```
1   /*
2    * Proyecto: SolucionLaboratorio09_2023_1
3    * Archivo:  Boleta.h
4    * Autor:    J. Miguel Guanira E.
5    *
6    * Created on 19 de octubre de 2023, 07:19 PM
7    */
8
9
10  #ifndef BOLETA_H
11  #define BOLETA_H
12  #include "Alumno.h"
13
14  class Boleta {
15  private:
16      class Alumno *pboleta;
17  public:
18      Boleta();
19      virtual ~Boleta();
20      void asignaPresencial();
21      void asignaSemipresencial();
22      void asignaVirtual();
23      void lee(ifstream &arch);
24      void imprime(ofstream &arch);
25      bool ultimo();
26      int GetEscala();
27      void actualiza(double);
28  };
29
30  #endif /* BOLETA_H */
31
32  /*
33   * Proyecto: SolucionLaboratorio09_2023_1
34   * Archivo:  Boleta.cpp
35   * Autor:    J. Miguel Guanira E.
36   *
37   * Created on 19 de octubre de 2023, 07:19 PM
38   */
39
40  #include <iostream>
41  #include <iomanip>
42  using namespace std;
43  #include "Presencial.h"
44  #include "Semipresencial.h"
45  #include "Virtual.h"
46
47  #include "Boleta.h"
48
49  Boleta::Boleta() {
50      pboleta = nullptr;
51  }
52
53  Boleta::~Boleta() {
54      if(pboleta!=nullptr) delete pboleta;
55  }
56
57  void Boleta::asignaPresencial() {
58      pboleta = new class Presencial;
59  }
60
61  void Boleta::asignaSemipresencial() {
62      pboleta = new class Semipresencial;
63  }
64
65  void Boleta::asignaVirtual() {
66      pboleta = new class Virtual;
```

```
67  }
68
69  void Boleta::lee(ifstream& arch) {
70      pboleta->leerDatos(arch); //Polimorfismo
71  }
72
73  bool Boleta::ultimo() {
74      return pboleta==nullptr;
75  }
76
77  void Boleta::imprime(ofstream& arch) {
78      pboleta->imprime(arch); //Polimorfismo
79  }
80
81  int Boleta::GetEscala() {
82      return pboleta->GetEscala();
83  }
84
85  void Boleta::actualiza(double monto) {
86      pboleta->actualiza(monto);
87  }
88
89  /*
90   * Proyecto: SolucionLaboratorio08_2023_1
91   * Archivo:  Tesoreria.h
92   * Autor:    J. Miguel Guanira E.
93   *
94   * Created on 18 de octubre de 2023, 09:26 PM
95   */
96
97
98  #ifndef TESORERIA_H
99  #define TESORERIA_H
100 #include "Boleta.h"
101 #include "Escala.h"
102 class Tesoreria {
103 private:
104     class Boleta lboleta[100];
105     class Escala lescala[10];
106 public:
107     Tesoreria();
108     void cargaescalas(const char *);
109     void cargaalumnos(const char *);
110     void actualiza(double);
111     void imprime(const char *);
112     void muestraEscala(int);
113     void imprimeLinea(ofstream &, char, int);
114 };
115
116 #endif /* TESORERIA_H */
117
118
119
120 /*
121  * Proyecto: SolucionLaboratorio08_2023_1
122  * Archivo:  Tesoreria.cpp
123  * Autor:    J. Miguel Guanira E.
124  *
125  * Created on 18 de octubre de 2023, 09:26 PM
126  */
127
128 #include <iostream>
129 #include <fstream>
130 #include <iomanip>
131 using namespace std;
132
```

```
133  #include "Tesoreria.h"
134
135  Tesoreria::Tesoreria() {
136      for(int i=0; i<10; i++)
137          lescala[i].SetCodigo(0);
138  }
139
140  void Tesoreria::cargaescalas(const char*nombArch) {
141      ifstream arch(nombArch, ios::in);
142      if(not arch.is_open()){
143          cout<<"No se pudo abrir el archivo "<<nombArch<<endl;
144          exit(1);
145      }
146      int cod;
147      double precio;
148
149      while(true){
150          arch>>cod;
151          if(arch.eof())break;
152          arch.get();
153          arch>>precio;
154          lescala[cod-1].SetCodigo(cod);
155          lescala[cod-1].SetPrecio(precio);
156      }
157  }
158
159  void Tesoreria::cargaalumnos(const char*nombArch) {
160      ifstream arch(nombArch, ios::in);
161      if(not arch.is_open()){
162          cout<<"No se pudo abrir el archivo "<<nombArch<<endl;
163          exit(1);
164      }
165
166      char tipo;
167      int nD=0;
168      while(true){
169          arch>>tipo;
170          if(arch.eof())break;
171          arch.get();
172          switch (tipo){
173              case 'P':
174                  lboleta[nD].asignaPresencial();
175                  break;
176              case 'S':
177                  lboleta[nD].asignaSemipresencial();
178                  break;
179              case 'V':
180                  lboleta[nD].asignaVirtual();
181                  break;
182          }
183          lboleta[nD].lee(arch);
184          nD++;
185      }
186
187  }
188
189  void Tesoreria::actualiza(double cred) {
190      int esc;
191      double precioEsc;
192      for(int i=0; not lboleta[i].ultimo(); i++){
193          esc = lboleta[i].GetEscala();
194          precioEsc = lescala[esc-1].GetPrecio();
195          lboleta[i].actualiza(cred*precioEsc);
196      }
197  }
198
```

```
199  void Tesoreria::imprime(const char*nombArch) {
200      ofstream arch(nombArch, ios::out);
201      if(not arch.is_open()){
202          cout<<"No se pudo abrir el archivo "<<nombArch<<endl;
203          exit(1);
204      }
205      arch.precision(2);
206      arch<<fixed;
207      arch<<left<<setw(10)<<"Codigo"<<setw(40)<<"Nombre"<<right
208          <<setw(3)<<"Escala"<<right<<setw(12)<<"Licencia"
209          <<setw(8)<<"Total"<<endl;
210      imprimeLinea(arch,'=',80);
211      for(int i=0; not lboleta[i].ultimo(); i++)
212          lboleta[i].imprime(arch);
213
214
215  }
216
217  void Tesoreria::imprimeLinea(ofstream&arch, char car, int nd) {
218      for(int i=0; i<nd; i++)
219          arch.put(car);
220      arch<<endl;
221  }
222
223  void Tesoreria::muestraEscala(int pos) {
224      cout<<lescala[pos].GetCodigo()<<','<<lescala[pos].GetPrecio()<<endl;
225  }
226
227  /*
228   * Proyecto: SolucionLaboratorio08_2023_1
229   * Archivo:  Alumno.h
230   * Autor:    J. Miguel Guanira E.
231   *
232   * Created on 18 de octubre de 2023, 08:53 PM
233   */
234
235
236  #ifndef ALUMNO_H
237  #define ALUMNO_H
238
239  class Alumno {
240  private:
241      int codigo;
242      char *nombre;
243      int escala ;
244      double total;
245  public:
246      Alumno();
247      virtual ~Alumno();
248      void SetTotal(double total);
249      double GetTotal() const;
250      void SetEscala(int escala);
251      int GetEscala() const;
252      void SetNombre(const char* nombre);
253      void GetNombre(char*) const;
254      void SetCodigo(int codigo);
255      int GetCodigo() const;
256      virtual void leerDatos(ifstream &);
257      virtual void imprime(ofstream&);
258      virtual void actualiza(double)=0;
259  };
260
261  #endif /* ALUMNO_H */
262
263  /*
264   * Proyecto: SolucionLaboratorio08_2023_1
```

```
265     * Archivo:  Alumno.cpp
266     * Autor:    J. Miguel Guanira E.
267     *
268     * Created on 18 de octubre de 2023, 08:53 PM
269     */
270
271    #include <iostream>
272    #include <fstream>
273    #include <iomanip>
274    using namespace std;
275    #include <cstring>
276    #include "Alumno.h"
277
278    Alumno::Alumno() {
279        nombre = nullptr;
280        total = 0;
281    }
282
283    Alumno::~Alumno() {
284        if(nombre!=nullptr) delete nombre;
285    }
286
287    void Alumno::SetTotal(double total) {
288        this->total = total;
289    }
290
291    double Alumno::GetTotal() const {
292        return total;
293    }
294
295    void Alumno::SetEscala(int escala) {
296        this->escala = escala;
297    }
298
299    int Alumno::GetEscala() const {
300        return escala;
301    }
302
303    void Alumno::GetNombre(char*cad) const {
304        if(nombre==nullptr) cad[0]=0;
305        else strcpy(cad,nombre);
306    }
307
308    void Alumno::SetNombre(const char* cad) {
309        if(nombre!=nullptr) delete nombre;
310        nombre = new char[strlen(cad)+1];
311        strcpy(nombre,cad);
312    }
313
314    void Alumno::SetCodigo(int codigo) {
315        this->codigo = codigo;
316    }
317
318    int Alumno::GetCodigo() const {
319        return codigo;
320    }
321
322    void Alumno::leerDatos(ifstream &arch) {
323        char nomb[60];
324        arch>>codigo;
325        if(arch.eof())return;
326        arch.get();
327        arch.getline(nomb,60,',');
328        SetNombre(nomb);
329        arch>>escala;
330        arch.get();
```

```
331 }
332
333 void Alumno::imprime(ofstream&arch) {
334     arch<<left<<setw(10)<<codigo<<setw(40)<<nombre<<right<<setw(3)<<escala;
335 }
336
337 /*
338  * Proyecto: SolucionLaboratorio08_2023_1
339  * Archivo:  Presencial.h
340  * Autor:    J. Miguel Guanira E.
341  *
342  * Created on 18 de octubre de 2023, 09:11 PM
343  */
344
345
346 #ifndef PRESENCIAL_H
347 #define PRESENCIAL_H
348 #include <fstream>
349 using namespace std;
350 #include "Alumno.h"
351
352 class Presencial : public Alumno{
353 private:
354     double recargo;
355     double total;
356 public:
357     void SetTotal(double total);
358     double GetTotal() const;
359     void SetRecargo(double recargo);
360     double GetRecargo() const;
361     void leerDatos(ifstream&);
362     void actualiza(double);
363     void imprime(ofstream&);
364 };
365
366 #endif /* PRESENCIAL_H */
367
368 /*
369  * Proyecto: SolucionLaboratorio08_2023_1
370  * Archivo:  Presencial.cpp
371  * Autor:    J. Miguel Guanira E.
372  *
373  * Created on 18 de octubre de 2023, 09:11 PM
374  */
375
376 #include <iostream>
377 #include <iomanip>
378 using namespace std;
379
380 #include "Presencial.h"
381
382 void Presencial::SetTotal(double total) {
383     this->total = total;
384 }
385
386 double Presencial::GetTotal() const {
387     return total;
388 }
389
390 void Presencial::SetRecargo(double recargo) {
391     this->recargo = recargo;
392 }
393
394 double Presencial::GetRecargo() const {
395     return recargo;
396 }
```

```
397
398 void Presencial::leerDatos(ifstream &arch) {
399     Alumno::leerDatos(arch);
400     arch>>recargo;
401     arch.get(); // Lee el cambio de línea
402 }
403
404 void Presencial::actualiza(double monto) {
405     total = monto*GetRecargo()/100;
406     Alumno::SetTotal(monto+total);
407 }
408
409 void Presencial::imprime(ofstream&arch) {
410     Alumno::imprime(arch);
411     arch<<right<<setw(15)<<"+"<<setw(10)<<Alumno::GetTotal()<<endl;
412 }
413
414 /*
415  * Proyecto: SolucionLaboratorio08_2023_1
416  * Archivo:  Semipresencial.h
417  * Autor:    J. Miguel Guanira E.
418  *
419  * Created on 18 de octubre de 2023, 09:15 PM
420  */
421
422
423 #ifndef SEMIPRESENCIAL_H
424 #define SEMIPRESENCIAL_H
425 #include <fstream>
426 using namespace std;
427 #include "Alumno.h"
428
429 class Semipresencial: public Alumno {
430 private:
431     double descuento;
432     double total;
433 public:
434     void SetTotal(double total);
435     double GetTotal() const;
436     void SetDescuento(double descuento);
437     double GetDescuento() const;
438     void leerDatos(ifstream&);
439     void actualiza(double);
440     void imprime(ofstream&);
441 };
442
443 #endif /* SEMIPRESENCIAL_H */
444
445 /*
446  * Proyecto: SolucionLaboratorio08_2023_1
447  * Archivo:  Semipresencial.cpp
448  * Autor:    J. Miguel Guanira E.
449  *
450  * Created on 18 de octubre de 2023, 09:15 PM
451  */
452
453 #include <iostream>
454 #include <iomanip>
455 using namespace std;
456
457 #include "Semipresencial.h"
458
459 void Semipresencial::SetTotal(double total) {
460     this->total = total;
461 }
462
```

```
463  double Semipresencial::GetTotal() const {
464      return total;
465  }
466
467  void Semipresencial::SetDescuento(double descuento) {
468      this->descuento = descuento;
469  }
470
471  double Semipresencial::GetDescuento() const {
472      return descuento;
473  }
474
475  void Semipresencial::leerDatos(ifstream&arch) {
476      Alumno::leerDatos(arch);
477      arch>>descuento;
478      arch.get();
479  }
480
481  void Semipresencial::actualiza(double monto) {
482      total = monto*GetDescuento()/100;
483      Alumno::SetTotal(monto-total);
484  }
485
486  void Semipresencial::imprime(ofstream&arch) {
487      Alumno::imprime(arch);
488      arch<<right<<setw(15)<<"-"<<setw(10)<<Alumno::GetTotal()<<endl;
489  }
490
491  /*
492   * Proyecto: SolucionLaboratorio08_2023_1
493   * Archivo:  Virtual.h
494   * Autor:    J. Miguel Guanira E.
495   *
496   * Created on 18 de octubre de 2023, 09:17 PM
497   */
498
499
500  #ifndef VIRTUAL_H
501  #define VIRTUAL_H
502  #include <fstream>
503  using namespace std;
504  #include "Alumno.h"
505
506  class Virtual: public Alumno {
507  private:
508      char *licencia;
509      double total;
510  public:
511      Virtual();
512      virtual ~Virtual();
513      void SetTotal(double total);
514      double GetTotal() const;
515      void SetLicencia(const char*);
516      void GetLicencia(char*) const;
517      void leerDatos(ifstream&);
518      void actualiza(double);
519      void imprime(ofstream&);
520  };
521
522  #endif /* VIRTUAL_H */
523
524  /*
525   * Proyecto: SolucionLaboratorio08_2023_1
526   * Archivo:  Virtual.cpp
527   * Autor:    J. Miguel Guanira E.
528   *
```

```
529     * Created on 18 de octubre de 2023, 09:17 PM
530     */
531
532    #include <iostream>
533    #include <iomanip>
534    using namespace std;
535    #include <cstring>
536    #include "Virtual.h"
537
538    Virtual::Virtual() {
539        licencia = nullptr;
540        total = 100;
541    }
542
543    Virtual::~Virtual() {
544        if(licencia!=nullptr) delete licencia;
545    }
546
547    void Virtual::SetTotal(double total) {
548        this->total = total;
549    }
550
551    double Virtual::GetTotal() const {
552        return total;
553    }
554
555    void Virtual::SetLicencia(const char* cad) {
556        if(licencia!=nullptr) delete licencia;
557        licencia = new char[strlen(cad)+1];
558        strcpy(licencia,cad);
559    }
560
561    void Virtual::GetLicencia(char*cad) const {
562        if(licencia==nullptr) cad[0]=0;
563        else strcpy(cad,licencia);
564    }
565
566    void Virtual::leerDatos(ifstream&arch) {
567        char lic[30];
568        Alumno::leerDatos(arch);
569        arch.getline(lic,30);
570        SetLicencia(lic);
571    }
572
573    void Virtual::actualiza(double monto) {
574        Alumno::SetTotal(monto+total);
575    }
576
577    void Virtual::imprime(ofstream&arch) {
578        Alumno::imprime(arch);
579        arch<<right<<setw(15)<<licencia<<setw(10)<<Alumno::GetTotal()<<endl;
580    }
581
582    /*
583     * Proyecto: SolucionLaboratorio08_2023_1
584     * Archivo:  Escala.h
585     * Autor:    J. Miguel Guanira E.
586     *
587     * Created on 18 de octubre de 2023, 09:07 PM
588     */
589
590
591    #ifndef ESCALA_H
592    #define ESCALA_H
593
594    class Escala {
```

```
595   private:
596       int codigo;
597       double precio;
598   public:
599       void SetPrecio(double precio);
600       double GetPrecio() const;
601       void SetCodigo(int codigo);
602       int GetCodigo() const;
603   };
604
605   #endif /* ESCALA_H */
606
607   /*
608    * Proyecto: SolucionLaboratorio08_2023_1
609    * Archivo:  Escala.cpp
610    * Autor:    J. Miguel Guanira E.
611    *
612    * Created on 18 de octubre de 2023, 09:07 PM
613    */
614
615   #include <iostream>
616   #include <iomanip>
617   using namespace std;
618
619   #include "Escala.h"
620
621   void Escala::SetPrecio(double precio) {
622       this->precio = precio;
623   }
624
625   double Escala::GetPrecio() const {
626       return precio;
627   }
628
629   void Escala::SetCodigo(int codigo) {
630       this->codigo = codigo;
631   }
632
633   int Escala::GetCodigo() const {
634       return codigo;
635   }
636
637   /*
638    * Proyecto: SolucionLaboratorio08_2023_1
639    * Archivo:  main.cpp
640    * Autor:    J. Miguel Guanira E.
641    *
642    * Created on 18 de octubre de 2023, 08:51 PM
643    */
644
645   #include <iostream>
646   #include <iomanip>
647   using namespace std;
648   #include "Tesoreria.h"
649   int main(int argc, char** argv) {
650       class Tesoreria caja;  // {}
651       caja.cargaescalas("escalas.csv");
652       caja.cargaalumnos("Alumnos.csv");
653       caja.actualiza(20);
654       caja.imprime("reporteDePagos.txt");
655
656   //    cout.precision(2);
657   //    cout<<fixed;
658   //    for(int i=0; i<10; i++)
659   //        tesoreria.muestraEscala(i);
660       return 0;
```

```
661     }
662
663
```