# Programación 4

SOLUCION DICIEMBRE 2014

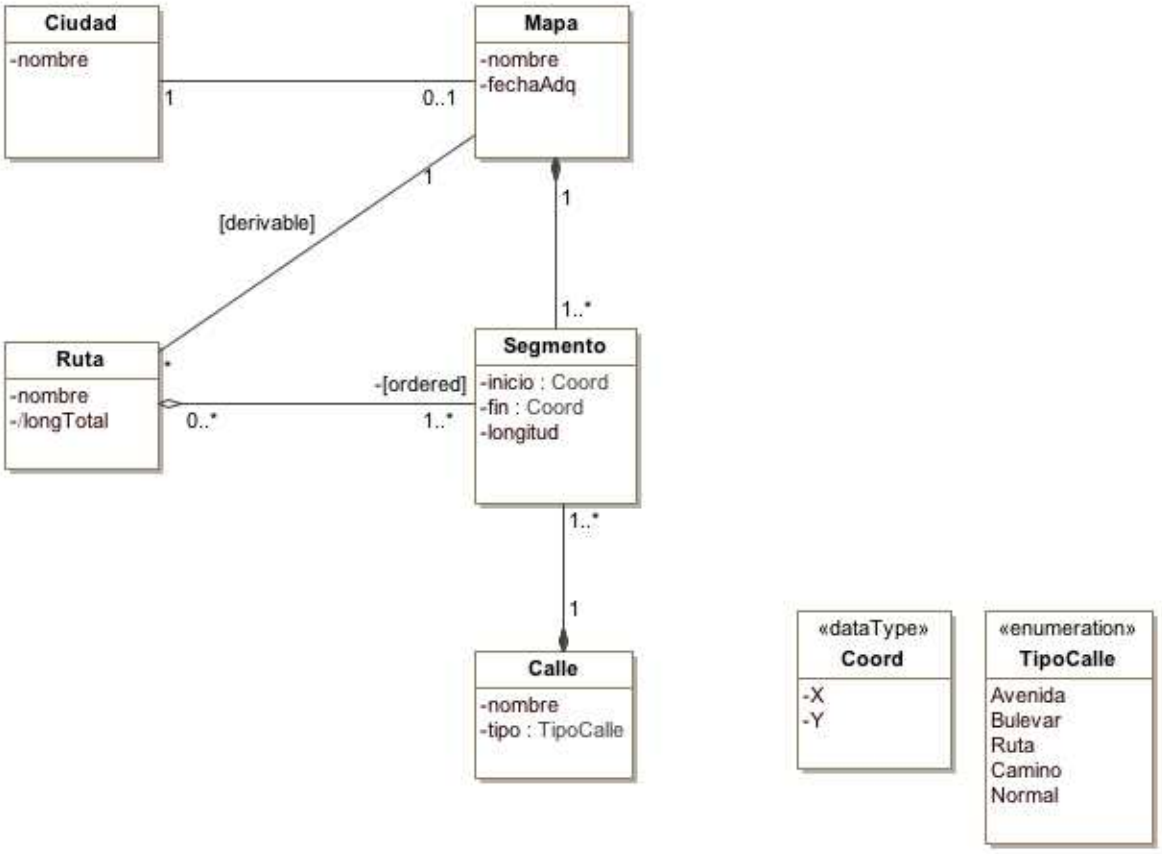
**Problema 1 (30 puntos)**

***Parte A:***

No es conveniente, ya que por ejemplo, en caso de requerir mantener memoria del sistema, esta quedaría distribuida entre diferentes controladores.

***Parte B:***

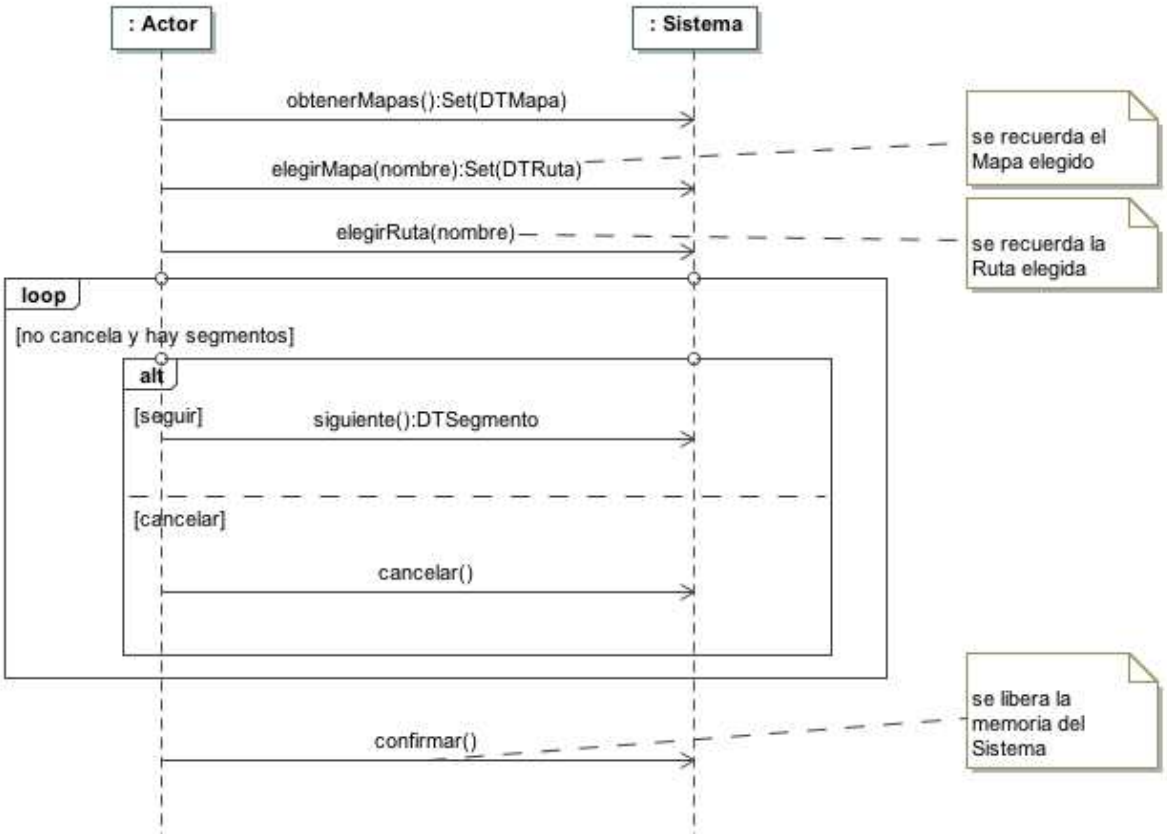
**a)**




Restricciones en lenguaje natural:

- Nombre identifica a Ciudad
- Nombre identifica a Mapa
- Nombre identifica a Ruta
- Nombre identifica a Calle
- El nombre del mapa debe coincidir con el nombre de la ciudad a la cual pertenece
- La longitud total de la ruta es la suma de las longitudes de sus segmentos
- Una ruta esta asociada al mismo mapa que contiene todos los segmentos que componen la ruta (asociación derivable)

**b)**

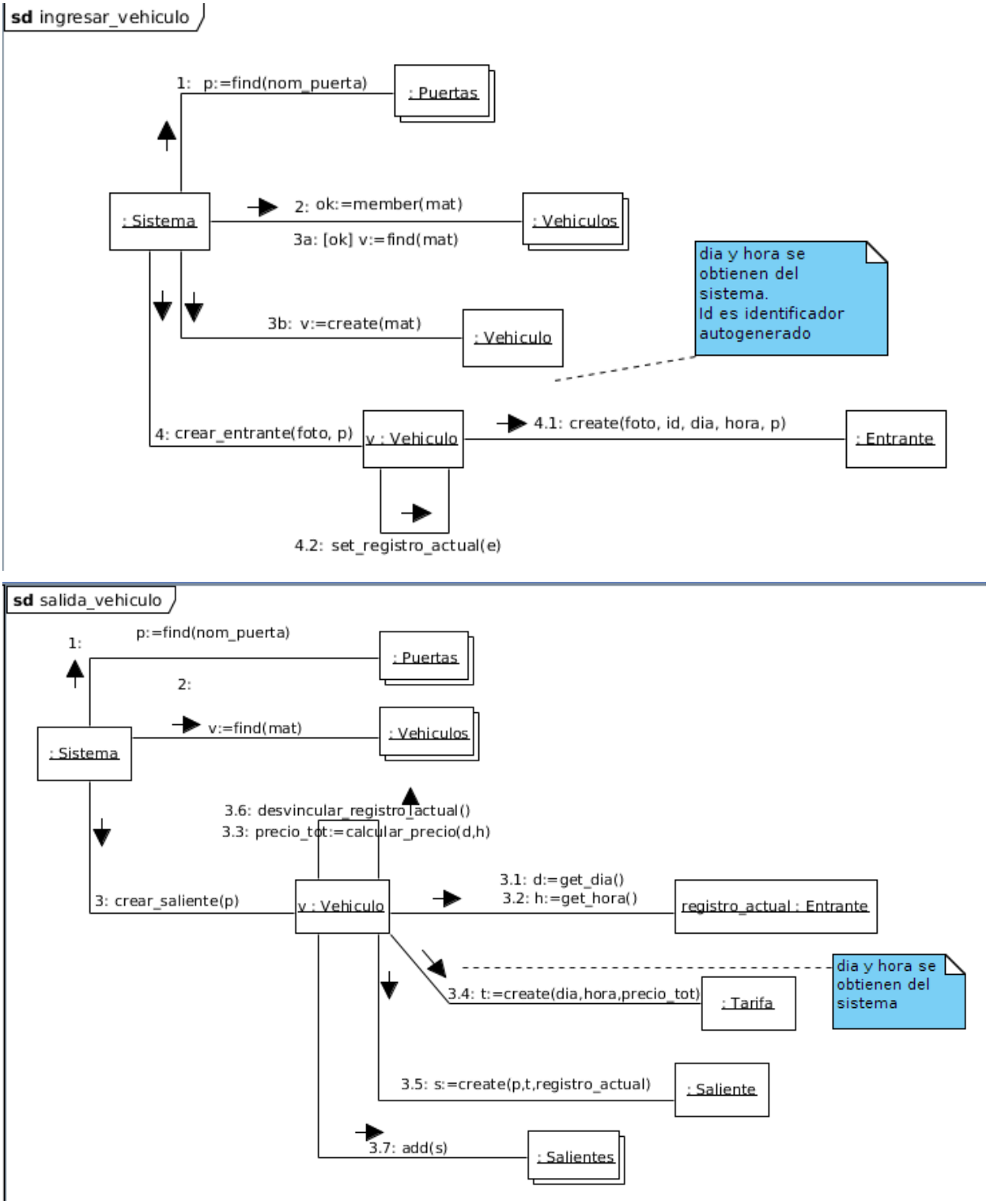


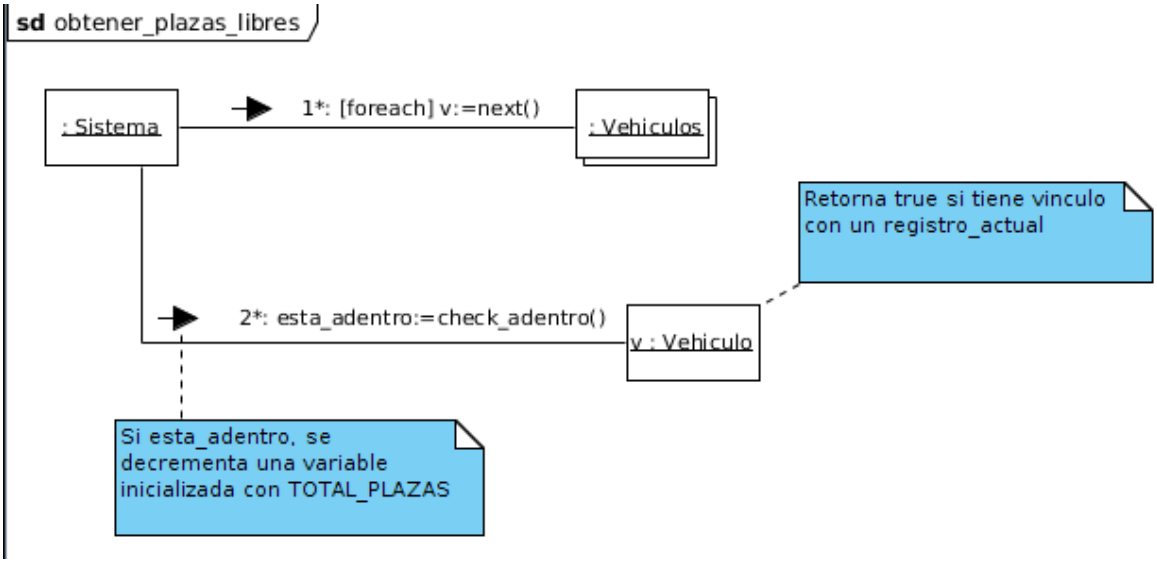

(se deben incluir los datatypes correspondientes a DTMapa, DTRuta, DTSegmento)

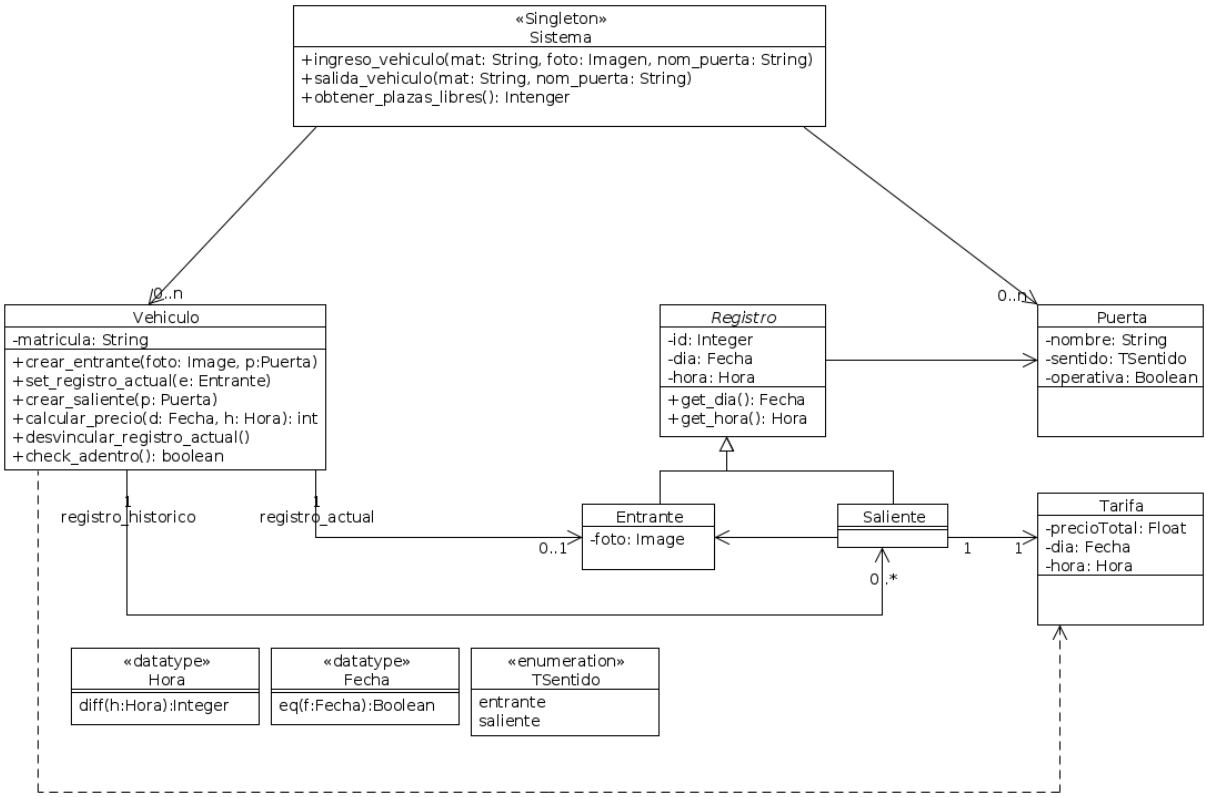
**Problema 2 (35 puntos)**

***Parte A:***

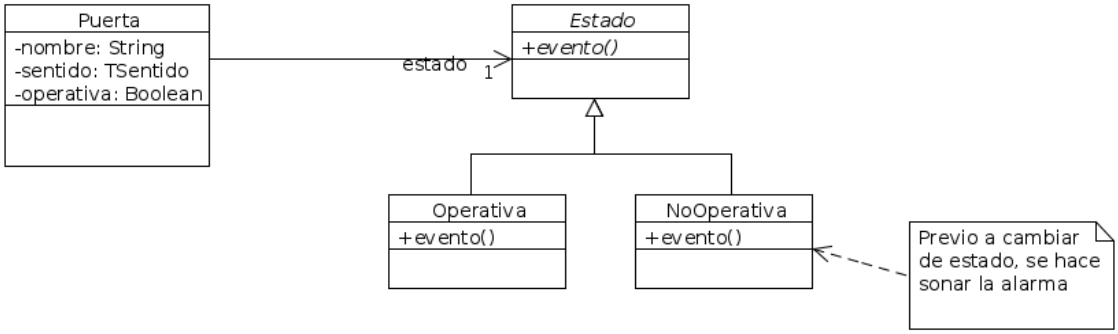
**a)**




sd obtener_plazas_libres
: Sistema
1*: [foreach] v:=next()
: Vehiculos
Retorna true si tiene vinculo con un registro_actual
2*: esta_adentro:=check_adentro()
v : Vehiculo
Si esta_adentro, se decrementa una variable inicializada con TOTAL_PLAZAS


«Singleton»
Sistema
+ingreso_vehiculo(mat: String, foto: Imagen, nom_puerta: String)
+salida_vehiculo(mat: String, nom_puerta: String)
+obtener_plazas_libres(): Intenger
0..n
0..n
Vehiculo
-matricula: String
+crear_entrante(foto: Image, p:Puerta)
+set_registro_actual(e: Entrante)
+crear_saliente(p: Puerta)
+calcular_precio(d: Fecha, h: Hora): int
+desvincular_registro_actual()
+check_adentro(): boolean
Registro
-id: Integer
-dia: Fecha
-hora: Hora
+get_dia(): Fecha
+get_hora(): Hora
Puerta
-nombre: String
-sentido: TSentido
-operativa: Boolean
1
registro_historico
1
registro_actual
Entrante
-foto: Image
0..1
Saliente
1
1
Tarifa
-precioTotal: Float
-dia: Fecha
-hora: Hora
0..*
«datatype»
Hora
diff(h:Hora):Integer
«datatype»
Fecha
eq(f:Fecha):Boolean
«enumeration»
TSentido
entrante
saliente

***Parte B:***




**Patrón State:**

- **Contexto:** Puerta
- **Estado:** Estado
- **EstadoConcreto:** Operativa, NoOperativa

**Problema 3 (35 puntos)**

***Parte A:***

```
class Tema{
public:
  void Tema();
  virtual ~ Tema () ;
  //getters y setters
  void agregarComentarioHijo(int idTema,int  idComentario);
  void agregarTema(String titulo);
  void eliminarComentario(int idTema, int idComentario);
 private:
   Set<Comentario *> comentarios;
   int id;
   String titulo;
};

Tema::~Tema(){
  set<Comentario *>::iterator it;
  for(it = comentarios.begin(); it != comentarios.end(); ++it){
    delete (*it);
   }
   comentarios.clear();
}

void Tema::agregarComentario(String comentario){
  Comentario * c=new Comentario(Util::getId,comentario);
  comentarios.insert(c);
}
```

```
void Tema:eliminarComentario(int idComentario){
  int encontre=0;
  set<Comentario *>::iterator it;
  it = comentarios.begin();
  while(it != comentarios.end() && encontre==0){
  if((*it).>getId()==idComentario){
    encontre=1;
    delete (*it);
  }else{
    ++it
  }
}

void Tema: agregarComentarioHijo(int idComentario,String comentarios);
  Comentario * c=new Comentario(Util::getId,comentario);
   int encontre=0;
   set<Comentario *>::iterator it;
   it = comentarios.begin();
   while(it != comentarios.end() && encontre==0){
     if((*it).getId()==idComentarios){
       encontre=1;
       (*it).getComentarios().insert(a);
     }else{
       ++it
   }
}

class ControladorTema
{
private:
  ControladorTerma();
  static ControladorTema* instance;
  Set<Tema *> temas;
public:
  ~ControladorTema();
  void eliminarComentario(int idTema ,int idComentario);
  void agregarComentarioHijo(int idTema,int idComentario,String
                             comentario);
  void agregarTema(String titulo,String correo)
  static ControladorTema *getInstance()
  {
     if(instance == NULL)
       instance = new ControladorTema();
     return instance;
   }
};

void ControladorTema::agregarComentarioHijo(int idTema,
                                            int dComentario,
                                            String comentario){
  int encontre=0;
  set<Tema *>::iterator it;
  it = temas.begin();
  while(it != temas.end() && encontre==0){
  if((*it).getId()==idTema){
    encontre=1;
    (*it).agregarComentarioHijo(idComentario,comentario);
  }else{
    ++it
    }
  }
```

```
}

void ControladorTema::eliminarComentario(int idTema ,
                                         int idComentario){
  int encontre=0;
  set<Tema *>::iterator it;
  it = temas.begin();
  while(it != temas.end() && encontre==0){
    if((*it).getId()==idTema){
      encontre=1;
      (*it).eliminarAporte(idAporte);
    }else{
      ++it
     }
   }
}

void ControladorTema::eliminarTema(int idTema){
  int encontre=0;
  set<Tema *>::iterator it;
  it = temas.begin();
  while(it != temas.end() && encontre==0){
    if((*it).getId()==idTema){
      encontre=1;
      delete (*it);
   }else{
     ++it
   }
  }
}

class Aporte{
private
  int id;
  Tema * tema;
 public:
   int getId(){return id;};
   void setId(int );
   virtual ~Aporte(){};
}

class Comentario: public class Aporte{
private:
  String comentario;
  Set<Comentario *> hijos;
  Comentario * padre;
 public:
  ….
  virtual ~Comentario();
}

Comentario::~Comentario(){
   set<Comentario *>::iterator it;
   for(it = comentarios.begin(); it != comentario.end(); ++it){
     delete (*it);
    }
    hijos.clear();
  }
```

*Parte B:*

**a)**

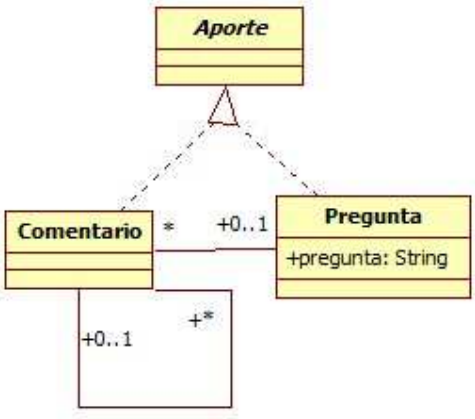
```
class ILog{
public
  virtual int enviar(String texto, String destino)=0;
};

class CorreoLog:public class ILog{
}

int CorreoLog:: enviar(String texto, String destino){
  try{
    MensajesUtil.getInstance().enviarCorreo(texto,destino);
    return 0;
  }catch(ExcepcionEnvioCorreo& e){
    return -1;
  }
}
```

**b)**

**i)**




**ii)**

```
class Pregunta: public class Aporte{
private String pregunta;
  Set<Comentario *> comentarios;
  public:
  ….
  virtual ~ Pregunta ();
}

Pregunta::~ Pregunta (){
  set<Comentario *>::iterator it;
  for(it = comentarios.begin(); it != comentario.end(); ++it){
    delete (*it);
  }
  comentarios.clear();
}
```

**iii)**

```
void AgregarPregunta(idTema:int,pregunta:String){
  int encontre=0;
  set<Tema *>::iterator it;
  it = temas.begin();
  while(it != temas.end() && encontre==0){
    if((*it).getId()==idTema){
      encontre=1;
      (*it).agregarPregunta(pregunta);
      ILog * log=new CorreoLog();
      log->enviar(pregunta, (*it).getCorreo());
    }else{
     ++it
     }
   }
}
```

Se agrega el pseudo atributo preguntas dentro de la clase temas:

```
void Tema::agregarPregunta(String pregunta){
  Pregunta* a=new Pregunta(Util::getId(),pregunta);
  preguntas.insert(a);
}
```