;
§  §   //Einlesen der n Meßreihen
§  ¨¹¹±for(int i = 0; i < n; i++)
§  §  5{
§  §  7¹³´if(nextMeasure())
§  §  5 6§{
§  §  5 6¨¹¹ sDaten[i] = new Data();
§  §  5 6¨¹¹ sDaten[i].kostsFitness = new double[daten.generation][2];
§  §  5 6§
§  §  5 6¨¹¹ sDaten[i].generation = daten.generation;
§  §  5 6¨¹¹ sDaten[i].population = daten.population;
§  §  5 6¨¹¹ sDaten[i].crossover = daten.crossover;
§  §  5 6¾¹¹ sDaten[i].mutation = daten.mutation;
§  §  5 6 }
§  §  5 ¶´else
§  §  5 ¸§{
§  §  5 ¸¨¹¹ p("\nsummarizeMeasure()");
§  §  5 ¸¨¹¹ p("Generation bis: " + daten.generation + " Population bis: " + daten.population);
§Â¹Ä  5 ¸¾¹¹ return false;
§  §  5 È }
§  §  7¹¹±for(int j = 0; j < daten.generation; j++)
§  §  5  5{
§  §  5  7¹¹ sDaten[i].kostsFitness[j][0] = daten.kostsFitness[j][0];
§  §  5  7¹¹ sDaten[i].kostsFitness[j][1] = daten.kostsFitness[j][1];
§  §  5  °}
§  §  °}
§  §
§  §   //Berechnen der Mittelwerte
§  ¨¹¹ gemitteltenDaten.kostsFitness = new double[sDaten[0].generation][2];
§  ¨¹¹ gemitteltenDaten.varianz = new double[sDaten[0].generation][2];
§  ¨¹¹±for(int i = 0; i < sDaten[0].generation; i++)
§  §  5{
§  §  7¹¹±for(int j = 0; j < n; j++)
§  §  5  5{
§  §  5  7¹¹ sumKosten = sumKosten + sDaten[j].kostsFitness[i][0];
§  §  5  7¹¹ sumFitness = sumFitness + sDaten[j].kostsFitness[i][1];
§  §  5  °}
§  §  7¹¹ sumKosten = sumKosten / n;
§  §  7¹¹ sumFitness = sumFitness / n;
§  §  7¹¹ gemitteltenDaten.kostsFitness[i][0] = sumKosten;
§  §  7¹¹ gemitteltenDaten.kostsFitness[i][1] = sumFitness;
§  §  5   //Berechnung der Varianz
§  §  7¹¹±for(int j = 0; j < n; j++)
§  §  5  5{
§  §  5  7¹¹ varianzKosten = (sumKosten - sDaten[j].kostsFitness[i][0])*
§  §  5  5         (sumKosten - sDaten[j].kostsFitness[i][0]);
§  §  5  7¹¹ varianzFitness = (sumFitness - sDaten[j].kostsFitness[i][1]) *
§  §  5  5         (sumFitness - sDaten[j].kostsFitness[i][1]);
§  §  5  °}
§  §  7¹¹ varianzKosten = varianzKosten / n-1;
§  §  7¹¹ varianzFitness = varianzFitness / n-1;
§  §  7¹¹ gemitteltenDaten.varianz[i][0] = varianzKosten;
§  §  7¹¹ gemitteltenDaten.varianz[i][1] = varianzFitness;
§  §  5
§  §  7¹¹ sumKosten = 0.0;
§  §  7¹¹ sumFitness = 0.0;
§  §  7¹¹ varianzKosten = 0.0;
§  §  7¹¹ varianzFitness = 0.0;
§  §  °}
§  ¨¹¹ gemitteltenDaten.generation = sDaten[0].generation;
§  ¨¹¹ gemitteltenDaten.population = sDaten[0].population;
§  ¨¹¹ gemitteltenDaten.crossover = sDaten[0].crossover;
§  ¨¹¹ gemitteltenDaten.mutation = sDaten[0].mutation;
§  §
§  §   //Suchen der besten (gemittelten) Fitness aus gemitteltenDaten
§  ¨¹¹±for(int i = 0; i < gemitteltenDaten.generation; i++)
§  §  5{
§  §  7¹³´if(bestFitness < gemitteltenDaten.kostsFitness[i][1])
§  §  5 6§{
§  §  5 6¨¹¹ gemitteltenDaten.bestFitness = gemitteltenDaten.kostsFitness[i][1];
§  §  5 6¾¹¹ bestFitness = gemitteltenDaten.bestFitness;
§  §  5 ¶ }
§  §  °}
§Â¹Ä¹¹ return true;
§  ©}  //summarizeMeasure
§
§
§/**
```

```
254     §/*Hier werden die Daten zusammengefaßt und
255     §/*Excel-lesbar wieder gespeichert
256     §*/
        § ¬¹¹¹¹¹¹¹¹¹
257     §Þßà public void writeSumData(int n)
        § ªË¹¹¹¹¹¹¹¹
258     §   §{
259     §   ¨¹¹í SumData sumTemp = new SumData();
260     §   §
261     §   §    //Variablen für Zeitmessung
262     §   ¨¹¹í long time1 = 0;
263     §   ¨¹¹í long time2 = 0;
264     §   §
265     §   ¨¹¹Í String printString = "";
266     §   §
267     §   ¨¹¹¹ p("Beginne mit dem Zusammenfassen der Daten: ");
268     §   ¨¹¹¹ time1 = System.currentTimeMillis();
269     §   ¨¹¹í int j = 0;
270     §   ¨¹¹¹±while(summarizeMeasure(n))
271     §   §   5{
272     §   §   7¹¹¹ sumTemp = new SumData();
273     §   §   7¹¹¹ sumTemp.generation = gemitteltеDaten.generation;
274     §   §   7¹¹¹ sumTemp.population = gemitteltеDaten.population;
275     §   §   7¹¹¹ sumTemp.crossover = gemitteltеDaten.crossover;
276     §   §   7¹¹¹ sumTemp.mutation = gemitteltеDaten.mutation;
277     §   §   7¹¹¹ sumTemp.bestFitness = gemitteltеDaten.bestFitness;
278     §   §   7¹¹¹ v.addElement(sumTemp); //Sind die Daten geschrieben, werden sie an den Vector angehängt
279     §   §   7¹¹¹ j++;
280     §   §   7¹³´if((j % 50) == 0)
281     §   §   5 ¶¾¹¹¹ p("Generation: " + gemitteltеDaten.generation + " Population: " + gemitteltеDaten.population);
282     §   §   5
283     §   §   °}
284     §   ¨¹¹¹ time2 = System.currentTimeMillis();
285     §   ¨¹¹¹ p("Zusammenfassen der Daten beendet:");
286     §   ¨¹¹¹ p("Zeit: " + (time2 - time1) + " Vektorlaenge: " + v.size() + "\n");
287     §   §
288     §   ¨¹¹¹´try
289     §   §   §{
290     §   §   ¨¹¹¹ p("Beginne mit dem Schreiben der Daten: ");
291     §   §   ¨¹¹¹ time1 = System.currentTimeMillis();
292     §   §   §
293     §   §   ¨¹¹¹ fileWriter = new FileWriter(fileOutput);
294     §   §   ¨¹¹¹ fileOut = new PrintWriter(fileWriter);
295     §   §   §
296     §   §   ¨¹¹¹±for(int i = 0; i < v.size(); )
297     §   §   §   5{
298     §   §   §   7¹¹¹ sumTemp = (SumData)v.elementAt(i);
299     §   §   §   7¹¹¹ printString = "Generationen: " + sumTemp.generation + " Population: " + sumTemp.population + "\n";
300     §   §   §   7¹¹¹ System.out.print(printString); //zur Vortschrittskontrolle wird dieser String ausgegeben
301     §   §   §   7¹¹¹ printString = printString + " 0,01  0,02  0,03  0,04  0,05  0,06  0,07  0,08  0,09  0,1\n";
302     §   §   §   §   5
303     §   §   §   §   5    /*Schreiben einer zusammengefaßten Messung*/
304     §   §   §   §   5    //Crossover
305     §   §   §   7¹¹¹±for(int k = 0; k < 5; k++)
306     §   §   §   §   5  5{
307     §   §   §   §   5  7¹¹¹ printString = printString + (0.2 + k * 0.1) + " ";
308     §   §   §   §   5  5    //Mutation
309     §   §   §   §   5  7¹¹¹±for(int m = 0; m < 10; m++)
310     §   §   §   §   5  5  5{
311     §   §   §   §   5  5  7¹¹¹ sumTemp = (SumData)v.elementAt(i);
312     §   §   §   §   5  5  7¹¹¹ printString = printString + sumTemp.bestFitness + "   ";
313     §   §   §   §   5  5  7¹¹¹ i++;
314     §   §   §   §   5  5  °}
315     §   §   §   §   5  7¹¹¹ printString = printString + "\n";
316     §   §   §   §   5  °}
317     §   §   §   §   7¹¹¹ fileOut.println(printString);
318     §   §   §   §   °}
319     §   §   §   §
320     §   §   ¨¹¹¹ fileOut.close();
321     §   §   §
322     §   §   ¨¹¹¹ time2 = System.currentTimeMillis();
323     §   §   ¨¹¹¹ p("Schreibvorgang erfolgreich beendet ");
324     §   §   ¨¹¹¹ p("Benoetigte Zeit: " + (time2 - time1));
325     §   §   §}
        §   §   § ¬¹¹¹¹¹¹¹¹¹
326     §   §   §ðîì catch(Exception e)
        §   §   § ªË¹¹¹¹¹¹¹¹
327     §   §   §   §{
328     §   §   §   ¨¹¹¹ fileOut.close();
329     §   §   §   ¨¹¹¹ time2 = System.currentTimeMillis();
330     §   §   §   ¨¹¹¹ p("Schreibvorgang unvollstaendig beendet ");
331     §   §   §   ¨¹¹¹ p("Benoetigte Zeit: " + (time2 - time1));
332     §   §   ©   ©}
333     §   §
334     §   ©}
335     §
336     §/**
337     §/*Gibt die Datei aus, die bei der Instanziierung des
338     §/*Objectes angegeben wurde
339     §*/
        § ¬¹¹¹¹¹¹¹¹¹
```

```
340   §Þßà public void printFile()
      § ªË¹¹¹¹¹¹¹¹
341   §  §{
342   §  ¨¹¹´try
343   §  §  §{
344   §  §  ¨¹¹í String s, s2 = new String();
345   §  §  ¨¹¹±while((s = input.readLine())!= null)
346   §  §  §  Ð¹¹ s2 += s + "\n";
347   §  §  ¨¹¹ input.close();
348   §  §  ¨¹¹ p(s2);
349   §  §  §}
      §  §  § ¬¹¹¹¹¹¹¹¹
350   §  §  §ðîìcatch(Exception e)
      §  §  § ªË¹¹¹¹¹¹¹¹
351   §  §  §  §{
352   §  §  §  ¨¹¹ p("" + e);
353   §  §  ©  ©}
354   §  ©}
355   ©}
356
```