Programación 4

SOLUCIÓN FEBRERO 2016

Problema 1

Parte A:

Instancia donde la restricción se cumple:

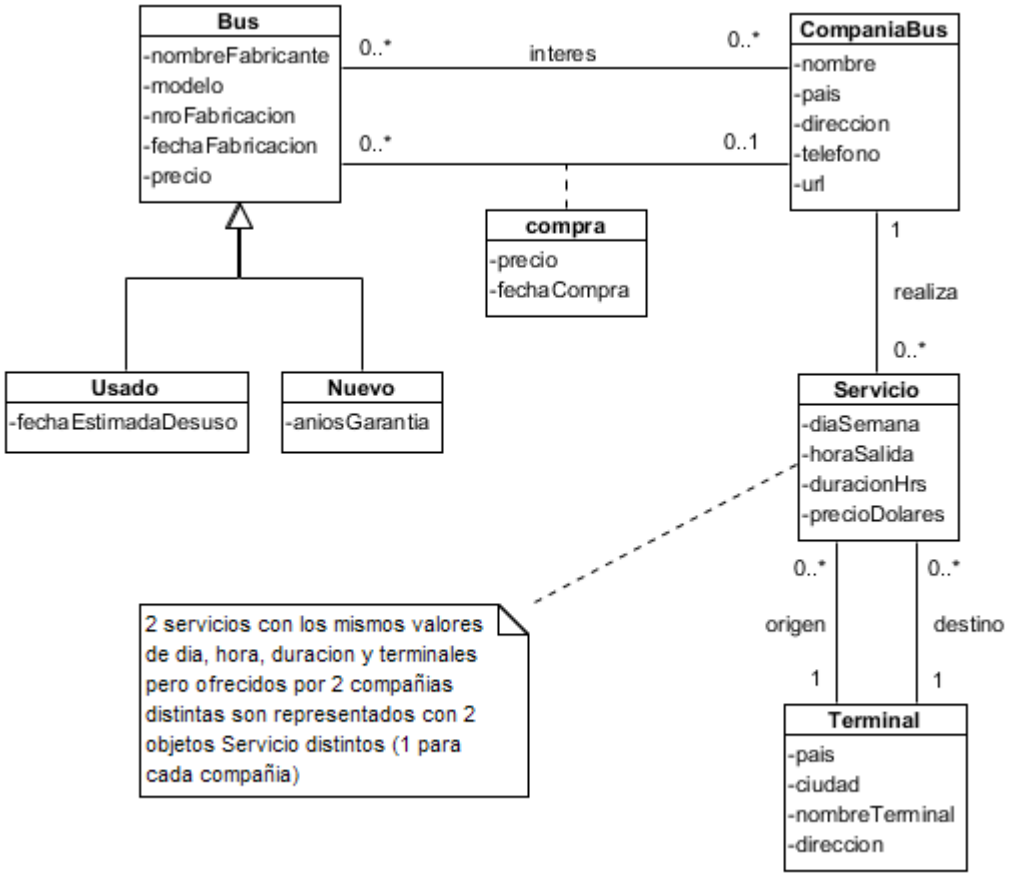
Pais	Ciudad	Pertenece
P1: "Uruguay" P2: "Argentina"	C1: ("Rosario", 4000) C2: ("Rosario", 12000)	L1: (P1, C1) L2: (P2, C2)

Instancia donde la restricción NO se cumple:

Pais	Ciudad	Pertenece
P1: "Uruguay"	C1: ("Rosario", 4000) C2: ("Rosario", 12000)	L1: (P1, C1) L2: (P1, C2)

Parte B:

i) Modelo de dominio



Restricciones no estructurales

Identificadores

1. En Bus, (nombreFabricante, modelo, nroFabricacion)
2. En CompaniaBus, (nombre, pais)
3. En Terminal, (pais, ciudad, nombreTerminal)

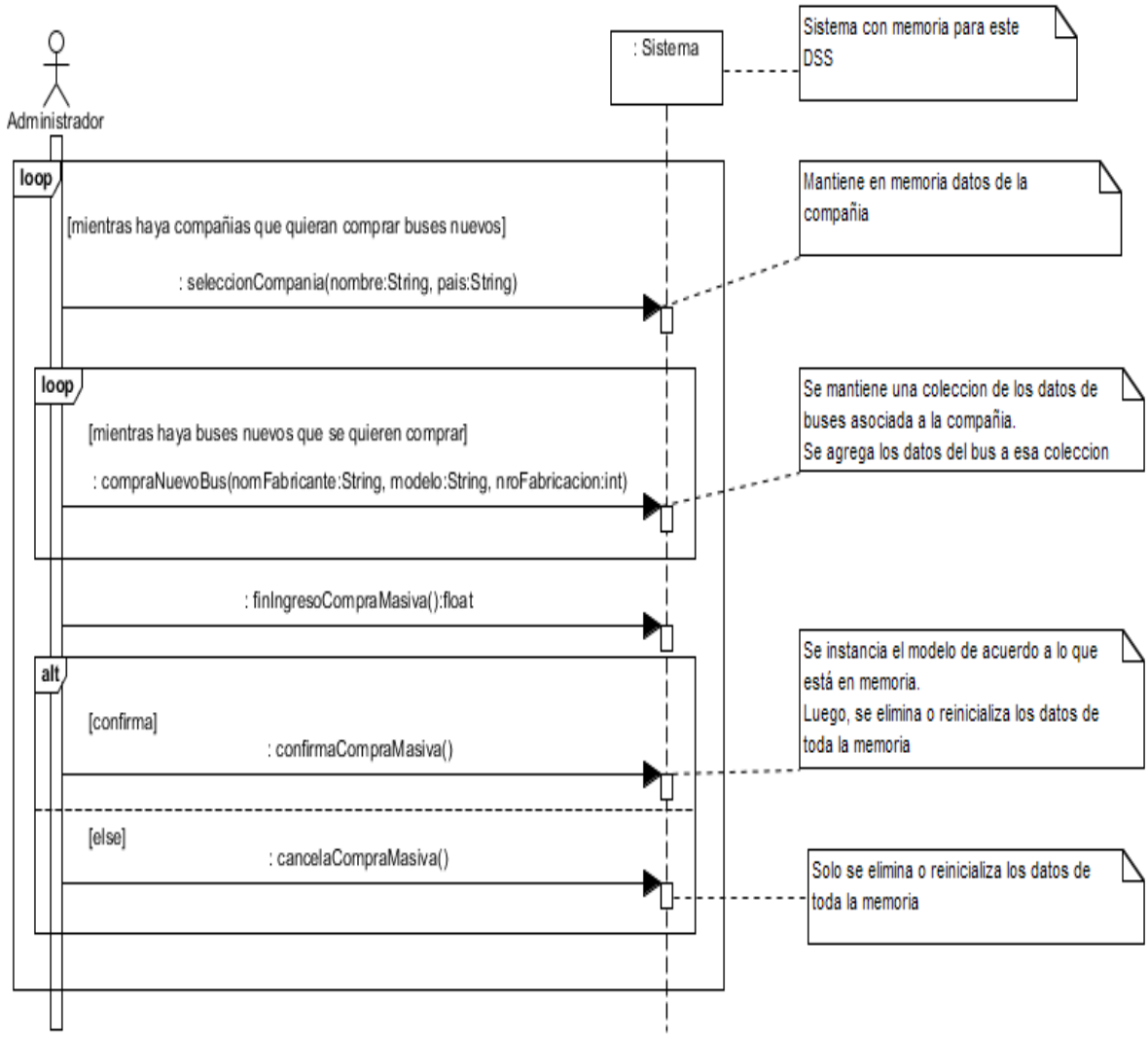
Valores de atributos

4. Dado un bus b, b.precio > 0
5. Dado un bus nuevo bn, bn.aniosGarantia > 0
6. Dado un servicio s, s.diaSemana ∈ {lunes, martes, miercoles, jueves, viernes, sabado, domingo}, s.horaSalida es un datavalue con 2 atributos: hh y mm, donde hh ∈ [1..24] y mm ∈ [0..59] y s.precioDolares > 0

Restricciones entre asociaciones

7. La fecha en que realiza una compra debe ser posterior a la fecha de fabricación
8. En el caso de que se realiza la compra de un bus usado, la fecha de la compra deber ser inferior a la fecha estimada de desuso
9. Dado un link de compra entre la compañía c y el bus usado b1 y existe un bus nuevo b2 tal que b1.nombreFabricante = b2.nombreFabricante, entonces debe existir un link de interes entre la compañía c y un bus nuevo b3 tal que b3.nombreFabricante = b1.nombreFabricante
10. Dado un servicio s, s.origen != s.destino
11. En la asociacion “realiza” se debe cumplir que cada compañía ofrece desde una misma terminal a lo sumo 2 servicios diarios a una misma terminal destino.

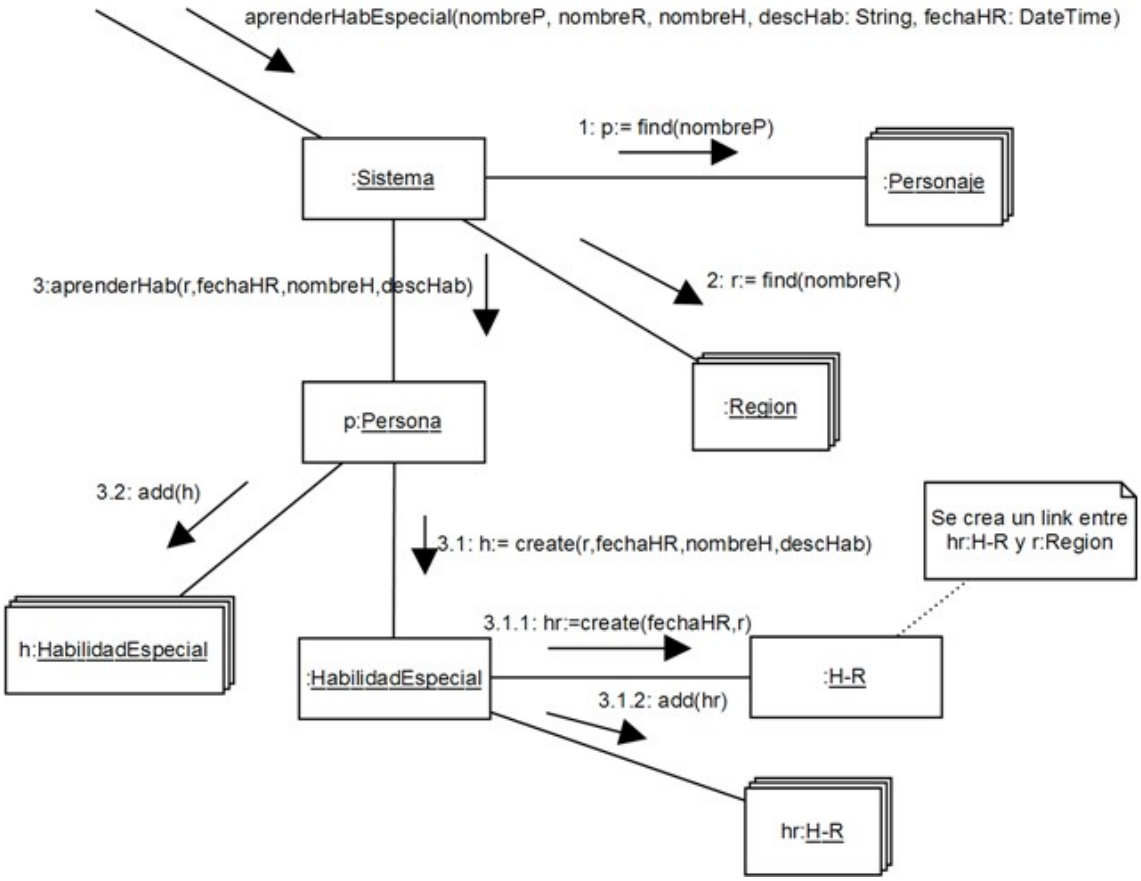
b) DSS para el caso de uso "Ingreso de compras masivas de buses nuevos de compañias



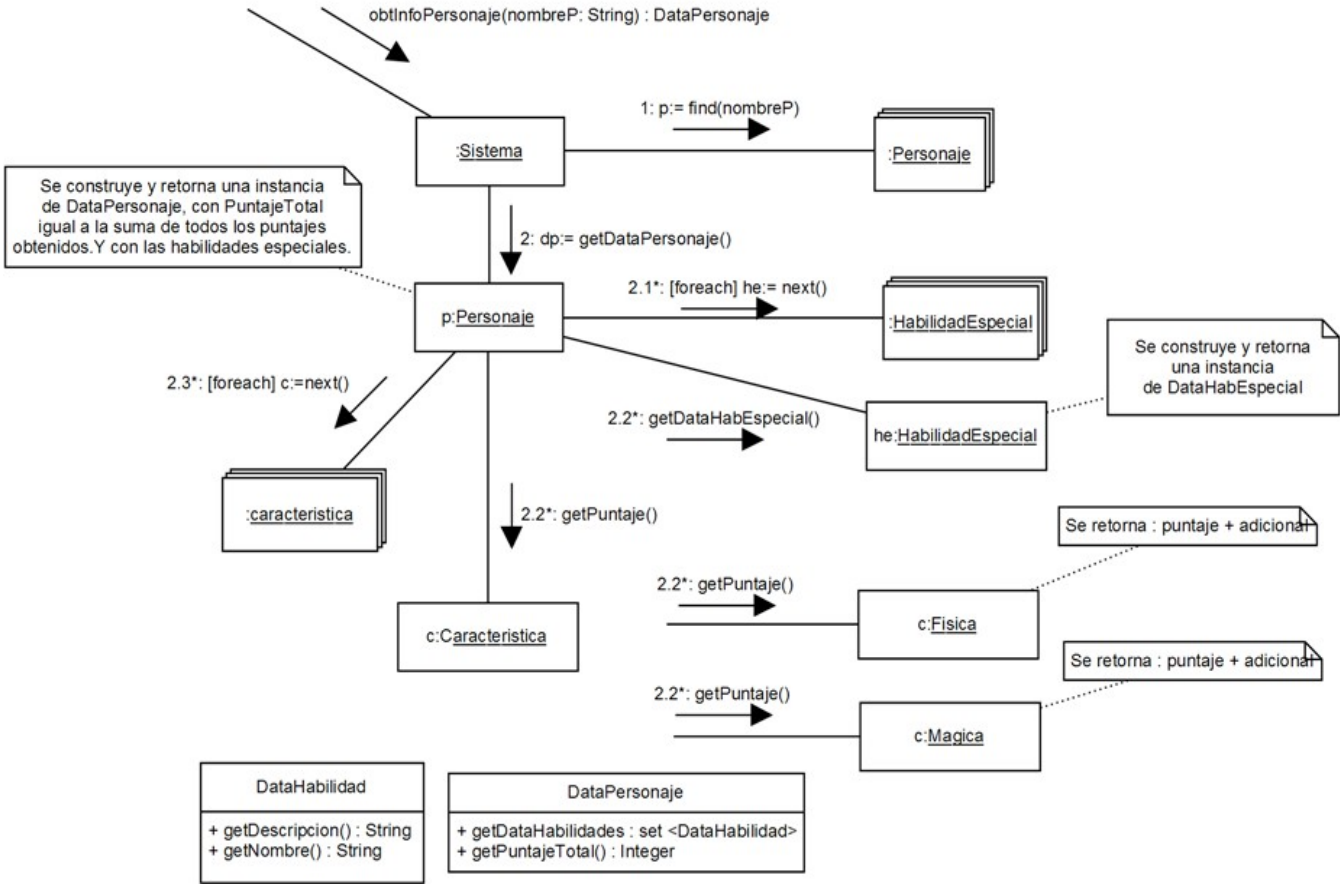
Problema 2

Parte a)

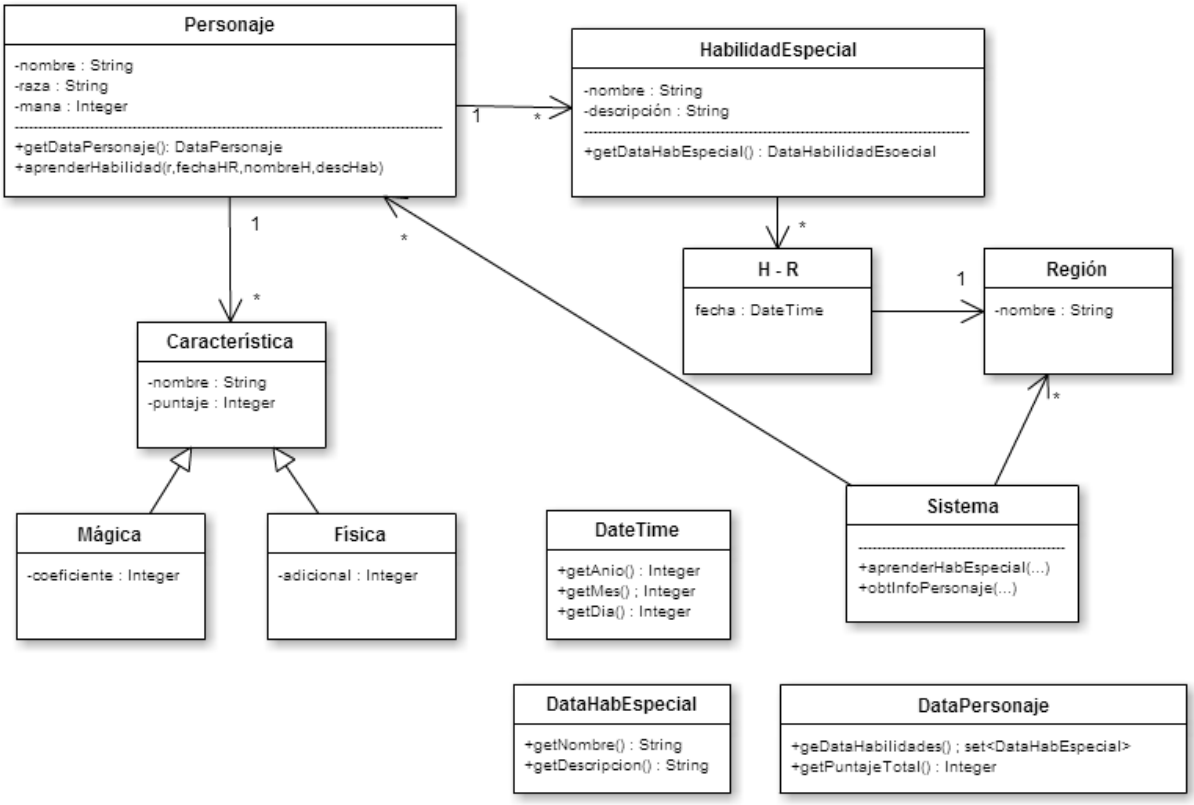
Aprender habilidad especial



Obtener información de un personaje



Parte b)


Personaje
-nombre : String
-raza : String
-mana : Integer
+getDataPersonaje(): DataPersonaje
+aprenderHabilidad(r,fechaHR,nombreH,descHab)
HabilidadEspecial
-nombre : String
-descripción : String
+getDataHabEspecial() : DataHabilidadEsoecial
H - R
fecha : DateTime
Región
-nombre : String
Característica
-nombre : String
-puntaje : Integer
Mágica
-coeficiente : Integer
Física
-adicional : Integer
DateTime
+getAnio() : Integer
+getMes() ; Integer
+getDia() : Integer
Sistema
+aprenderHabEspecial(...)
+obtInfoPersonaje(...)
DataHabEspecial
+getNombre() : String
+getDescripcion() : String
DataPersonaje
+geDataHabilidades() ; set<DataHabEspecial>
+getPuntajeTotal() : Integer
1
*

Problema 3

```
////////////////// DateTime /////////////////

------------------- DateTime.h -------------------
class DateTime: public OrderedKey {
public:
    DateTime();
    DateTime(int, int, int, int, int, int);

    int getAnio();
    int getMes();
    int getDia();
    int getHora();
    int getMinuto();
    int getSegundos();
    string getStr();
    virtual ~DateTime();
    ComparisonRes compare(OrderedKey *k) const;
private:
    int anio;
    int mes;
    int dia;
    int hora;
    int minuto;
    int segundos;
};
------------------- DateTime.cpp ----------------
DateTime::DateTime(){}

DateTime::DateTime(int anio, int mes, int dia, int hora, int minuto,
                   int segundos) {
    this->anio = anio;
    this->mes = mes;
    this->dia = dia;
    this->hora = hora;
    this->minuto = minuto;
    this->segundos = segundos;
}

int DateTime::getAnio() {
    return this->anio;
}

int DateTime::getMes() {
    return this->mes;
}

int DateTime::getDia() {
      return this->dia;
}

int DateTime::getHora() {
    return this->hora;
}

int DateTime::getMinuto() {
    return this->minuto;
}
```

```
int DateTime::getSegundos() {
    return this->segundos;
}

DateTime::~DateTime() {
}
////////////////// TConversor //////////////////
enum TConversor {Audio, SMS};

////////////////// Cliente //////////////////

------------------ Cliente.h ------------------
class Cliente {
private:
    string nombre;
    string apellido;
    string ci;
    string historia;
    string telefono;
    OrderedDictionary* citas;
    Conversor* conversor;
public:
    Cliente(string, string, string, string, string);
    byte* generarNotifUltimaCita();
    void agregarCitaMedica(DateTime fecha, string medico, int numero,
string observ);
    void borrarCitaMedica(DateTime fecha);
    virtual ~Cliente();
};
------------------ Cliente.cpp -----------------
Cliente::Cliente(string nombre, string apellido, string ci, string
historia, string telefono) {
    citas = new OrderedDictionary();
    conversor = new Conversor();
    this->nombre = nombre;
    this->apellido = apellido;
    this->ci = ci;
    this->historia = historia;
    this->telefono = telefono;
}

byte* Cliente::generarNotifUltimaCita(){
    CitaMedica* cita = (CitaMedica*) citas->getMax(); //asumo que Ci-
taMedica implementa ICollectible
    DataNotificacion notif = DataNotificacion(nombre, apellido, tele-
fono,
        cita->fecha, cita->medico, cita->numero);
    return conversor->getNotif(notif);
}

void Cliente::agregarCitaMedica(DateTime fecha, string medico, int nu-
mero, string observ) {
    CitaMedica* cita = new CitaMedica(fecha, medico, numero, observ);
    citas->add(&fecha, cita);
}

void Cliente::borrarCitaMedica(DateTime fecha) {
    CitaMedica* to_remove = citas->find(&fecha);
    citas->remove(&fecha);
    delete to_remove;
}
```

```
Cliente::~Cliente(){
    delete conversor;
    delete citas;
}
////////////////// Conversor /////////////////
------------------ Conversor.h ------------------
class Conversor {
public:
    Conversor();
    byte* getNotif(DataNotificacion info);
    virtual ~Conversor();
};

------------------ Conversor.cpp ----------------
Conversor::Conversor() {
}

byte* Conversor::getNotif(DataNotificacion info) {
    StrategyConversor* strategy;
    if(Configuracion::getInstance()->getTipoConversor() == Audio) {
        strategy = new ConversorAudio();
    } else {
        strategy = new ConversorSMS();
    }
    return strategy->generarNotif(info);
}

Conversor::~Conversor() {}

////////////////// Configuracion /////////////////
------------------ Configuracion.h ------------------
class Configuracion {
private:
    Configuracion();
    static Configuracion* instance;
public:
    static Configuracion* getInstance();
    TConversor getTipoConversor();
    ~Configuracion();
};
------------------ Configuracion.cpp ----------------
Configuracion *Configuracion::instance = NULL;

Configuracion::Configuracion(){}

Configuracion* Configuracion::getInstance() {
    if (instance==NULL)
        instance = new Configuracion();
    return instance;
}

TConversor Configuracion::getTipoConversor() {
      //No implementar
}

Configuracion::~Configuracion(){}
```

```
////////////////// StrategyConversor /////////////////
------------------ StrategyConversor.h ------------------
class StrategyConversor {
protected:
    string notif2str(DataNotificacion info);
public:
    StrategyConversor();
    virtual ~StrategyConversor();
    virtual byte* generarNotif(DataNotificacion info) = 0;
};

------------------ StrategyConversor.cpp ----------------
StrategyConversor::StrategyConversor(){}

string StrategyConversor::notif2str(DataNotificacion info){
    Sistema *sistema = Sistema::getInstance();
    string mensaje = "Estimado cliente " + info.getnombre() + " " +
        info.getapellido() + ", le recordamos que el dia " +
        info.getfecha().getStr() + " tiene una cita con el dr " +
        info.getmedico()+ " y su numero es el " +
        sistema->intToStr(info.getnumero());
    return mensaje;
}

StrategyConversor::~StrategyConversor(){}

////////////////// ConversorSMS /////////////////
------------------ ConversorSMS.h ------------------
class ConversorSMS: public StrategyConversor {
public:
    ConversorSMS();
    byte* generarNotif(DataNotificacion info);
    virtual ~ConversorSMS();
};
------------------ ConversorSMS.cpp ----------------
ConversorSMS::ConversorSMS(){}

byte* ConversorSMS::generarNotif(DataNotificacion info) {
      Sistema *sistema = Sistema::getInstance();
    string sms = "SMSC#\nSender:" + info.getnumtel();
      sms = sms + "\nTimeStamp:" + sistema->getFechaActual();
      sms = sms + "\nTP_PID:00\nTP_DCS:00\nTP_DCS-popis:Uncompressed
Text\n\n";
      sms = sms + notif2str(info) + "\\Length";
      int length = sms.length();
      sms = sms + ":" + sistema->intToStr(length); //el largo segun la
nota es la cantidad de bytes hasta Length inclusive
    return sistema->getStrToBytes(sms);
}
ConversorSMS::~ConversorSMS(){}
```

```
////////////////// ConversorAudio //////////////////
------------------- ConversorAudio.h -------------------
class ConversorAudio: public StrategyConversor {
public:
    ConversorAudio();
    byte* generarNotif(DataNotificacion info);
    virtual ~ConversorAudio();
private:
    byte* getTextToSpeech(string text);
};
------------------- ConversorAudio.cpp -----------------
ConversorAudio::ConversorAudio(){}

byte* ConversorAudio::generarNotif(DataNotificacion info) {
      return getTextToSpeech(notif2str(info));
}
byte* ConversorAudio::getTextToSpeech(string text) {
      //byte* nuevo = (byte*)text.c_str();
    //return nuevo;
    // No implementar
}
ConversorAudio::~ConversorAudio(){}
```