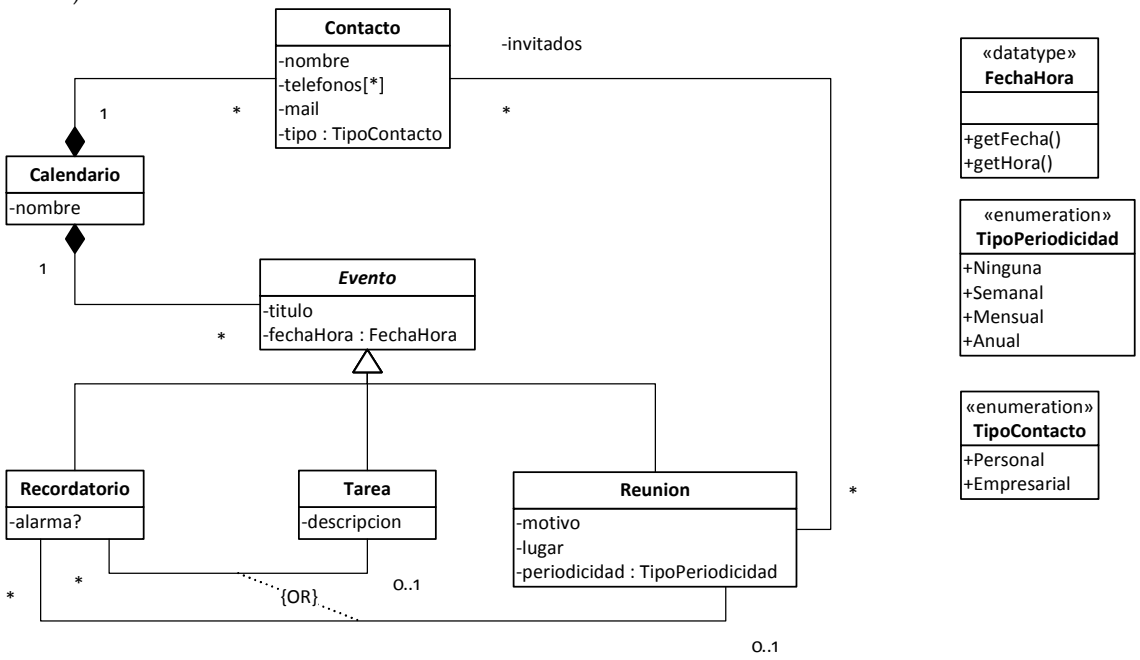
Programación 4

SOLUCIÓN EXAMEN DICIEMBRE 2011

Problema 1 (30 puntos)

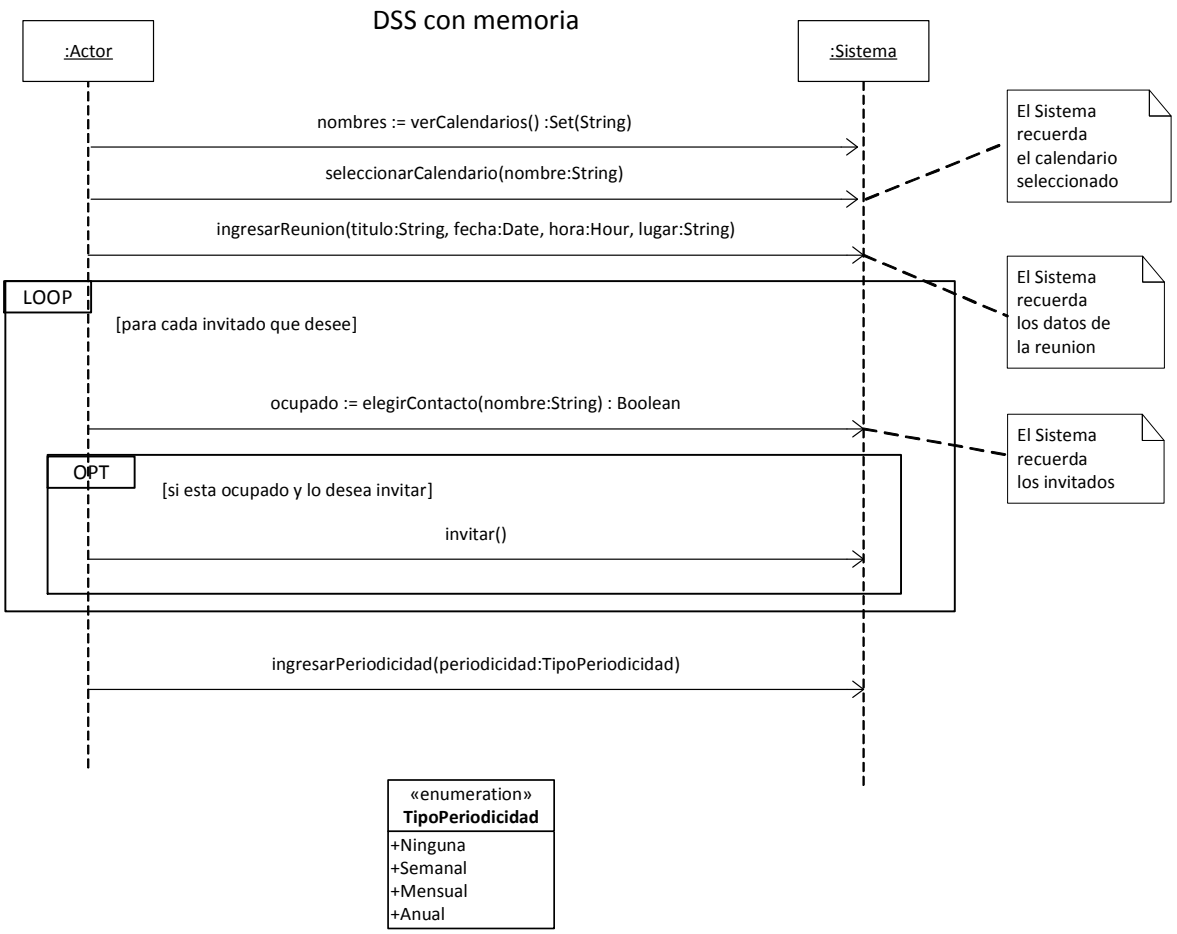
Parte i)



Restricciones:

- El nombre identifica al Calendario.
- El mail identifica al Contacto.
- Los invitados a una reunión deben pertenecer al mismo calendario que contiene la reunión.
- Un Recordatorio debe estar en el mismo calendario que la Tarea o Reunión que está recordando.

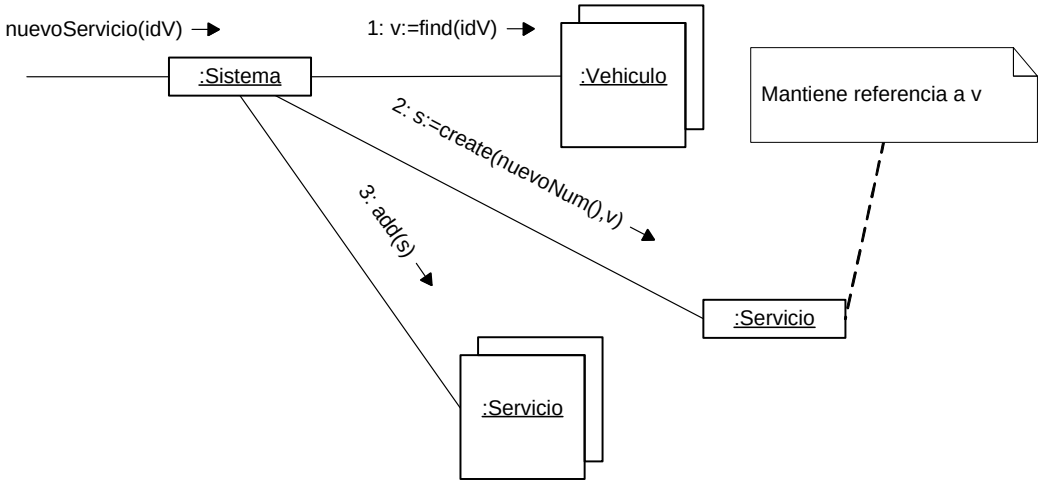
Parte ii)

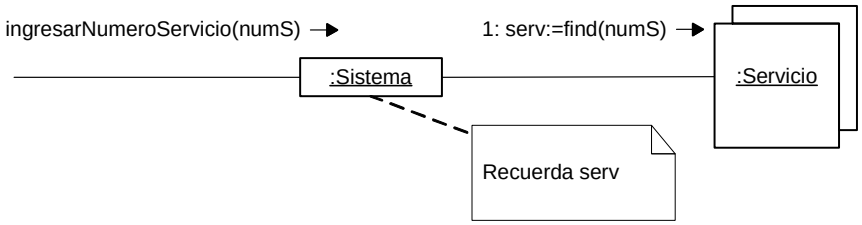

DSS con memoria
:Actor
:Sistema
nombres := verCalendarios() :Set(String)
seleccionarCalendario(nombre:String)
El Sistema recuerda el calendario seleccionado
ingresarReunion(titulo:String, fecha:Date, hora:Hour, lugar:String)
El Sistema recuerda los datos de la reunion
LOOP
[para cada invitado que desee]
ocupado := elegirContacto(nombre:String) : Boolean
El Sistema recuerda los invitados
OPT
[si esta ocupado y lo desea invitar]
invitar()
ingresarPeriodicidad(periodicidad:TipoPeriodicidad)
«enumeration»
TipoPeriodicidad
+Ninguna
+Semanal
+Mensual
+Anual

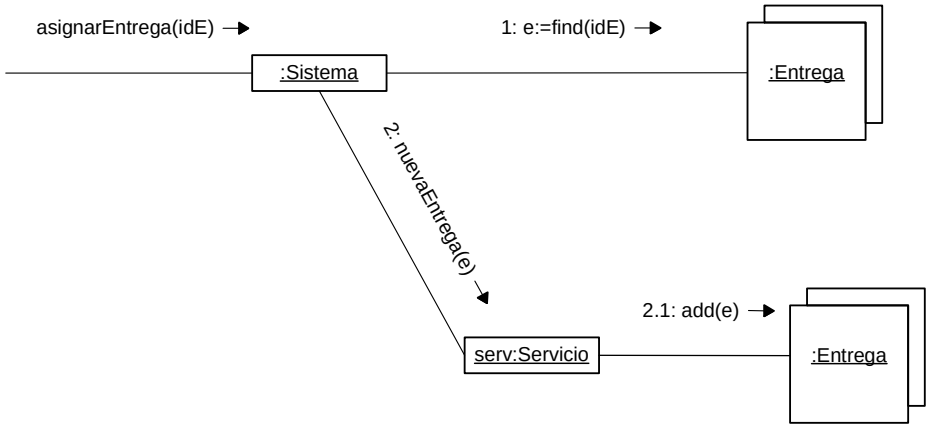
Problema 2 (35 puntos)

Parte A:

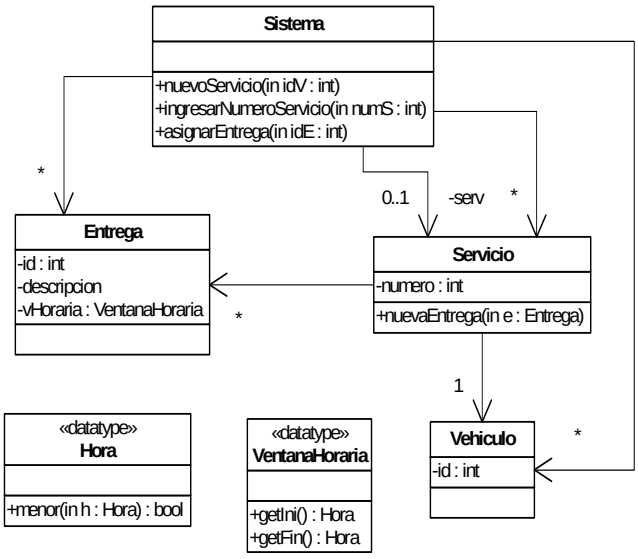
a)





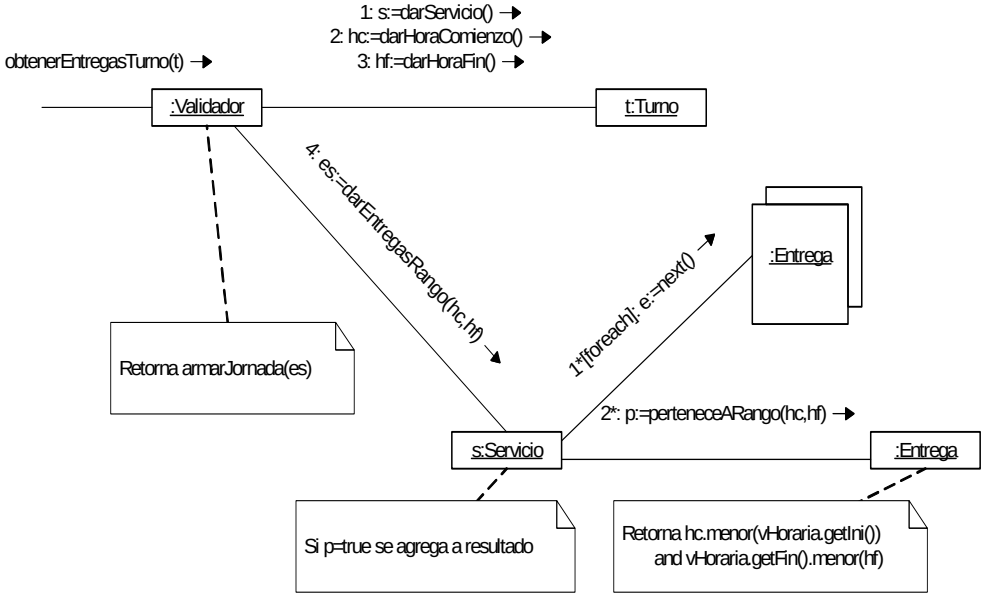


b)



Parte B:

a)



b)

Se utiliza el patrón Strategy.
Roles:
- Contexto: Validador
- Estrategia abstracta: Reglamentación
- Estrategias concretas: RegVigente y RegAlternativa

Problema 3 (35 puntos)

i.

```
// Foro.h


class Foro

{

private:

  static Foro *instance;

  IDictionary *usuarios;

  IDictionary *elementos;


  Foro();

public:

  static Foro *getInstance();


  void eliminarElemento(int idE);

  void crearMensaje(int idM, String nick, String mensaje);

};


// Foro.cpp


Foro *Foro::instance = NULL;


Foro::Foro()

{

  usuarios = new Dict();

  elementos = new Dict();

}
```

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

void Foro::eliminarElemento(int idE)
{
    IKey *k = new KeyInteger(idE);
    Elemento *e = (Elemento *) elementos->find(k);
    e->eliminar();
    elementos->remove(k);
    delete e;
    delete k;
}

void Foro::crearMensaje(int idM, String nick, String mensaje)
{
   IKey *k = new KeyString(nick);
   Usuario *u = (Usuario *) usuarios->find(k);
   Mensaje *m = new Mensaje(idM, u, mensaje);
   elementos->add(new KeyInteger(idM), m);
   delete k;
}

// Elemento.h
```

```
class Elemento: public ICollectible
{
private:
  Usuario *usr;
  int id;

public:
  Elemento(int id, Usuario *u);
  Usuario *getUsr();

  void eliminar();
  virtual void eliminarEspecifico() = 0;

  virtual ~Elemento();
};

// Elemento.cpp

Elemento::Elemento(int id, Usuario *u)
{
  this->id = id;
  this->usr = u;
  usr->agregar(this);
}

void Elemento::eliminar()
{
   usr->eliminar(this);
```

```
  eliminarEspecifico();
}

Elemento::~Elemento()
{
}

// Mensaje.h

class Mensaje: public Elemento
{
private:
 String texto;
 ICollection *comentarios;
public:
 Mensaje(int idM, Usuario *u, String texto);
 void eliminarEspecifico();
 ~Mensaje();
};

// Mensaje.cpp

Mensaje::Mensaje(int idM, Usuario *u, String texto):
      Elemento(idM, u)
{
  this->texto = texto;
  this->comentarios = new Col();
}
```

```
void Mensaje::eliminarEspecifico()
{
  IIterator *it = comentarios->getIterator();
  while(it->hasCurrent()){
    Comentario *c = (Comentario *) it->getCurrent();
    c->eliminar();
    delete c;
  }
  delete it;
}

Mensaje::~Mensaje()
{
    delete comentarios;
}

// Comentario.h

class Comentario: public Elemento
{
private:
  String contenido;
  Fecha fecha;
public:
  void eliminarEspecifico();
};
```

```
// Comentario.cpp

void Comentario::eliminarEspecifico()
{
  int n = getUsr()->getCantComentarios();
  getUsr()->setCantComentarios(n-1);
}
```

ii.

```
// Fecha.h

class Fecha
{
private:
 int dia, mes, anio;
public:
 Fecha(int dia, int mes, int anio);
 bool operator < (Fecha f);
 Fecha &operator = (Fecha f);
};

ostream& operator << (ostream& stream, Fecha f);
```