```
1  /**************************************************************************
2  ************************ Fordeling - formål ************************
3  **************************************************************************
4
5  Klassen tar i bruk de ressurser og behov som er lest inn i klassene
6  Fag, Laerer og Aarstrinn og fordeler disse så godt det lar seg gjøre.
7
8  Deretter lagres resultatet i underklasser per trinn og fag for hvert
9  trinn. Dette resultatet kan så returneres til main med egne metoder.
10
11 Kodet av Andreas Neverdahl
12
13 **************************************************************************
14 ************************ Oppbygging av klassen ************************
15 **************************************************************************
16
17 Fordeling                    - Opprettes i main. Se formål.
18 .fordelLaerere()             - Kalles i main. Kjører ressursfordeling.
19 .ledigLaerer()               - Brukes av fordelLaerere().
20 .trinnplan()                 - Kalles i main. Genererer oversikt for
21                                trinn.
22 .finnTrinnLaerere()          - Brukes av fordelLaerere().
23     underklasse Trinn        - Opprettes av super for lagring av
24                                fordeling.
25         underklasse Faginfo - Samme som Trinn.
26         .leggTilLaerer()    - Brukes av fordelLaerere(). Lagrer
           lærere.
27
28 **************************************************************************/
29
30 import java.util.ArrayList;
31 import javax.swing.JOptionPane;
32 import javax.swing.JTextArea;
33
34 public class Fordeling
35 {
36     // Datafelt
37     private int ressursFoerFordeling = 0;
38     private int skoleBehov = 0;
39     private ArrayList<Trinn> trinn = new ArrayList<Trinn>();
40
41
42     // Konstruktører
43
44     public Fordeling(Fag[] fag, Aarstrinn[] aarstrinn, Laerer[]
       laerer)
45     {
46         for ( int i = 0 ; i < aarstrinn.length ; i++ )
47         {
48             trinn.add(new Trinn(aarstrinn[i], fag));
49
50             for ( int j = 0 ; j < fag.length ; j++)
51             {
52                 skoleBehov += aarstrinn[i].getTimer(j);
53             }
54         }
55
56         for ( int i = 0 ; i < laerer.length ; i++ )
57         {
58             ressursFoerFordeling += laerer[i].getTilgjengeligeTimer();
59         }
60     }
61
62     // Metoder
63
64     public void fordelLaerere(
65         Fag[] fag, Aarstrinn[] aarstrinn, Laerer[] laerer)
```

```
66
67      /*
68      Metoden for selve fordelinga.
69
70      Prioriterer lærere med fordypning innenfor hvert fag,
71      men bruker lærere uten fordypning der det trengs.
72      */
73
74      {
75          if (skoleBehov > ressursFoerFordeling)
76          {
77              JOptionPane.showMessageDialog(
78                  null,
79                  "Innlest timebehov er større enn tilgjengelige " +
80                  "lærerressurser!" +
81                  "\n\nTimebehov: " + skoleBehov +
82                  "\nTilgjengelig: " + ressursFoerFordeling +
83                  "\n\nProgrammet vil nå avslutte.",
84                  "Feil innverdier!", JOptionPane.ERROR_MESSAGE );
85              System.exit(0);
86          }
87          else
88          {
89              // For hvert fag skolen tilbyr
90              for ( int i = 0 ; i < fag.length ; i++)
91              {
92                  boolean ledigFordypning = true;
93                  // For hvert innlest trinn
94                  for ( int j = 0 ; j < aarstrinn.length ; j++ )
95                  {
96                      // Hvor mange timer som er bundet til faget
97                      int timerBundet = 0;
98                      int behov = aarstrinn[j].getTimer(i);
99
100                     // Kjører fordeling til behovet til fag
101                     // i på trinn j er møtt
102                     while ( behov > timerBundet )
103                     {
104                         if (ledigFordypning)
105                             // Finner lærer med flest ledige
                                fordypningstimer
106                             fag[i].tilgjengeligLaerer(laerer);
107
108                         // Setter denne læreren som førstevalg
109                         int aktLaerer = fag[i].getLaererIndeks();
110                         // Midlertidig verdi av bundet tid hos
                            aktLaerer
111                         int timerBundetLaerer = 0;
112                         int tilgjengeligeTimer =
113                             laerer[aktLaerer].getTilgjengeligeTimer();
114
115                         // Dersom aktuell lærer har fordypningstimer
116                         // igjen
117                         if ( tilgjengeligeTimer > 0 )
118                         {
119                             if ( tilgjengeligeTimer > behov )
120                             {
121                                 timerBundetLaerer = behov;
122                                 timerBundet = timerBundetLaerer;
123                             }
124                             else
125                             {
126                                 timerBundetLaerer =
                                    tilgjengeligeTimer;
127                                 timerBundet += timerBundetLaerer;
128                             }
129                         }
```

```
130                         else
131                         // Ikke mer fordypningstimer igjen.
132                         // Bruker annen laerer.
133                         {
134                             // Er false helt til neste fag skal
                                fordeles.
135                             ledigFordypning = false;
136                             aktLaerer = ledigLaerer(laerer);
137                             tilgjengeligeTimer =
138                                 laerer[aktLaerer].
                                    getTilgjengeligeTimer();
139
140                             if ( tilgjengeligeTimer > behov )
141                             {
142                                 timerBundetLaerer = behov;
143                                 timerBundet = timerBundetLaerer;
144                             }
145                             else
146                             {
147                                 timerBundetLaerer =
148                                     tilgjengeligeTimer;
149                                 timerBundet += timerBundetLaerer;
150                             }
151                         }
152                         // Trekker bundet tid fra potten til gjeldende
153                         // laerer
154                         laerer[aktLaerer].setTilgjengeligeTimer(
155                             timerBundetLaerer);
156
157                         // Registrerer lærer og antall timer i
158                         // resultatsobjektene
159                         this.trinn.get(j).faginfo.get(i).leggTilLaerer
                            (
160                             aktLaerer, timerBundetLaerer);
161
162
163                     }   // Fordeling Aarstrinn[j]-loekke
164
165                 }   // Aarstrinn[]-loekke
166
167             }   // Fag[]-loekke
168
169             this.finnTrinnLaerere();
170         }
171
172     }   // fordelLaerere()
173
174     private int ledigLaerer(Laerer[] laerer)
175     {
176         /*
177         Returnerer indeks til lærer i angitt array med flest ledige
178         timer
179         */
180
181         int ledigIndeks = 0;
182         int flestTimer = 0;
183         for (int i = 0; i < laerer.length ; i++)
184         {
185             if (laerer[i].getTilgjengeligeTimer() > flestTimer)
186             {
187                 flestTimer = laerer[i].getTilgjengeligeTimer();
188                 ledigIndeks = i;
189             }
190         }
191
192         return ledigIndeks;
193
```

```
194      }   // ledigLaerer()
195
196      public JTextArea trinnPlan(Laerer[] laerer)
197      {
198          /*
199          Returnerer et JTextArea-objekt med trinnvis oversikt over hvor
200          mange timer hver lærer har i hvert fag.
201          */
202
203          JTextArea txt = new JTextArea();
204
205          for (int i = 0 ; i < this.trinn.size() ; i++)
206          {
207              txt.append( "Oversikt over lærertimer på " +
208                  this.trinn.get(i).navn + ". trinn:\n" +
209                  "Fag\tLærer\ttimer");
210
211              for (int j = 0; j < this.trinn.get(i).faginfo.size(); j++)
212              {
213                  // Sjekker at gjeldende fag er aktuelt på trinnet
214                  if (this.trinn.get(i).faginfo.get(j).behov > 0)
215                  {
216                      txt.append( "\n" + this.trinn.get(i).faginfo.
217                          get(j).navn + ": \t");
218
219                      for (int k = 0; k < this.trinn.get(i).faginfo.
220                          get(j).laererIndeks.size(); k++)
221                      {
222                          // Hopper ned en linje med innrykk dersom
                             flere
223                          // lærere har samme fag
224                          if (k > 0)
225                              txt.append("\n\t" + laerer[trinn.get(i).
226                              faginfo.get(j).laererIndeks.get(k)].
227                              getLaererNavn() + "\t" +
228                              this.trinn.get(i).faginfo.get
229                              (j).laererTimer.get(k));
230                          else
231                              txt.append(laerer[trinn.get(i).faginfo.get
232                              (j).laererIndeks.get(k)].getLaererNavn() +
233                              "\t" + this.trinn.get(i).faginfo.get(j).
234                              laererTimer.get(k));
235                      }
236                  }
237              }
238              txt.append("\n\n");
239          }
240
241          return txt;
242      }
243
244      public JTextArea laererRessursEtterFordeling(Laerer[] laerer)
245      {
246          /*
247          Returnerer et JTextArea-objekt med oversikt over gjenværende
248          lærerressurser.
249          */
250
251          JTextArea txt = new JTextArea();
252          txt.setText(
253              "Gjennværende lærerressurser etter fordeling på fag:\n\n"+
254              "Navn\tLedige timer\tSpesielle oppgaver\n");
255          int restLaererTimer = 0;
256
257          for (int i = 0 ; i < laerer.length ; i++ )
258          {
259              restLaererTimer += laerer[i].getTilgjengeligeTimer();
```

```
260             txt.append( laerer[i].getLaererNavn() + ":\t" + laerer[i].
261                 getTilgjengeligeTimer() + "\t");
262
263             String spesOppg = "";
264             for (int j = 0 ; j < 3 ; j++ )
265             {
266                 if (!spesOppg.isEmpty() && !laerer[i].
267                     getSpesielleOppgaver(j).isEmpty())
268
269                     spesOppg += ", " + laerer[i].
270                         getSpesielleOppgaver(j);
271                 else
272                     spesOppg += laerer[i].getSpesielleOppgaver(j);
273             }
274             if (spesOppg.isEmpty())
275                 spesOppg = "Nei";
276             txt.append(spesOppg + "\n");
277         }
278         txt.append(
279             "\nTotalt antall timer tilgjengelig før fordeling:\t" +
280             ressursFoerFordeling +
281             "\nSkolens timebehov:\t\t" + skoleBehov +
282             "\nResterende antall timer tilgjengelig:\t" +
283             restLaererTimer);
284
285         return txt;
286     }
287
288     public void finnTrinnLaerere()
289     {
290         /*
291         Fyller arraylists med trinnlærere ut fra lærere i
            trinn.faginfo
292         */
293
294         for ( int i = 0; i < this.trinn.size(); i++ )
295         {
296             this.trinn.get(i).laererIndeks.clear();
297
298             for ( int j = 0; j < this.trinn.get(i).faginfo.size();
                j++)
299             {
300                 for ( int k = 0 ; k <
                    this.trinn.get(i).faginfo.get(j).
301                     laererIndeks.size() ; k++ )
302                 {
303                     if (!this.trinn.get(i).laererIndeks.contains(
304                         this.trinn.get(i).faginfo.get(j).laererIndeks.
305                         get(k)))
306                         this.trinn.get(i).laererIndeks.add(
307                             this.trinn.get(i).faginfo.get(j).
308                             laererIndeks.get(k));
309                 }
310             }
311         }
312     }
313
314     // Underklasser
315
316     public class Trinn
317     {
318         // Datafelt
319
320         private String navn; // 1. klasse/1/Vg1 e.l.
321         private ArrayList<Integer> laererIndeks =
322             new ArrayList<Integer>();
323         private ArrayList<FagInfo> faginfo =
```

```
324             new ArrayList<FagInfo>();
325
326         // Konstruktoer
327
328         public Trinn( Aarstrinn aarstrinn, Fag[] fag )
329         {
330             navn = aarstrinn.getTrinn();
331             for ( int i = 0 ; i < fag.length ; i++ )
332             {
333                 faginfo.add(new FagInfo( i, aarstrinn, fag[i]));
334             }
335         }
336
337         // Underklasser
338
339         public class FagInfo
340         {
341             // Datafelt
342             private String navn;
343             private int behov;
344             private ArrayList<Integer> laererIndeks =
345                 new ArrayList<Integer>();
346             private ArrayList<Integer> laererTimer =
347                 new ArrayList<Integer>();
348
349             // Konstruktoer
350             public FagInfo(
351                 int fagIndeks, Aarstrinn aarstrinn, Fag fag )
352             {
353                 navn = fag.getFagNavn();
354                 behov = aarstrinn.getTimer(fagIndeks);
355             }
356
357             // Metoder
358             public void leggTilLaerer(int indeks, int timer)
359             {
360                 laererIndeks.add(indeks);
361                 laererTimer.add(timer);
362             }
363
364
365         } // class fag
366
367     } // class trinn
368
369 }
```