```
1   /*
2    * Proyecto: SolucionLaboratorio08_2023_1
3    * Archivo:  Alumno.h
4    * Autor:    J. Miguel Guanira E.
5    *
6    * Created on 18 de octubre de 2023, 08:53 PM
7    */
8
9
10  #ifndef ALUMNO_H
11  #define ALUMNO_H
12
13  class Alumno {
14  private:
15      int codigo;
16      char *nombre;
17      int escala ;
18      double total;
19  public:
20      Alumno();
21      virtual ~Alumno();
22      void SetTotal(double total);
23      double GetTotal() const;
24      void SetEscala(int escala);
25      int GetEscala() const;
26      void SetNombre(const char* nombre);
27      void GetNombre(char*) const;
28      void SetCodigo(int codigo);
29      int GetCodigo() const;
30      void leerDatos(ifstream &);
31      void imprime(ofstream&);
32  };
33
34  #endif /* ALUMNO_H */
35
36  /*
37   * Proyecto: SolucionLaboratorio08_2023_1
38   * Archivo:  Alumno.cpp
39   * Autor:    J. Miguel Guanira E.
40   *
41   * Created on 18 de octubre de 2023, 08:53 PM
42   */
43
44  #include <iostream>
45  #include <fstream>
46  #include <iomanip>
47  using namespace std;
48  #include <cstring>
49  #include "Alumno.h"
50
51  Alumno::Alumno() {
52      nombre = nullptr;
53  }
54
55  Alumno::~Alumno() {
56      if(nombre!=nullptr) delete nombre;
57  }
58
59  void Alumno::SetTotal(double total) {
60      this->total = total;
61  }
62
63  double Alumno::GetTotal() const {
64      return total;
65  }
66
```

```
67  void Alumno::SetEscala(int escala) {
68      this->escala = escala;
69  }
70
71  int Alumno::GetEscala() const {
72      return escala;
73  }
74
75  void Alumno::GetNombre(char*cad) const {
76      if(nombre==nullptr) cad[0]=0;
77      else strcpy(cad,nombre);
78  }
79
80  void Alumno::SetNombre(const char* cad) {
81      if(nombre!=nullptr) delete nombre;
82      nombre = new char[strlen(cad)+1];
83      strcpy(nombre,cad);
84  }
85
86  void Alumno::SetCodigo(int codigo) {
87      this->codigo = codigo;
88  }
89
90  int Alumno::GetCodigo() const {
91      return codigo;
92  }
93
94  void Alumno::leerDatos(ifstream &arch) {
95      char nomb[60];
96      arch>>codigo;
97      if(arch.eof())return;
98      arch.get();
99      arch.getline(nomb,60,',');
100     SetNombre(nomb);
101     arch>>escala;
102     arch.get();
103 }
104
105 void Alumno::imprime(ofstream&arch) {
106     arch<<left<<setw(10)<<codigo<<setw(40)<<nombre<<right<<setw(3)<<escala;
107 }
108
109 /*
110  * Proyecto: SolucionLaboratorio08_2023_1
111  * Archivo:  Presencial.h
112  * Autor:    J. Miguel Guanira E.
113  *
114  * Created on 18 de octubre de 2023, 09:11 PM
115  */
116
117
118 #ifndef PRESENCIAL_H
119 #define PRESENCIAL_H
120 #include <fstream>
121 using namespace std;
122 #include "Alumno.h"
123
124 class Presencial : public Alumno{
125 private:
126     double recargo;
127     double total;
128 public:
129     void SetTotal(double total);
130     double GetTotal() const;
131     void SetRecargo(double recargo);
132     double GetRecargo() const;
```

```
133         void leerDatos(ifstream&);
134         void actualiza(double);
135         void imprime(ofstream&);
136     };
137
138     #endif /* PRESENCIAL_H */
139
140     /*
141      * Proyecto: SolucionLaboratorio08_2023_1
142      * Archivo:  Presencial.cpp
143      * Autor:    J. Miguel Guanira E.
144      *
145      * Created on 18 de octubre de 2023, 09:11 PM
146      */
147
148     #include <iostream>
149     #include <iomanip>
150     using namespace std;
151
152     #include "Presencial.h"
153
154     void Presencial::SetTotal(double total) {
155         this->total = total;
156     }
157
158     double Presencial::GetTotal() const {
159         return total;
160     }
161
162     void Presencial::SetRecargo(double recargo) {
163         this->recargo = recargo;
164     }
165
166     double Presencial::GetRecargo() const {
167         return recargo;
168     }
169
170     void Presencial::leerDatos(ifstream &arch) {
171         Alumno::leerDatos(arch);
172         arch>>recargo;
173         arch.get(); // Lee el cambio de línea
174     }
175
176     void Presencial::actualiza(double monto) {
177         total = monto*GetRecargo()/100;
178         Alumno::SetTotal(monto+total);
179     }
180
181     void Presencial::imprime(ofstream&arch) {
182         Alumno::imprime(arch);
183         arch<<right<<setw(15)<<"+"<<setw(10)<<Alumno::GetTotal()<<endl;
184     }
185
186     /*
187      * Proyecto: SolucionLaboratorio08_2023_1
188      * Archivo:  Semipresencial.h
189      * Autor:    J. Miguel Guanira E.
190      *
191      * Created on 18 de octubre de 2023, 09:15 PM
192      */
193
194
195     #ifndef SEMIPRESENCIAL_H
196     #define SEMIPRESENCIAL_H
197     #include <fstream>
198     using namespace std;
```

```
199  #include "Alumno.h"
200
201  class Semipresencial: public Alumno {
202  private:
203      double descuento;
204      double total;
205  public:
206      void SetTotal(double total);
207      double GetTotal() const;
208      void SetDescuento(double descuento);
209      double GetDescuento() const;
210      void leerDatos(ifstream&);
211      void actualiza(double);
212      void imprime(ofstream&);
213  };
214
215  #endif /* SEMIPRESENCIAL_H */
216
217  /*
218   * Proyecto: SolucionLaboratorio08_2023_1
219   * Archivo:  Semipresencial.cpp
220   * Autor:    J. Miguel Guanira E.
221   *
222   * Created on 18 de octubre de 2023, 09:15 PM
223   */
224
225  #include <iostream>
226  #include <iomanip>
227  using namespace std;
228
229  #include "Semipresencial.h"
230
231  void Semipresencial::SetTotal(double total) {
232      this->total = total;
233  }
234
235  double Semipresencial::GetTotal() const {
236      return total;
237  }
238
239  void Semipresencial::SetDescuento(double descuento) {
240      this->descuento = descuento;
241  }
242
243  double Semipresencial::GetDescuento() const {
244      return descuento;
245  }
246
247  void Semipresencial::leerDatos(ifstream&arch) {
248      Alumno::leerDatos(arch);
249      arch>>descuento;
250      arch.get();
251  }
252
253  void Semipresencial::actualiza(double monto) {
254      total = monto*GetDescuento()/100;
255      Alumno::SetTotal(monto-total);
256  }
257
258  void Semipresencial::imprime(ofstream&arch) {
259      Alumno::imprime(arch);
260      arch<<right<<setw(15)<<"-"<<setw(10)<<Alumno::GetTotal()<<endl;
261  }
262
263  /*
264   * Proyecto: SolucionLaboratorio08_2023_1
```

```
265     * Archivo:  Virtual.h
266     * Autor:    J. Miguel Guanira E.
267     *
268     * Created on 18 de octubre de 2023, 09:17 PM
269     */
270
271
272   #ifndef VIRTUAL_H
273   #define VIRTUAL_H
274   #include <fstream>
275   using namespace std;
276   #include "Alumno.h"
277
278   class Virtual: public Alumno {
279   private:
280       char *licencia;
281       double total;
282   public:
283       Virtual();
284       virtual ~Virtual();
285       void SetTotal(double total);
286       double GetTotal() const;
287       void SetLicencia(const char*);
288       void GetLicencia(char*) const;
289       void leerDatos(ifstream&);
290       void actualiza(double);
291       void imprime(ofstream&);
292   };
293
294   #endif /* VIRTUAL_H */
295
296   /*
297    * Proyecto: SolucionLaboratorio08_2023_1
298    * Archivo:  Virtual.cpp
299    * Autor:    J. Miguel Guanira E.
300    *
301    * Created on 18 de octubre de 2023, 09:17 PM
302    */
303
304   #include <iostream>
305   #include <iomanip>
306   using namespace std;
307   #include <cstring>
308   #include "Virtual.h"
309
310   Virtual::Virtual() {
311       licencia = nullptr;
312       total = 100;
313   }
314
315   Virtual::~Virtual() {
316       if(licencia!=nullptr) delete licencia;
317   }
318
319   void Virtual::SetTotal(double total) {
320       this->total = total;
321   }
322
323   double Virtual::GetTotal() const {
324       return total;
325   }
326
327   void Virtual::SetLicencia(const char* cad) {
328       if(licencia!=nullptr) delete licencia;
329       licencia = new char[strlen(cad)+1];
330       strcpy(licencia,cad);
```

```
331 }
332
333 void Virtual::GetLicencia(char*cad) const {
334     if(licencia==nullptr) cad[0]=0;
335     else strcpy(cad,licencia);
336 }
337
338 void Virtual::leerDatos(ifstream&arch) {
339     char lic[30];
340     Alumno::leerDatos(arch);
341     arch.getline(lic,30);
342     SetLicencia(lic);
343 }
344
345 void Virtual::actualiza(double monto) {
346     Alumno::SetTotal(monto+total);
347 }
348
349 void Virtual::imprime(ofstream&arch) {
350     Alumno::imprime(arch);
351     arch<<right<<setw(15)<<licencia<<setw(10)<<Alumno::GetTotal()<<endl;
352 }
353
354 /*
355  * Proyecto: SolucionLaboratorio08_2023_1
356  * Archivo:  Escala.h
357  * Autor:    J. Miguel Guanira E.
358  *
359  * Created on 18 de octubre de 2023, 09:07 PM
360  */
361
362
363 #ifndef ESCALA_H
364 #define ESCALA_H
365
366 class Escala {
367 private:
368     int codigo;
369     double precio;
370 public:
371     void SetPrecio(double precio);
372     double GetPrecio() const;
373     void SetCodigo(int codigo);
374     int GetCodigo() const;
375 };
376
377 #endif /* ESCALA_H */
378
379 /*
380  * Proyecto: SolucionLaboratorio08_2023_1
381  * Archivo:  Escala.cpp
382  * Autor:    J. Miguel Guanira E.
383  *
384  * Created on 18 de octubre de 2023, 09:07 PM
385  */
386
387 #include <iostream>
388 #include <iomanip>
389 using namespace std;
390
391 #include "Escala.h"
392
393 void Escala::SetPrecio(double precio) {
394     this->precio = precio;
395 }
396
```

```
397  double Escala::GetPrecio() const {
398      return precio;
399  }
400
401  void Escala::SetCodigo(int codigo) {
402      this->codigo = codigo;
403  }
404
405  int Escala::GetCodigo() const {
406      return codigo;
407  }
408
409  /*
410   * Proyecto: SolucionLaboratorio08_2023_1
411   * Archivo:  Tesoreria.h
412   * Autor:    J. Miguel Guanira E.
413   *
414   * Created on 18 de octubre de 2023, 09:26 PM
415   */
416
417
418  #ifndef TESORERIA_H
419  #define TESORERIA_H
420  #include "Presencial.h"
421  #include "Semipresencial.h"
422  #include "Virtual.h"
423  #include "Escala.h"
424  class Tesoreria {
425  private:
426      class Presencial lpresencial[100];
427      class Semipresencial lsemipresencial[100];
428      class Virtual lvirtual[100];
429      class Escala lescala[10];
430  public:
431      Tesoreria();
432      void cargaescalas(const char *);
433      void cargaalumnos(const char *);
434      void actualiza(double);
435      void imprime(const char *);
436      void muestraEscala(int);
437      void imprimeLinea(ofstream &, char, int);
438  };
439
440  #endif /* TESORERIA_H */
441
442  /*
443   * Proyecto: SolucionLaboratorio08_2023_1
444   * Archivo:  Tesoreria.cpp
445   * Autor:    J. Miguel Guanira E.
446   *
447   * Created on 18 de octubre de 2023, 09:26 PM
448   */
449
450  #include <iostream>
451  #include <fstream>
452  #include <iomanip>
453  using namespace std;
454
455  #include "Tesoreria.h"
456
457  Tesoreria::Tesoreria() {
458      for(int i=0; i<10; i++)
459          lescala[i].SetCodigo(0);
460  }
461
462  void Tesoreria::cargaescalas(const char*nombArch) {
```

```
463         ifstream arch(nombArch, ios::in);
464         if(not arch.is_open()){
465             cout<<"No se pudo abrir el archivo "<<nombArch<<endl;
466             exit(1);
467         }
468         int cod;
469         double precio;
470 
471         while(true){
472             arch>>cod;
473             if(arch.eof())break;
474             arch.get();
475             arch>>precio;
476             lescala[cod-1].SetCodigo(cod);
477             lescala[cod-1].SetPrecio(precio);
478         }
479     }
480 
481     void Tesoreria::cargaalumnos(const char*nombArch) {
482         ifstream arch(nombArch, ios::in);
483         if(not arch.is_open()){
484             cout<<"No se pudo abrir el archivo "<<nombArch<<endl;
485             exit(1);
486         }
487         //S,202121791,GONZALES/VELASQUEZ/WALTER,25,1
488         //P,202115802,MINAYA/AMEZQUITA/RHONY-JAIME,20,5
489         //V,202318072,RIVERA/MONTERO/GLORIA-OFELIA,323R33-2,3
490         char tipo;
491         int nP=0, nS=0, nV=0;
492         while(true){
493             arch>>tipo;
494             if(arch.eof())break;
495             arch.get();
496             switch (tipo){
497                 case 'P':
498                     lpresencial[nP].leerDatos(arch);
499                     nP++;
500                     break;
501                 case 'S':
502                     lsemipresencial[nS].leerDatos(arch);
503                     nS++;
504                     break;
505                 case 'V':
506                     lvirtual[nV].leerDatos(arch);
507                     nV++;
508                     break;
509             }
510         }
511         lpresencial[nP].SetCodigo(0);
512         lsemipresencial[nS].SetCodigo(0);
513         lvirtual[nV].SetCodigo(0);
514     }
515 
516     void Tesoreria::actualiza(double cred) {
517         int esc;
518         double precioEsc;
519         for(int i=0; lpresencial[i].GetCodigo(); i++){
520             esc = lpresencial[i].GetEscala();
521             precioEsc = lescala[esc-1].GetPrecio();
522             lpresencial[i].actualiza(cred*precioEsc);
523         }
524         for(int i=0; lsemipresencial[i].GetCodigo(); i++){
525             esc = lsemipresencial[i].GetEscala();
526             precioEsc = lescala[esc-1].GetPrecio();
527             lsemipresencial[i].actualiza(cred*precioEsc);
528         }
```

```
529         for(int i=0; lvirtual[i].GetCodigo(); i++){
530             esc = lvirtual[i].GetEscala();
531             precioEsc = lescala[esc-1].GetPrecio();
532             lvirtual[i].actualiza(cred*precioEsc);
533         }
534     }
535
536     void Tesoreria::imprime(const char*nombArch) {
537         ofstream arch(nombArch, ios::out);
538         if(not arch.is_open()){
539             cout<<"No se pudo abrir el archivo "<<nombArch<<endl;
540             exit(1);
541         }
542         arch.precision(2);
543         arch<<fixed;
544         arch<<left<<setw(10)<<"Codigo"<<setw(40)<<"Nombre"<<right
545             <<setw(3)<<"Escala"<<right<<setw(12)<<"Licencia"
546             <<setw(8)<<"Total"<<endl;
547         imprimeLinea(arch,'=',80);
548         for(int i=0; lpresencial[i].GetCodigo(); i++)
549             lpresencial[i].imprime(arch);
550
551         for(int i=0; lsemipresencial[i].GetCodigo(); i++)
552             lsemipresencial[i].imprime(arch);
553
554         for(int i=0; lvirtual[i].GetCodigo(); i++)
555             lvirtual[i].imprime(arch);
556     }
557
558     void Tesoreria::imprimeLinea(ofstream&arch, char car, int nd) {
559         for(int i=0; i<nd; i++)
560             arch.put(car);
561         arch<<endl;
562     }
563
564     void Tesoreria::muestraEscala(int pos) {
565         cout<<lescala[pos].GetCodigo()<<','<<lescala[pos].GetPrecio()<<endl;
566     }
567
568     /*
569      * Proyecto: SolucionLaboratorio08_2023_1
570      * Archivo:  main.cpp
571      * Autor:    J. Miguel Guanira E.
572      *
573      * Created on 18 de octubre de 2023, 08:51 PM
574      */
575
576     #include <iostream>
577     #include <iomanip>
578     using namespace std;
579     #include "Tesoreria.h"
580     int main(int argc, char** argv) {
581         class Tesoreria caja;  // {}
582         caja.cargaescalas("escalas.csv");
583         caja.cargaalumnos("Alumnos.csv");
584         caja.actualiza(20);
585         caja.imprime("reporteDePagos.txt");
586
587     //    cout.precision(2);
588     //    cout<<fixed;
589     //    for(int i=0; i<10; i++)
590     //        tesoreria.muestraEscala(i);
591         return 0;
592     }
```