Ejercicio de Iteración
(Estructura de Control While/While)

Una Asignatura desea procesar sus secciones y los respectivos alumnos de estas. Por cada alumno se conoce la nota del Primer, Segundo y Tercer parcial.

Imprima:

Por Alumno

La nota final

Por Sección:

El total de Aprobados y el promedio de la sección.

Por Asignatura

El promedio y la cantidad de aprobados

Fase 1 Análisis

1 Lista de responsabilidades

Alumno	Clase
pI pII pIII	Atributos /Alumno
calcNotaFinal	Métodos/Alumno
InterfaceAlumno	Clase
leerPI leerPII leerPIII reporteAlumno leerRespuesta	Métodos/InterfaceAlumno
Seccion	Clase
contAprob contAlumnos acumNotas	Atributo/Sección
procesarAlumno calcPromedioSeccion	Método/Sección
InterfaceSeccion	Clase
reporteSeccion leerRespuesta	Método /InterfaceSección
Asignatura	Clase
contSec acumProm acumAlumApro	Atributo/Asignatura
procesarSeccion calcPromedioAsignatura	Método/Asignatura
InterfaceAsignatura	Clase
reporteAsignatura	Método/InterfaceAsinatura

2 Lógica de los métodos y Frecuencias

Frecuencia	Lógica
n Tanto como alumnos se procesen	**calcNotaFinal** notaFinal = pI +pII+pIII
Frecuencia	**Lógica**

n
Tanto como alumnos se procesen
	procesarAlumno
	// contar Aprobados
	if (alu.calcDefinitiva()>= 47.5)
	contAprob++;
	// contar alumnos de la seccion
	contAlumSec ++;
	// Acumular Notas
	acumNotas+= alu.calcDefinitiva();

n
Tanto como secciones se procesen
	calcPromedioSeccion
	float prom;
	if (contAlumSec >0)
	prom = acumNotas / contAlumSec;
	else
	prom = 0.0;
	return prom;

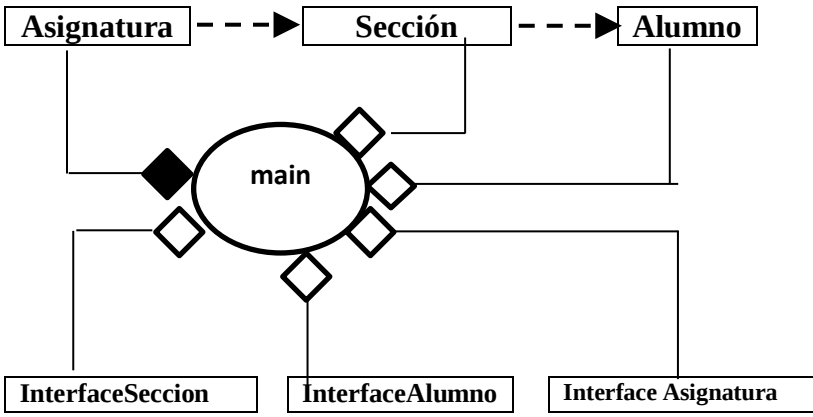
n
Tanto como secciones se procesen
	procesarSeccion
	// Acumular aprobados
	acumAlumnosAprob+= sec.getContAprob();
	// Contar secciones
	contSec++;
	/ Acumular Promedios
	acumProm+= sec.calcPromedio();

1	**calcPromedioAsignatura**
	float prom;
	if (contSec > 0)
	prom = acumProm / contSec;
	else
	prom = 0.0;
	return prom;

3 Estructura de la iteración

```
  float pI, pII, pIII;
  int opc = 1;
int main ()
{
  Alumno alu;
  InterfaceAlumno intAlu;
  Seccion sec;
  InterfaceSeccion intSec;
  Asignatura asi;
  InterfaceAsignatura intAsig;
while (opc == 1) // Iteración que procesa varias secciones
{
      // Se inicializan los contadores, acumuladores,
      mayores y menores de la Seccion
   sec.setContAprob(0);
   sec.setContAlumSec(0);
   sec.setAcumNotas(0.0);

      int resp = 1;
    while (resp == 1) // Iteración que procesa varios alumnos
   {
      // se solicitan los datos del alumno
            pI = intAlu.leerParcialI();
            pII = intAlu.leerParcialII();
            pIII = intAlu.leerParcialIII();

       // se setea o asigna estado al objeto
            alu.setParcialI(pI);
            alu.setParcialII(pII);
            alu.setParcialIII(pIII);

      // se procesan los datos el alumno
            sec.procesarAlumno(alu);

      // se imprime el reporte por alumno
         intAlu.reporteAlumno(alu.calcDefinitiva());

      // se pregunta si se desea procesar
      //otro alumno de la sección
     resp = intAlu.leerRespuesta();
   } // Fin de la iteración de alumnos

  // procesar los datos de la sección
     asi.procesarSeccion(sec);

     // se imprime reporte de la seccion

intSec.reporteSeccion(sec.getContAprob(),
sec.calcPromedio (), sec.getContAlumSec());

  // se pregunta si se desea procesar otra sección
          opc = intSec.leerRespuesta();
} // Fin de la iteracion de secciones
// se imprime reporte de la asignatura
          intAsig.reporteAsignatura(asi.calcPromedio(),
      asi.getAcumAlumnosAprob());

      return 0;
}
```

4 Relación entre clases



Fase 2 Diseño

Deberá realizar esta fase

Fase 3 Implementación

```
#ifndef ALUMNO_H_
#define ALUMNO_H_

class Alumno {
private:
	float parcialI, parcialII,parcialIII;
public:
	Alumno();
	void setParcialI (float pI);
	void setParcialII (float pII);
	void setParcialIII (float pIII);
	float getPacialI ();
	float getPacialII ();
	float getPacialIII ();
	float calcDefinitiva ();
};

#endif /* ALUMNO_H_ */
```

```
//Alumno.cpp
#include "Alumno.h"

Alumno::Alumno() {
parcialI = 0.0;
parcialII = 0.0;
parcialIII = 0.0;
}

void Alumno::setParcialI (float pI)
{
  parcialI = pI;
}
void Alumno::setParcialII (float pII)
{
  parcialII = pII;
}

void Alumno::setParcialIII (float pIII)
{
	parcialIII = pIII;
}

float Alumno::getPacialI ()
{
 return  parcialI;
}

float Alumno::getPacialII ()
{
	return parcialII;
}

float Alumno::getPacialIII ()
{
 return parcialIII;
}

float Alumno::calcDefinitiva ()
{
 return (parcialI + parcialII + parcialIII);}
```

```
// INTERFACEALUMNo.h
#ifndef INTERFACEALUMNO_H_
#define INTERFACEALUMNO_H_
#include <iostream>
using namespace std;
class InterfaceAlumno {
public:
	InterfaceAlumno();
	float leerParcialI();
	float leerParcialII();
	float leerParcialIII();
	void reporteAlumno(float d);
	int leerRespuesta();
};

#endif /* INTERFACEALUMNO_H_ */
```

```
#include "InterfaceAlumno.h"

InterfaceAlumno::InterfaceAlumno() {
	}

float InterfaceAlumno::leerParcialI()
{
  float pI;
   cout<< "Nota del Primer Parcial";
   cin>> pI;
   return pI;
}

float InterfaceAlumno::leerParcialII()
{
	float pII;
	   cout<< "Nota del Segunro Parcial";
	   cin>> pII;
	   return pII;
}

float InterfaceAlumno::leerParcialIII()
{
	float pIII;
	   cout<< "Nota del Tercer Parcial";
	   cin>> pIII;
	   return pIII;
}

void InterfaceAlumno::reporteAlumno(float d)
{
   cout<< "Nota definitiva es"<< d;
}

int InterfaceAlumno::leerRespuesta()
{   int resp ;
	cout<< "Desea procesar otro Alumno  1 Si 2
NO";
    cin >> resp;
    return resp;
}
```

```
// SECCION.h

#ifndef SECCION_H_
#define SECCION_H_
#include "Alumno.h"

class Seccion {
private:
private:
	int contAprob, contAlumSec;
	float acumNotas;
public:
	Seccion();
	void setContAprob (int cA);
	int getContAprob ();
	void setContAlumSec (int cAS);
	int getContAlumSec ();
	void setAcumNotas (float aN);
    float getAcumNotas ();
	void procesarAlumno(Alumno alu);
	float calcPromedio ();
};

#endif /* SECCION_H_ */
```

```
// SECCION.cpp

#include "Seccion.h"

Seccion::Seccion() {
	contAprob = 0;
	contAlumSec = 0;
	acumNotas = 0.0;
}

void Seccion::setContAprob (int cA)
{
 contAprob = cA;
}

void Seccion::setContAlumSec (int cAS)
{
 contAlumSec = cAS;
}

void Seccion::setAcumNotas (float aN)
{
  acumNotas = aN;
}


int Seccion::getContAprob ()
{
  return contAprob;
}

int Seccion::getContAlumSec ()
{
  return contAlumSec;
}

float Seccion::getAcumNotas()
{
 return acumNotas;
}


float Seccion::calcPromedio()
{float prom;

  if (contAlumSec >0)
      prom = acumNotas / contAlumSec;

  else
        prom = 0.0;
  return prom;
}
void Seccion::procesarAlumno(Alumno alu)
{
  // contar Aprobados

       if (alu.calcDefinitiva()>= 47.5)
              contAprob++;
 // contar alumnos de la seccion

        contAlumSec ++;
 // Acumular Notas
         acumNotas+= alu.calcDefinitiva();
}
```

```
//INTERFACESECCION.h
#ifndef INTERFACESECCION_H_
#define INTERFACESECCION_H_
#include <iostream>
using namespace std;

class InterfaceSeccion {
public:
	InterfaceSeccion();
	void reporteSeccion(int a, float p, int
cA);
	int leerRespuesta();
};

#endif /* INTERFACESECCION_H_ */
```

```
//INTERFACESECCION.cpp

#include "InterfaceSeccion.h"

InterfaceSeccion::InterfaceSeccion() {
	// TODO Auto-generated constructor stub

}
```

```
void InterfaceSeccion::reporteSeccion(int a,
float p, int cA )
{   cout<< "REPORTE DE SECCION" <<endl<<endl ;
      cout<< "Cantidad de Aprobados" << a;
      cout << "Promedio de la sección"<< p;
      cout << "Cantidad de Alumnos"<< cA;
}

int InterfaceSeccion::leerRespuesta()
{  int resp ;
cout<< "Desea procesar otra Seccion  1 Si 2 NO";
cin >> resp;
return resp;

}
```

// Asignatura.h

```
#ifndef ASIGNATURA_H_
#define ASIGNATURA_H_
#include "Seccion.h"

class Asignatura {
float acumProm;
int contSec, acumAlumnosAprob;
public:
      Asignatura();
      void setAcumProm (float aP);
      void setContSec (int cS);
      void  setAcumAlumnosAprob (int aAA);
      float getAcumProm ();
      int getcontSec ();
      int getAcumAlumnosAprob ();
      float calcPromedio();
      void procesarSeccion(Seccion sec);
};

#endif /* ASIGNATURA_H_ */
```

// Asignatura.cpp

```
#include "Asignatura.h"

Asignatura::Asignatura() {
      acumProm= 0.0;
    contSec = 0;
    acumAlumnosAprob = 0;

}


void Asignatura::setAcumProm (float aP)
{
  acumProm = aP;
}
void Asignatura::setContSec (int cS)
{
 contSec = cS;
}

void Asignatura::setAcumAlumnosAprob(int aAA)
{
 acumAlumnosAprob = aAA;
}

float  Asignatura::getAcumProm()
{
 return acumProm;
}
int Asignatura::getcontSec()
{
  return contSec;
}

int Asignatura::getAcumAlumnosAprob()
{
 return acumAlumnosAprob;
}

float Asignatura::calcPromedio()
{
float prom;
  if (contSec > 0 )
        prom = acumProm / contSec;
  else
         prom = 0.0;
  return prom;
}

void Asignatura::procesarSeccion(Seccion sec)
{
       // Acumular aprobados
      acumAlumnosAprob+= sec.getContAprob();

     // Contar secciones
       contSec++;

    // Acumular Promedios
       acumProm+= sec.calcPromedio();
}
```

```
// INTERFACEASIGNATURA.h
#ifndef INTERFACEASIGNATURA_H_
#define INTERFACEASIGNATURA_H_
#include <iostream>
using namespace std;

class InterfaceAsignatura {
public:
      InterfaceAsignatura();
      void reporteAsignatura (float p, int cA);
};

#endif /* INTERFACEASIGNATURA_H_ */

// INTERFACEASIGNATURA.cpp
#include "InterfaceAsignatura.h"
```

```
InterfaceAsignatura::InterfaceAsignatura() {
	// TODO Auto-generated constructor stub

}

void InterfaceAsignatura::reporteAsignatura
(float p, int cA)
{
 cout << "Promedio de la asignatura"<< p;
 cout<< "Cantidad de Alumnos aprobados"<< cA;

}
```

// Principal.cpp

```
#include "Alumno.h"
#include "InterfaceAlumno.h"
#include "Seccion.h"
#include "InterfaceSeccion.h"
#include "Asignatura.h"
#include "InterfaceAsignatura.h"

int main ()
{
  Alumno alu;
  InterfaceAlumno intAlu;
  Seccion sec;
  InterfaceSeccion intSec;
  Asignatura asi;
  InterfaceAsignatura intAsig;

  float pI, pII, pIII;

  int opc = 1;

while (opc == 1) // Iteración que procesa varias
secciones
{
// Se inicializan los contadores, acumuladores,
mayores y menores  de la Seccion
    sec.setContAprob(0);
    sec.setContAlumSec(0);
    sec.setAcumNotas(0.0);

int resp = 1;
while (resp == 1)//Iteración que procesa varios alumnos
 {
 // se solicitan los datos del alumno
                pI = intAlu.leerParcialI();
                pII = intAlu.leerParcialII();
                pIII = intAlu.leerParcialIII();
// se setean o asigna estado al objeto
                alu.setParcialI(pI);
                alu.setParcialII(pII);
                alu.setParcialIII(pIII);

 // se procesan los datos el alumno
                sec.procesarAlumno(alu);

// se imprime el reporte por alumno
intAlu.reporteAlumno(alu.calcDefinitiva());
// se pregunta si se desea procesar otro alumno de la sección
resp = intAlu.leerRespuesta();
       } // Fin de la iteracion de alumnos

// procesar los datos de la seccion
        asi.procesarSeccion(sec);

// se imprime reporte de la seccion

intSec.reporteSeccion(sec.getContAprob(),
sec.calcPromedio (), sec.getContAlumSec());

// se pregunta si se desea procesar otra seccion
        opc = intSec.leerRespuesta();

} // Fin de la iteracion de secciones

// se imprime reporte de la asignatura

intAsig.reporteAsignatura(asi.calcPromedio(),
asi.getAcumAlumnosAprob());

return 0;
}
```