```
1   /*
2    * Proyecto: SolucionLaboratorio08_2023_1
3    * Archivo:  main.cpp
4    * Autor:    J. Miguel Guanira E.
5    *
6   < * Created on 18 de octubre de 2023, 08:51 PM
7    */
8
9   #include <iostream>
10  #include <iomanip>
11  using namespace std;
12  #include "Tesoreria.h"
13
14  int main(int argc, char** argv) {
15      class Tesoreria caja;  // {}
16
17      caja.cargaAlumnos("Alumnos.csv");
18      caja.actualizaBoleta("escalas.csv");
19      caja.imprimeBoleta("reporteDePagos.txt");
20
21      return 0;
22  }
23
24  /*
25   * Proyecto: SolucionLaboratorio089_2023_1
26   * Archivo:  Tesoreria.h
27   * Autor:    J. Miguel Guanira E.
28   *
29   * Creado el 29 de mayo de 2024, 10:40 AM
30   */
31
32
33  #ifndef TESORERIA_H
34  #define TESORERIA_H
35
36  #include "Arbol.h"
37
38  class Tesoreria {
39  private:
40      class Arbol aboleta;
41  public:
42      void cargaEscalas(const char *);
43      void cargaAlumnos(const char *);
44      void actualizaBoleta(const char *);
45      void imprimeBoleta(const char *);
46      void imprimeLinea(ofstream &, char, int);
47  };
48
49  #endif /* TESORERIA_H */
50
51  /*
52   * Proyecto: SolucionLaboratorio09_2023_1
53   * Archivo:  Tesoreria.cpp
54   * Autor:    J. Miguel Guanira E.
55   *
56   * Creado el 29 de mayo de 2024, 10:40 AM
57   */
58
59  #include <iostream>
60  #include <fstream>
61  #include <iomanip>
62  using namespace std;
63
64  #include "Tesoreria.h"
65
66  void Tesoreria::cargaAlumnos(const char*nombArch) {
```

```
67          aboleta.crear(nombArch);
68      }
69
70      void Tesoreria::actualizaBoleta(const char*nombArch) {
71          aboleta.cargaEscalas(nombArch);
72          aboleta.actualizaBoleta();
73      }
74
75      void Tesoreria::imprimeBoleta(const char*nombArch) {
76          aboleta.mostrarEnOrden(nombArch);
77      }
78
79      /*
80       * Proyecto: SolucionLaboratorio10_2023_1
81       * Archivo:  Nodo.h
82       * Autor:    J. Miguel Guanira E.
83       *
84       * Created on 4 de junio de 2024, 06:24 PM
85       */
86
87
88      #ifndef NODO_H
89      #define NODO_H
90
91      #include "Nodo.h"
92      #include "Boleta.h"
93      class Nodo {
94      private:
95          class Boleta dboleta;
96          class Nodo *izq;
97          class Nodo *der;
98      public:
99          Nodo();
100         friend class Arbol;
101     };
102
103     #endif /* NODO_H */
104
105     /*
106      * Proyecto: SolucionLaboratorio10_2023_1
107      * Archivo:  Nodo.cpp
108      * Autor:    J. Miguel Guanira E.
109      *
110      * Created on 4 de junio de 2024, 06:24 PM
111      */
112
113     #include <iostream>
114     #include <iomanip>
115     using namespace std;
116
117     #include "Nodo.h"
118
119     Nodo::Nodo() {
120         izq = nullptr;
121         der = nullptr;
122     }
123
124     /*
125      * Proyecto: SolucionLaboratorio10_2023_1
126      * Archivo:  Nodo.cpp
127      * Autor:    J. Miguel Guanira E.
128      *
129      * Created on 4 de junio de 2024, 06:24 PM
130      */
131
132     #include <iostream>
```

```
133  #include <iomanip>
134  using namespace std;
135
136  #include "Nodo.h"
137
138  Nodo::Nodo() {
139      izq = nullptr;
140      der = nullptr;
141  }
142
143  /*
144   * Proyecto: SolucionLaboratorio10_2023_1
145   * Archivo:  Arbol.h
146   * Autor:    J. Miguel Guanira E.
147   *
148   * Created on 4 de junio de 2024, 06:26 PM
149   */
150
151
152  #ifndef ARBOL_H
153  #define ARBOL_H
154  #include <fstream>
155
156  #include "Nodo.h"
157  #include "Escala.h"
158
159  class Arbol {
160      private:
161      class Nodo *raiz;
162      class Escala lescala[10];
163      void insertarR(class Nodo *&arbol, const class Boleta &boleta);
164      void mostrarEnOrdenR(ofstream &arch, class Nodo *arbol);
165      void actualizaBoletaR(class Nodo *arbol);
166      void eliminarR(class Nodo* raiz);
167  public:
168      Arbol();
169      virtual ~Arbol();
170      void cargaEscalas(const char *);
171      void crear(const char*);
172      void actualizaBoleta();
173      void mostrarEnOrden(const char*);
174      void imprimeLinea(ofstream&, char , int );
175      void eliminar();
176  };
177
178  #endif /* ARBOL_H */
179
180  /*
181   * Proyecto: SolucionLaboratorio10_2023_1
182   * Archivo:  Arbol.cpp
183   * Autor:    J. Miguel Guanira E.
184   *
185   * Created on 4 de junio de 2024, 06:26 PM
186   */
187
188  #include <iostream>
189  #include <iomanip>
190  using namespace std;
191  #include "Arbol.h"
192  #include "Boleta.h"
193
194  Arbol::Arbol() {
195      raiz = nullptr;
196  }
197
198  Arbol::~Arbol() {
```

```
199         eliminarR(raiz);
200     }
201
202     void Arbol::cargaEscalas(const char*nombArch) {
203         ifstream arch(nombArch, ios::in);
204         if(not arch.is_open()){
205             cout<<"No se pudo abrir el archivo "<<nombArch<<endl;
206             exit(1);
207         }
208         int cod;
209         double precio;
210
211         while(true){
212             arch>>cod;
213             if(arch.eof())break;
214             arch.get();
215             arch>>precio;
216             lescala[cod-1].SetCodigo(cod);
217             lescala[cod-1].SetPrecio(precio);
218         }
219     }
220
221     void Arbol::crear(const char*nombArch) {
222         ifstream arch(nombArch, ios::in);
223         if(not arch.is_open()){
224             cout<<"ERROR: No se pudo abrir el archivo "<<nombArch<<endl;
225             exit(1);
226         }
227         class Boleta boleta;
228         char tipo;
229         while(true){
230             arch>>tipo;
231             if(arch.eof())break;
232             arch.get();
233             boleta.asignaMemoria(tipo);
234             boleta.leeDatos(arch);
235             insertarR(raiz,boleta);
236         }
237         boleta.inicializa();
238     }
239
240     void Arbol::insertarR(class Nodo*& arbol, const class Boleta& dato) {
241         if(arbol ==nullptr){
242             arbol = new class Nodo;
243             arbol->dboleta = dato;
244             return;
245         }
246         if(arbol->dboleta>dato)
247             insertarR(arbol->izq,dato);
248         else
249             insertarR(arbol->der,dato);
250     }
251
252     void Arbol::mostrarEnOrden(const char*nombArch) {
253         ofstream arch(nombArch, ios::out);
254         if(not arch.is_open()){
255             cout<<"ERROR: No se pudo abrir el archivo "<<nombArch<<endl;
256             exit(1);
257         }
258         arch<<left<<setw(10)<<"Codigo"<<setw(40)<<"Nombre"<<setw(8)<<"Escala"
259             <<setw(10)<<"Creditos"<<setw(12)<<"Licencia"<<setw(10)<<"Total"<<endl;
260
261         imprimeLinea(arch,'=',90);
262         mostrarEnOrdenR(arch,raiz);
263     }
264
```

```
265  void Arbol::mostrarEnOrdenR(ofstream &arch, class Nodo *raiz) {
266      if(raiz){
267          mostrarEnOrdenR(arch,raiz->izq);
268          arch<<raiz->dboleta;
269          mostrarEnOrdenR(arch,raiz->der);
270      }
271  }
272
273  void Arbol::imprimeLinea(ofstream&arch, char car, int nd) {
274      for(int i=0; i<nd; i++)
275          arch.put(car);
276      arch<<endl;
277  }
278
279  void Arbol::actualizaBoleta() {
280      actualizaBoletaR(raiz);
281  }
282
283  void Arbol::actualizaBoletaR(class Nodo *raiz) {
284      if(raiz == nullptr) return;
285      int esc = raiz->dboleta.GetEscala();
286      double precio = lescala[esc-1].GetPrecio();
287
288      raiz->dboleta.actualizaBoleta(precio);
289      actualizaBoletaR(raiz->izq);
290      actualizaBoletaR(raiz->der);
291  }
292
293  void Arbol::eliminar() {
294      eliminarR(raiz);
295  }
296
297  void Arbol::eliminarR(class Nodo* raiz) {
298      if(raiz == nullptr)return;
299      eliminarR(raiz->izq);
300      eliminarR(raiz->der);
301      delete raiz;
302  }
303
304  /*
305   * Proyecto: SolucionLaboratorio10_2023_1
306   * Archivo:  Boleta.h
307   * Autor:    J. Miguel Guanira E.
308   *
309   * Creado el 04 de junio de 2024, 10:40 AM
310   */
311
312
313  #ifndef BOLETA_H
314  #define BOLETA_H
315  #include <fstream>
316
317  #include "Alumno.h"
318
319  class Boleta {
320  private:
321      class Alumno *pboleta;
322  public:
323      Boleta();
324      virtual ~Boleta();
325      void inicializa();
326      void asignaMemoria(char tipo);
327      void leeDatos(ifstream &arch) const;
328      void actualizaBoleta(double);
329      void imprimeBoleta(ofstream&) const;
330      bool operator >(const class Boleta &dato)const;
```

```
331         int GetCodigo()const;
332         int GetEscala()const;
333     };
334
335     void operator >>(ifstream &arch, class Boleta &boleta);
336     void operator <<(ofstream &arch, const class Boleta &boleta);
337
338     #endif /* BOLETA_H */
339
340     /*
341      * Proyecto: SolucionLaboratorio10_2023_1
342      * Archivo:  Boleta.cpp
343      * Autor:    J. Miguel Guanira E.
344      *
345      * Creado el 04 de junio de 2024, 10:40 AM
346      */
347
348     #include <iostream>
349     #include <iomanip>
350     using namespace std;
351     #include "Presencial.h"
352     #include "Semipresencial.h"
353     #include "Virtual.h"
354
355     #include "Boleta.h"
356
357     Boleta::Boleta() {
358         inicializa();
359     }
360
361     Boleta::~Boleta() {
362         if(pboleta!=nullptr) delete pboleta;
363     }
364
365
366     void Boleta::inicializa() {
367         pboleta = nullptr;
368     }
369
370     void Boleta::asignaMemoria(char tipo) {
371         switch(tipo){
372             case 'P':
373                 pboleta = new class Presencial;
374                 break;
375             case 'S':
376                 pboleta = new class Semipresencial;
377                 break;
378             case 'V':
379                 pboleta = new class Virtual;
380                 break;
381         }
382     }
383
384     void Boleta::leeDatos(ifstream& arch) const{
385         pboleta->lee(arch); // Aquí aplicamos Polimorfismo
386     }
387
388     void Boleta::actualizaBoleta(double precioEscala) {
389         pboleta->actualizaTotal(precioEscala);
390     }
391
392     void Boleta::imprimeBoleta(ofstream&arch) const {
393         pboleta->imprime(arch);
394     }
395
396     bool Boleta::operator >(const class Boleta &dato)const {
```

```
397         return pboleta->GetCodigo()>dato.GetCodigo();
398     }
399
400     int Boleta::GetCodigo() const{
401         return pboleta->GetCodigo();
402     }
403
404     int Boleta::GetEscala() const {
405         return pboleta->GetEscala();
406     }
407
408     void operator >>(ifstream &arch, class Boleta &boleta){
409         boleta.leeDatos(arch);
410     }
411
412     void operator <<(ofstream &arch, const class Boleta &boleta) {
413         boleta.imprimeBoleta(arch);
414     }
415
416     /*
417      * Proyecto: SolucionLaboratorio10_2023_1
418      * Archivo:  Escala.h
419      * Autor:    J. Miguel Guanira E.
420      *
421      * Creado el 04 de junio de 2024, 10:40 AM
422      */
423
424
425     #ifndef ESCALA_H
426     #define ESCALA_H
427
428     class Escala {
429     private:
430         int codigo;
431         double precio;
432     public:
433         Escala();
434         void SetPrecio(double precio);
435         double GetPrecio() const;
436         void SetCodigo(int codigo);
437         int GetCodigo() const;
438     };
439
440     #endif /* ESCALA_H */
441
442     /*
443      * Proyecto: SolucionLaboratorio10_2023_1
444      * Archivo:  Escala.cpp
445      * Autor:    J. Miguel Guanira E.
446      *
447      * Creado el 04 de junio de 2024, 10:40 AM
448      */
449
450     #include <iostream>
451     #include <iomanip>
452     using namespace std;
453
454     #include "Escala.h"
455
456     Escala::Escala() {
457         codigo = 0;
458     }
459
460     void Escala::SetPrecio(double precio) {
461         this->precio = precio;
462     }
```

```
463
464 double Escala::GetPrecio() const {
465     return precio;
466 }
467
468 void Escala::SetCodigo(int codigo) {
469     this->codigo = codigo;
470 }
471
472 int Escala::GetCodigo() const {
473     return codigo;
474 }
475
476 /*
477  * Proyecto: SolucionLaboratorio10_2023_1
478  * Archivo:  Alumno.h
479  * Autor:    J. Miguel Guanira E.
480  *
481  * Creado el 04 de junio de 2024, 10:40 AM
482  */
483
484 #ifndef ALUMNO_H
485 #define ALUMNO_H
486 #include <fstream>
487
488 class Alumno {
489 private:
490     int codigo;
491     char *nombre;
492     int escala ;
493     double creditos;
494     double total;
495 public:
496     Alumno();
497     virtual ~Alumno();
498     void SetTotal(double total);
499     double GetTotal() const;
500     void SetEscala(int escala);
501     int GetEscala() const;
502     void SetNombre(const char* nombre);
503     void GetNombre(char*) const;
504     void SetCodigo(int codigo);
505     int GetCodigo() const;
506     void SetCreditos(double creditos);
507     double GetCreditos() const;
508
509     virtual void lee(ifstream &);
510     virtual void actualizaTotal(double);
511     virtual void imprime(ofstream&);
512 };
513
514 #endif /* ALUMNO_H */
515
516 /*
517  * Proyecto: SolucionLaboratorio10_2023_1
518  * Archivo:  Alumno.cpp
519  * Autor:    J. Miguel Guanira E.
520  *
521  * Creado el 04 de junio de 2024, 10:40 AM
522  */
523
524 #include <iostream>
525 #include <fstream>
526 #include <iomanip>
527 using namespace std;
528 #include <cstring>
```

```cpp
#include "Alumno.h"

Alumno::Alumno() {
    nombre = nullptr;
    total = 0;
}

Alumno::~Alumno() {
    if(nombre!=nullptr) delete nombre;
}

void Alumno::SetTotal(double total) {
    this->total = total;
}

double Alumno::GetTotal() const {
    return total;
}

void Alumno::SetEscala(int escala) {
    this->escala = escala;
}

int Alumno::GetEscala() const {
    return escala;
}

void Alumno::GetNombre(char*cad) const {
    if(nombre==nullptr) cad[0]=0;
    else strcpy(cad,nombre);
}

void Alumno::SetNombre(const char* cad) {
    if(nombre!=nullptr) delete nombre;
    nombre = new char[strlen(cad)+1];
    strcpy(nombre,cad);
}

void Alumno::SetCodigo(int codigo) {
    this->codigo = codigo;
}

int Alumno::GetCodigo() const {
    return codigo;
}

void Alumno::SetCreditos(double creditos) {
    this->creditos = creditos;
}

double Alumno::GetCreditos() const {
    return creditos;
}

void Alumno::lee(ifstream&arch) {
        char nomb[60],c;
        arch>>codigo;
        arch.get();
        arch.getline(nomb,60,',');
        SetNombre(nomb);
        arch>>escala>>c>>creditos;
        arch.get();
}

void Alumno::actualizaTotal(double pago) {
    total = pago;  // también   SetTotal(pago);
```

```
595  }
596
597  void Alumno::imprime(ofstream&arch) {
598      arch.precision(2);
599      arch<<fixed;
600      arch<<left<<setw(10)<<codigo<<setw(40)<<nombre<<right<<setw(3)<<escala
601          <<setw(12)<<creditos;
602
603  }
604
605  /*
606   * Proyecto: SolucionLaboratorio10_2023_1
607   * Archivo:  Presencial.h
608   * Autor:    J. Miguel Guanira E.
609   *
610   * Creado el 04 de junio de 2024, 10:40 AM
611   */
612
613
614  #ifndef PRESENCIAL_H
615  #define PRESENCIAL_H
616  #include <fstream>
617  using namespace std;
618  #include "Alumno.h"
619
620  class Presencial : public Alumno{
621  private:
622      double recargo;
623      double total;
624  public:
625      Presencial();
626      void SetTotal(double total);
627      double GetTotal() const;
628      void SetRecargo(double recargo);
629      double GetRecargo() const;
630      void lee(ifstream &);
631      void actualizaTotal(double);
632      void imprime(ofstream&);
633  };
634
635  #endif /* PRESENCIAL_H */
636
637  /*
638   * Proyecto: SolucionLaboratorio10_2023_1
639   * Archivo:  Presencial.cpp
640   * Autor:    J. Miguel Guanira E.
641   *
642   * Creado el 04 de junio de 2024, 10:40 AM
643   */
644
645  #include <iostream>
646  #include <iomanip>
647  using namespace std;
648
649  #include "Presencial.h"
650
651  Presencial::Presencial() {
652      total = 0.0;
653  }
654
655  void Presencial::SetTotal(double total) {
656      this->total = total;
657  }
658
659  double Presencial::GetTotal() const {
660      return total;
```

```
661 }
662
663 void Presencial::SetRecargo(double recargo) {
664     this->recargo = recargo;
665 }
666
667 double Presencial::GetRecargo() const {
668     return recargo;
669 }
670
671 void Presencial::lee(ifstream&arch) {
672     Alumno::lee(arch);
673     arch>>recargo;
674     arch.get(); // Saco el cambio de línea porque la línea empieza en
675                 // un caracter
676 }
677
678 void Presencial::actualizaTotal(double precioEscala) {
679     total = precioEscala*GetCreditos()*(1+GetRecargo()/100.0);
680     Alumno::SetTotal(total);
681 }
682
683 void Presencial::imprime(ofstream&arch) {
684     Alumno::imprime(arch);
685     arch<<setw(22)<<total<<endl;
686 }
687
688 /*
689  * Proyecto: SolucionLaboratorio10_2023_1
690  * Archivo:  Semipresencial.h
691  * Autor:    J. Miguel Guanira E.
692  *
693  * Creado el 04 de junio de 2024, 10:40 AM
694  */
695
696
697 #ifndef SEMIPRESENCIAL_H
698 #define SEMIPRESENCIAL_H
699 #include <fstream>
700 using namespace std;
701 #include "Alumno.h"
702
703 class Semipresencial: public Alumno {
704 private:
705     double descuento;
706     double total;
707 public:
708     Semipresencial();
709     void SetTotal(double total);
710     double GetTotal() const;
711     void SetDescuento(double descuento);
712     double GetDescuento() const;
713     void lee(ifstream&);
714     void actualizaTotal(double);
715     void imprime(ofstream&);
716 };
717
718 #endif /* SEMIPRESENCIAL_H */
719
720 /*
721  * Proyecto: SolucionLaboratorio10_2023_1
722  * Archivo:  Semipresencial.cpp
723  * Autor:    J. Miguel Guanira E.
724  *
725  * Creado el 04 de junio de 2024, 10:40 AM
726  */
```

```
727
728 #include <iostream>
729 #include <iomanip>
730 using namespace std;
731
732 #include "Semipresencial.h"
733
734 Semipresencial::Semipresencial() {
735     total = 0.0;
736 }
737
738 void Semipresencial::SetTotal(double total) {
739     this->total = total;
740 }
741
742 double Semipresencial::GetTotal() const {
743     return total;
744 }
745
746 void Semipresencial::SetDescuento(double descuento) {
747     this->descuento = descuento;
748 }
749
750 double Semipresencial::GetDescuento() const {
751     return descuento;
752 }
753
754 void Semipresencial::lee(ifstream&arch) {
755     Alumno::lee(arch);
756     arch>>descuento;
757     arch.get(); // Saco el cambio de línea porque la línea empieza en
758                 // un caracter
759 }
760
761 void Semipresencial::actualizaTotal(double precioEscala) {
762     total = precioEscala*GetCreditos()*(1-descuento/100.0);
763     Alumno::SetTotal(total);
764 }
765
766 void Semipresencial::imprime(ofstream&arch) {
767     Alumno::imprime(arch);
768     arch<<setw(22)<<total<<endl;
769 }
770
771 /*
772  * Proyecto: SolucionLaboratorio10_2023_1
773  * Archivo:  Virtual.h
774  * Autor:    J. Miguel Guanira E.
775  *
776  * Creado el 04 de junio de 2024, 10:40 AM
777  */
778
779
780 #ifndef VIRTUAL_H
781 #define VIRTUAL_H
782 #include <fstream>
783 using namespace std;
784 #include "Alumno.h"
785
786 class Virtual: public Alumno {
787 private:
788     char *licencia;
789     double total;
790 public:
791     Virtual();
792     virtual ~Virtual();
```

```
793         void SetTotal(double total);
794         double GetTotal() const;
795         void SetLicencia(const char*);
796         void GetLicencia(char*) const;
797         void lee(ifstream&);
798         void actualizaTotal(double);
799         void imprime(ofstream&);
800     };
801
802     #endif /* VIRTUAL_H */
803
804     /*
805      * Proyecto: SolucionLaboratorio10_2023_1
806      * Archivo:  Virtual.cpp
807      * Autor:    J. Miguel Guanira E.
808      *
809      * Creado el 04 de junio de 2024, 10:40 AM
810      */
811
812     #include <iostream>
813     #include <iomanip>
814     using namespace std;
815     #include <cstring>
816     #include "Virtual.h"
817
818     Virtual::Virtual() {
819         licencia = nullptr;
820         total = 0.0;
821     }
822
823     Virtual::~Virtual() {
824         if(licencia!=nullptr) delete licencia;
825     }
826
827     void Virtual::SetTotal(double total) {
828         this->total = total;
829     }
830
831     double Virtual::GetTotal() const {
832         return total;
833     }
834
835     void Virtual::SetLicencia(const char* cad) {
836         if(licencia!=nullptr) delete licencia;
837         licencia = new char[strlen(cad)+1];
838         strcpy(licencia,cad);
839     }
840
841     void Virtual::GetLicencia(char*cad) const {
842         if(licencia==nullptr) cad[0]=0;
843         else strcpy(cad,licencia);
844     }
845
846     void Virtual::lee(ifstream&arch) {
847         char lic[10];
848         Alumno::lee(arch);
849         arch>>lic;
850         arch.get();
851         SetLicencia(lic);
852     }
853
854     void Virtual::actualizaTotal(double precioEscala) {
855         total = precioEscala*GetCreditos() + 100.0;
856         Alumno::SetTotal(total);
857     }
858
```

```
859  void Virtual::imprime(ofstream&arch) {
860      Alumno::imprime(arch);
861      arch<<setw(12)<<licencia<<setw(10)<<total<<endl;
862  }
863
```