Planteamiento:
De un curso se conoce el código y el turno que puede ser (1) Diurno, (2) Nocturno ó (3) Sabatino. El curso tienen varios estudiantes y de cada uno se solicita: nombre y el resultado de cada una de las tres evaluaciones contempladas en el curso. Tenga en cuenta que la evaluación del curso es en escala de 100 ptos. Se pide calcular y mostrar:

Por Estudiante

a) Nota final

Por Curso

a) Número de estudiantes aprobados
b) E nombre del estudiante con la mayor nota
c) Nota promedio del curso
d) Porcentaje de alumnos aprobados
e) La nota promedio de los reprobados

Fase 1 Análisis

Paso 1 Lista de responsabilidades

Lista	Responsabilidad
Estudiante	Clase
nombre	Atributo
nota1	Atributo
nota2	Atributo
nota3	Atributo
calcularNotaFinal	Método
InterfaceEstudiante	Clase
leerNombre	Método
leerNota1	Método
leerNota2	Método
leerNota2	Método
reporteEstudiante	Método
leerRespuesta	Método
Curso	Clase
codCur	Atributo
turnCur	Atributo
contAprobados	Atributo
mayor	Atributo
auxNomb	Atributo
acumNotas	Atributo
contReprobados	Atributo
acumNotasReprob	Atributo
contEstudiantes	Atributo
procesarEstudiante	Método
• Contar los aprobados • Buscar el mejor estudiante • Acumular notas finales • Contar los reprobados • Acumular las notas de los reprobados • Contar los estudiantes procesados	
calcularPromedioCurso	Método
calcularPorcentajeReprobados	Método
calcularNotaPromedioReprobados	Método
InterfaceCurso	Clase
leerCodigo	Método
leerTurno	Método
reporteCurso	Método

Paso 2 Lógica de los métodos

Nombre del método	Clase	Frec
calcularNotaFinal return nota1+nota2+nota3	Estud	3
procesarEstudiante (necesitar el objeto est) **// Contar aprobados** if (est.calcularNotaFinal ( ) > = 47,5) contAprobados = contAprobados + 1 else { contReprobados = contReprobados + 1; acumNotasReprob + = est.calcularNotaFinal ( ); } // **Buscar nombre del mejor estudiante** if (est.calcularNotaFinal ( ) > mayor) { mayor = est.calcularNotaFinal ( ); auxNomb = est.getNombre ( ); } // **Acumular las notas finales** acumNotas = acumNotas + est.calcularNotaFinal ( ) **// Contar los estudiantes procesados** contEstudiantes ++;	Curso	3
calcularPromedioCurso return acumNotas / 3;	Curso	1
calcularPorcentajeReprobados return contReprobados * 100 / contEstudiantes ;	Curso	1
calcularNotaPromedioReprobados return acumNotasReprob / contReprobados;		

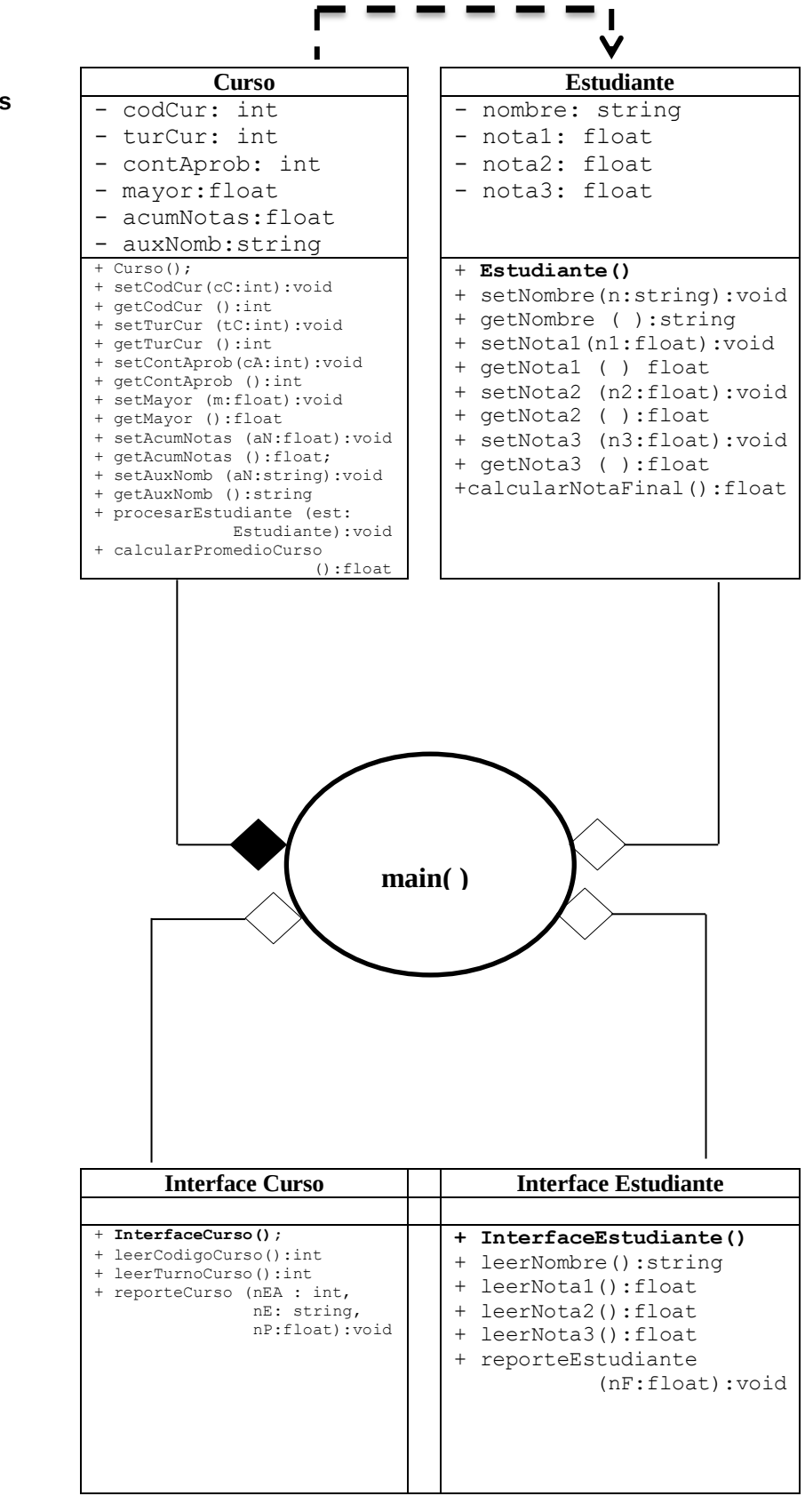
Paso 3 Estructura de la Iteración

Leer datos del Curso
Activar los métodos set de cada atributo que se lee por teclado

```
while (resp == 1)
{
        // leer por teclado los datos del estudiantes

n = intEst.leerNombre();
n1 = intEst.leerNota1();
n2 = intEst.leerNota2();
n3 = intEst.leerNota3();


// Actualizar con el metodo set objeto est

est.setNombre(n);
est.setNota1(n1);
est.setNota2(n2);
est.setNota3(n3);

cur.procesarEstudiante(est);

intEst.reporteEstudiante(est.calcularNotaFinal());
resp = intEst.leerRespuesta();
}

intCur.reporteCurso(cur.getContAprob(),
cur.getAuxNomb(), cur.calcularPromedioCurso(),
cur.calcularPorcentajeReprobados(),
cur.calcularNotaPromedioReprobados() );
```

Fase 2 Diseño

AÑADA LOS ATRIBUTOS Y MÉTODOS FALTANTES EN LAS CLASES



Fase 3 Implementación

// Estudiante.h

```
#ifndef ESTUDIANTE_H_
#define ESTUDIANTE_H_
# include <string>
using namespace std;

class Estudiante {
private:
        string nombre;
        float nota1,nota2,nota3;

public:
        Estudiante();
        void setNombre (string n);
        string getNombre ( );
        void setNota1(float n1);
        float getNota1 ( );
        void setNota2 (float n2);
        float getNota2 ( );
        void setNota3 (float n3);
        float getNota3 ( );
        float calcularNotaFinal ( );
};

#endif /* ESTUDIANTE_H_ */
```

// Estudiante.cpp

```
#include "Estudiante.h"

Estudiante::Estudiante() {
        nombre = " ";
        nota1 = 0.0;
        nota2 = 0.0;
        nota3 = 0.0;
}

void Estudiante::setNombre(string n)
{
  nombre = n;
}

string Estudiante::getNombre()
{
 return nombre;
}

void Estudiante::setNota1(float n1)
{
 nota1 = n1;
}

float Estudiante::getNota1()
{
 return nota1;
}

void Estudiante::setNota2(float n2)
{
 nota2 = n2;
}

float Estudiante::getNota2()
{
 return nota2;
}

void Estudiante::setNota3(float n3)
{
 nota3 = n3;
}

float Estudiante::getNota3()
{
 return nota3;
}

float  Estudiante::calcularNotaFinal()
{
 return nota1 + nota2 + nota3;
}
```

```
// Curso.h
#ifndef CURSO_H_
#define CURSO_H_
#include <string>
using namespace std;
#include "Estudiante.h"

class Curso {
private:
        int codCur, turCur, contAprob, contReprobados,
contEstudiantes;
        float mayor, acumNotas, acumNotasReprob;
        string auxNomb;

public:
        Curso();
        void setContEstudiantes (int cE);
        int getContEstudiantes ();
        void setContReprob (int cR);
        int getContReprob ();
        void setAcumNotasReprob (float aNR);
        float getAcumNotasReprob ();
        void setCodCur (int cC);
        int getCodCur ();
        void setTurCur (int tC);
        int getTurCur ();
        void setContAprob (int cA);
        int getContAprob ();
        void setMayor (float m);
        float getMayor ();
        void setAcumNotas (float aN);
        float getAcumNotas ();
        void setAuxNomb (string aN);
        string getAuxNomb ();
        void procesarEstudiante ( Estudiante est);
        float calcularPromedioCurso ();
        float calcularPorcentajeReprobados ();
        float calcularNotaPromedioReprobados ();
};
#endif /* CURSO_H_ */
```

```
// Curso.cpp

#include "Curso.h"

Curso::Curso() {
        acumNotas = 0.0;
        auxNomb = " ";
        codCur = 0;
        contAprob = 0;
        mayor = 0.0;
        turCur = 0;
        contReprobados = 0;
        acumNotasReprob = 0.0;
        contEstudiantes = 0;
}

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

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

void Curso::setAuxNomb(string aN)
{
 auxNomb = aN;
}

string Curso::getAuxNomb()
{
 return auxNomb;
}

void Curso::setCodCur(int cC)
{
  codCur = cC;

}

int Curso::getCodCur()
{
 return codCur;
}

void Curso::setTurCur(int tC)
{
  turCur = tC;
}

int Curso::getTurCur()
{
 return turCur;
}

void Curso::setContAprob(int cA)
{
  contAprob = cA;

}

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

void Curso::setMayor(float m)
{
 mayor = m;
}

float Curso::getMayor()
{
 return mayor;
}

void Curso::procesarEstudiante(Estudiante est)
{  // Contar los aprobados

if (est.calcularNotaFinal( )>= 47.5 )
                        contAprob ++;
 else
     { contReprobados = contReprobados + 1;
       acumNotasReprob+= est.calcularNotaFinal();
      }

// Buscar mejor estudiante

   if (est.calcularNotaFinal() > mayor )
   {   mayor = est.calcularNotaFinal();
       auxNomb = est.getNombre();
   }

   // Acumular Notas

   acumNotas += est.calcularNotaFinal();

   // Contar los estudiantes procesados
       contEstudiantes ++;
}

float Curso::calcularPromedioCurso()
{
 return acumNotas / contEstudiantes;
}

void Curso::setContReprob (int cR)
{
 contReprobados = cR;
}
int Curso::getContReprob()
{
return contReprobados;
}
void Curso::setAcumNotasReprob (float aNR)
{
 acumNotasReprob = aNR;
}

float Curso::getAcumNotasReprob ()
{
 return acumNotasReprob;
}
```

```
void Curso::setContEstudiantes (int cE)
{
contEstudiantes = cE;
}

int Curso::getContEstudiantes()
{
return contEstudiantes;
}

float Curso::calcularPorcentajeReprobados ()
{
  return  contReprobados * 100 / contEstudiantes ;
}

float Curso::calcularNotaPromedioReprobados ()
{
 return acumNotasReprob / contReprobados;
}
```

//InterfaceCurso.h

```
#ifndef INTERFACECURSO_H_
#define INTERFACECURSO_H_
#include <string>
#include <iostream>
using namespace std;

class InterfaceCurso {
public:
        InterfaceCurso();
        int leerCodigoCurso ();
        int leerTurnoCurso ();
        void reporteCurso (int nEA, string nE,float nP, float
pR, float nPR );
};

#endif /* INTERFACECURSO_H_ */
```

//InterfaceCurso.cpp

```
#include "InterfaceCurso.h"

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

}

int InterfaceCurso::leerCodigoCurso ()
{int cC;
   cout << "Introduzca el código del curso :";
   cin >> cC;
 return cC;

}

int InterfaceCurso::leerTurnoCurso ()
{int tC;
cout << "Introduzca el turno del curso :";
 cin >> tC;
return tC;

}
void InterfaceCurso::reporteCurso (int nEA, string nE,float
nP,float pR, float nPR )
{
            cout << "REPORTE DEl CURSO "<<endl;
           cout << "Número de estudiantes aprobados:"<<
nEA<< endl;
           cout << "Nombre del estudiante con mayor nota:"<<
nE<< endl;
           cout << "Nota promedio del curso:"<< nP<< endl;
           cout << "Porcentaje de reprobados:"<< pR<< endl;
           cout << "Nota promedio de los reprobadoso:"<<
nPR<< endl;
}          cout << "Nota promedio del curso:"<< nP<< endl;
}
```

// InterfaceEstudiante.h

```
#ifndef INTERFACEESTUDIANTE_H_
#define INTERFACEESTUDIANTE_H_
#include <string>
#include <iostream>
using namespace std;

class InterfaceEstudiante {

public:
        InterfaceEstudiante();
        string leerNombre ();
        float leerNota1 ();
        float leerNota2 ();
        float leerNota3 ();
        void reporteEstudiante (float nF);
       int leerRespuesta();

};

#endif /* INTERFACEESTUDIANTE_H_ */
```

```
// InterfaceEstudiante.cpp

#include "InterfaceEstudiante.h"

InterfaceEstudiante::InterfaceEstudiante()
{          }

string InterfaceEstudiante::leerNombre()
{ string n;
   cout << "Introduzca el nombre del estudiante:";
   cin >> n;
  return n;
}

float InterfaceEstudiante::leerNota1()
{ float n1;
   cout << "Introduzca la primera nota :";
   cin >> n1;
  return n1;
}

float InterfaceEstudiante::leerNota2()
{ float n2;
   cout << "Introduzca la segunda nota :";
   cin >> n2;
  return n2;
}

float InterfaceEstudiante::leerNota3()
{ float n3;
   cout << "Introduzca la tercera nota :";
   cin >> n3;
  return n3;
}

void InterfaceEstudiante::reporteEstudiante(float nF)
{
  cout << "REPORTE DE ESTUDIANTE"<<endl;
  cout << "La nota final fue es de:"<< nF<< endl;
}

int InterfaceEstudiante::leerRespuesta()
{ int resp;
            cout << "Desea procesar otro estudiante presione uno
(1) para seguir, otro numero para finalizar :";
             cin >> resp;
           return resp;
}
```

```
// Principal.cpp
#include "Estudiante.h"
#include "Curso.h"
#include "InterfaceEstudiante.h"
#include "InterfaceCurso.h"


int main ()
{
 Curso cur;
 Estudiante est;
 InterfaceCurso intCur;
 InterfaceEstudiante intEst;

 int cC, tC, resp;
 string n;
 float n1,n2,n3;
 resp = 1;

  // leer por teclado los datos del curso

  cC = intCur.leerCodigoCurso();
  tC = intCur.leerTurnoCurso();

  // actualizar el estado de los atributos del objeto cur
  cur.setCodCur(cC);
  cur.setTurCur(tC);


  while (resp == 1)
  {
            // leer por teclado los datos del estudiantes

   n = intEst.leerNombre();
   n1 = intEst.leerNota1();
   n2 = intEst.leerNota2();
   n3 = intEst.leerNota3();


   // Actualizar con el metodo set objeto est

   est.setNombre(n);
   est.setNota1(n1);
   est.setNota2(n2);
   est.setNota3(n3);

   cur.procesarEstudiante(est);

   intEst.reporteEstudiante(est.calcularNotaFinal());
   resp = intEst.leerRespuesta();
  }

   intCur.reporteCurso(cur.getContAprob(),
cur.getAuxNomb(), cur.calcularPromedioCurso(),
cur.calcularPorcentajeReprobados(),
cur.calcularNotaPromedioReprobados() );


return 0;}
```

DATOS DE PRUEBA

CÓDIGO: 216 Turno 2

Nombre	Nota 1	Nota 2	Nota 3	Nota Final
JOSE	20	18	30	68
JUAN	25	30	32	87
ROSA	10	12	20	42
PABLO	3	3	3	9

Número de estudiantes Aprobados: 2
Nombre del mejor estudiante: Juan
La nota promedio del curso es: 51,5
Porcentaje de reprobados 50%
Nota promedio de reprobados : 25,5