Programación 4

SOLUCIÓN FEBRERO 2015

Problema 1

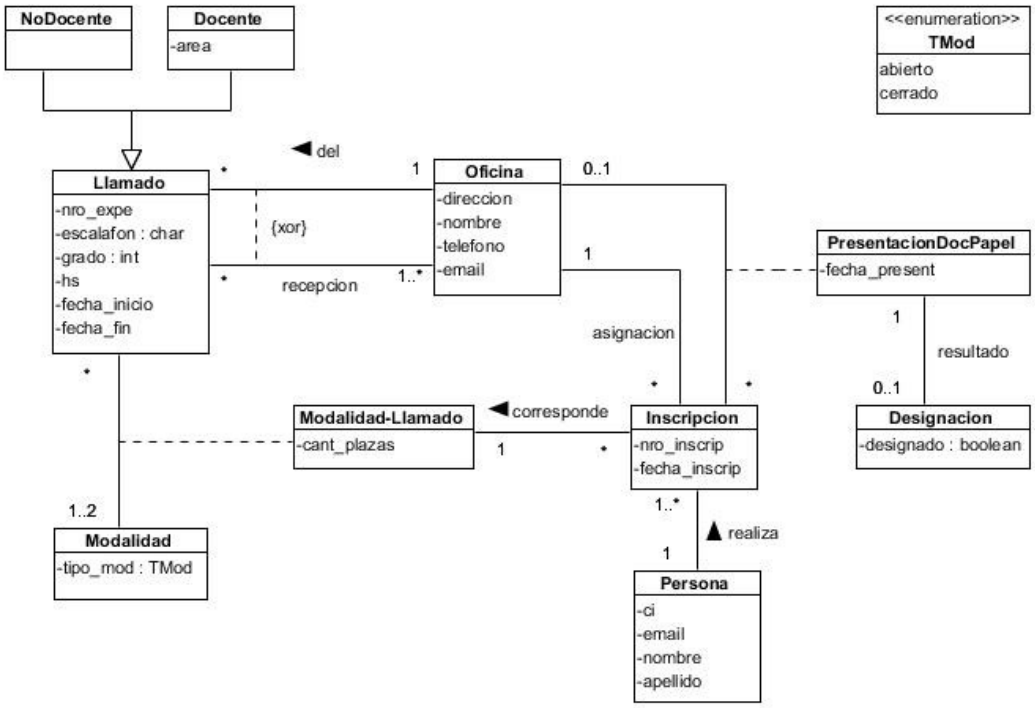
Parte A:

No es una instancia válida.

La inconsistencia está dada por los elementos l1: (p1, p2, 1/1/2014) y l3: (p1, p2, 1/1/2015). El par (p1, p2) sólo puede aparecer una vez en la instancia del tipo asociación.

Parte B:

a)



Restricciones no estructurales

Identificadores

1) Llamado::nro_expe, Oficina::direccion, Inscripcion::nro_inscrip

2) Persona::ci, Persona::email (diferentes; no es el par (ci, email))

Valores de atributos

3) [*] Dado un llamado L, L.hs > 0, dado una modalidad-llamado ml, ml.cant_plazas > 0

4) Dado un llamado L, si L.escalafon = 'G' entonces 1 <= L.grado <= 5

5) x es Docente si y sólo si x.escalafon = 'G'

6) Dado un llamado L, L.fecha_inicio <= L.fecha_fin

7) Dada una inscripción I,

I.corresponde.modalidad-llamado.fecha_inicio <= I.fecha_inscripcion <= I.corresponde.modalidad-llamado.I.corresponde.modalidad-llamado.fecha_fin

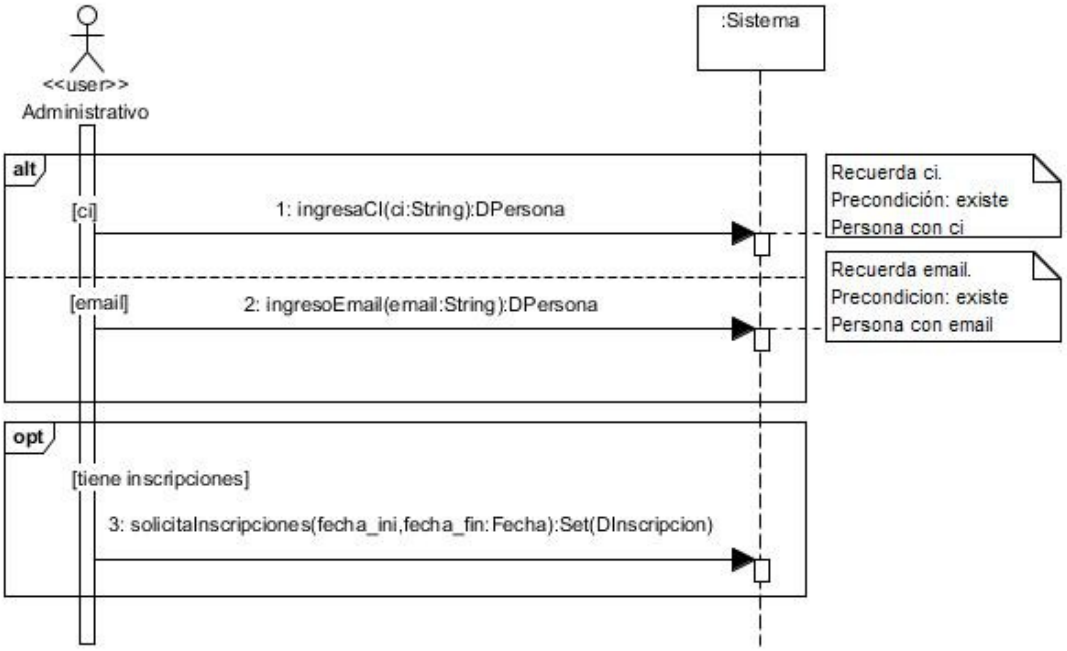
8) Dada una PresentacionDocPapel p, p.inscripcion.fecha_inscripcion <= p.fecha_present

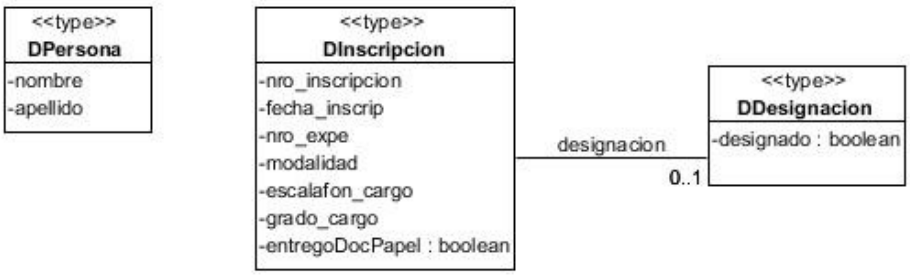
Restricciones entre asociaciones

9) La oficina en la que se presenta la documentación papel es la oficina receptora asignada en la inscripción

10) La oficina a la que es asignado en la inscripción a la modalidad de un llamado es una oficina receptora definida para el llamado.

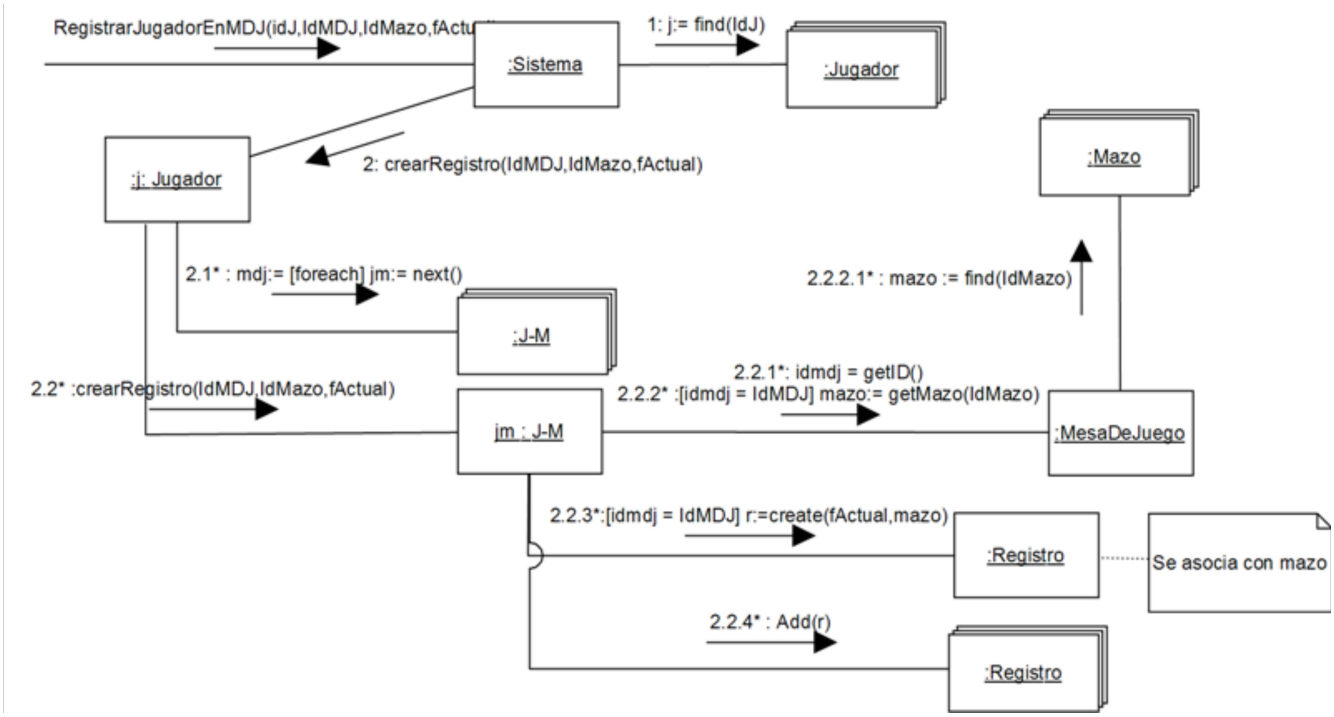
b)

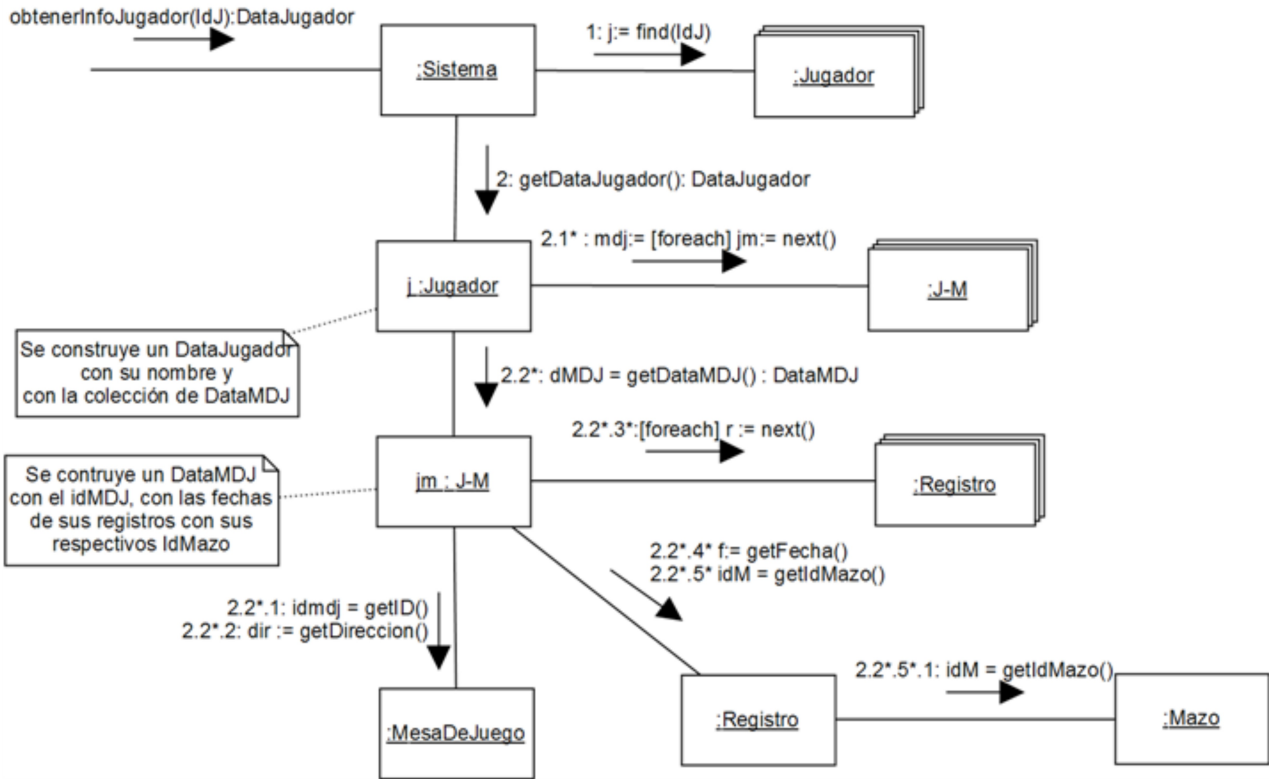




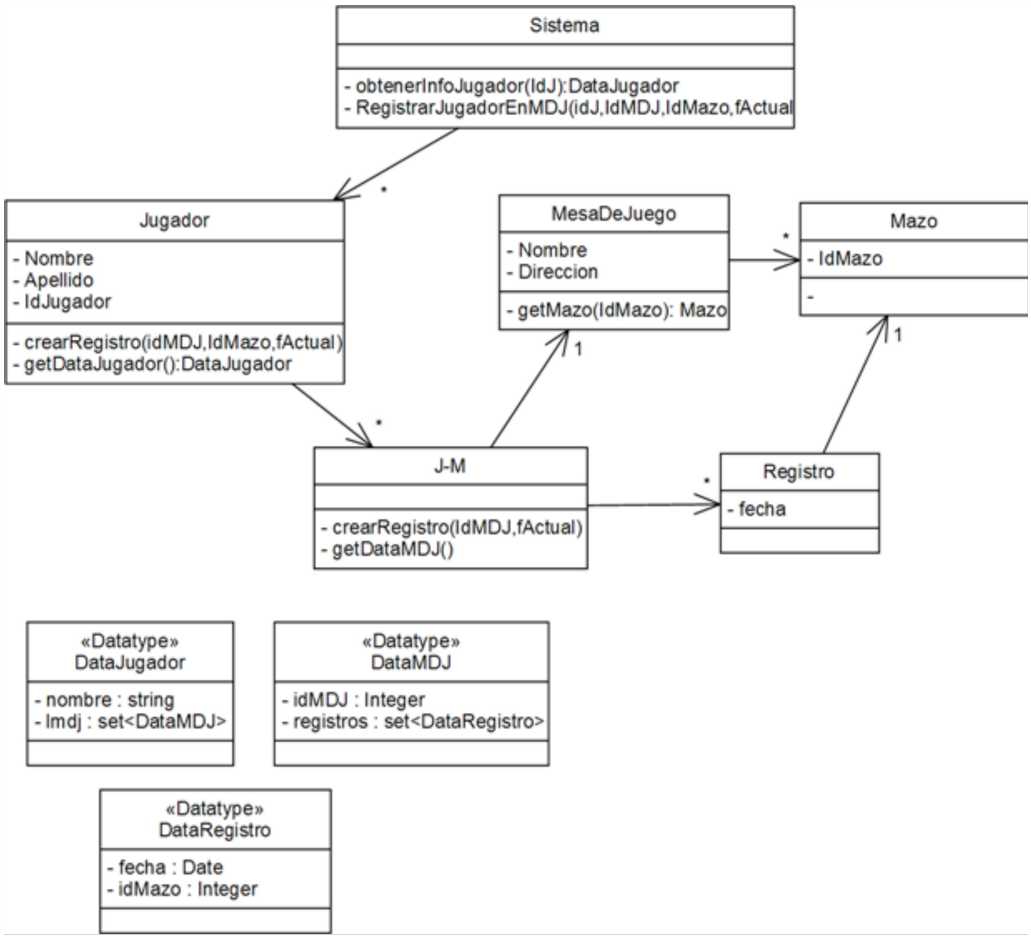
Problema 2

a)


RegistrarJugadorEnMDJ(idJ,IdMDJ,IdMazo,fActu
:Sistema
1: j:= find(IdJ)
:Jugador
:j: Jugador
2: crearRegistro(IdMDJ,IdMazo,fActual)
:Mazo
2.1* : mdj:= [foreach] jm:= next()
2.2.2.1* : mazo := find(IdMazo)
:J-M
2.2* :crearRegistro(IdMDJ,IdMazo,fActual)
2.2.1*: idmdj = getID()
2.2.2* :[idmdj = IdMDJ] mazo:= getMazo(IdMazo)
jm : J-M
:MesaDeJuego
2.2.3*:[idmdj = IdMDJ] r:=create(fActual,mazo)
:Registro
Se asocia con mazo
2.2.4* : Add(r)
:Registro


obtenerInfoJugador(IdJ):DataJugador
:Sistema
1: j:= find(IdJ)
:Jugador
2: getDataJugador(): DataJugador
2.1* : mdj:= [foreach] jm:= next()
j :Jugador
:J-M
Se construye un DataJugador con su nombre y con la colección de DataMDJ
2.2*: dMDJ = getDataMDJ() : DataMDJ
2.2*.3*:[foreach] r := next()
Se contruye un DataMDJ con el idMDJ, con las fechas de sus registros con sus respectivos IdMazo
jm : J-M
:Registro
2.2*.4* f:= getFecha()
2.2*.5* idM = getIdMazo()
2.2*.1: idmdj = getID()
2.2*.2: dir := getDireccion()
2.2*.5*.1: idM = getIdMazo()
:MesaDeJuego
:Registro
:Mazo

b)


Sistema
- obtenerInfoJugador(IdJ):DataJugador
- RegistrarJugadorEnMDJ(idJ,IdMDJ,IdMazo,fActual
Jugador
- Nombre
- Apellido
- IdJugador
- crearRegistro(idMDJ,IdMazo,fActual)
- getDataJugador():DataJugador
MesaDeJuego
- Nombre
- Direccion
- getMazo(IdMazo): Mazo
Mazo
- IdMazo
-
J-M
- crearRegistro(IdMDJ,fActual)
- getDataMDJ()
Registro
- fecha
«Datatype»
DataJugador
- nombre : string
- lmdj : set<DataMDJ>
«Datatype»
DataMDJ
- idMDJ : Integer
- registros : set<DataRegistro>
«Datatype»
DataRegistro
- fecha : Date
- idMazo : Integer
*
1

Problema 3

```
// Servicio.H
class Servicio: public Publicacion
{
  private:
    string nombre_anunciante;
    string tel_anunciante;
    string email_anunciante;

  public:
     ~Publicacion();

    FormaPago* obtenerFormaDePago(string preg);
    int obtenerRating();
};

// AdminPublicaciones.H
class AdminPublicaciones
{
  private:
    map<int,Publicacion*> Publicaciones;
     static AdminPublicaciones* instance;

  public:
    ~AdminPublicaciones();
    static AdminPublicaciones *getInstance();

    void realizarPregunta(int pub_id , string preg);
    set<Producto*> obtenerProductos();
    set<Servicio*> obtenerServicio();
    void borrarPublicacion(int pub_id);
    int obtenerIdNuevaPublicacion();
    int obtenerIdNuevaPregunta(int pub_id);
};

// AdminPublicaciones.CPP
AdminPublicaciones *AdminPublicaciones::getInstance()
   {
    if(instance == NULL)
       instance = new AdminPublicaciones();
    return instance;
   }

void AdminPublicaciones::realizarPregunta(int pub_id , string preg)
  {
    Publicacion *pub = Publicaciones.find(pub_id)->second;
    pub.realizarPregunta(preg);
  };

set< Producto*> AdminPublicaciones::obtenerProductos()
  {
    set <Producto*> res;
    for (map<int, Publicacion*>::iterator it=Publicaciones.begin();
it!=Publicaciones.end(); ++it)
    if(dynamic_cast<Producto*>(it->second) != NULL)
      res.insert(dynamic_cast<Producto*>(it->second));
    return res;
  };
```

```
void AdminPublicaciones::borrarPublicacion(int pub_id)
  {
    std::map<int,Publicacion >::iterator it =
Publicaciones.find(pub_id);
    Publicaciones.erase(it);
  };

// Publicacion.H
class Publicacion
{
  private:
    map<int, Pregunta*> Preguntas;
    map<int, Pregunta*> Pendientes;
    int id;
    string lema;
    string descripcion;
    Date fecha_publicado;

  public:
    virtual ~Publicacion();
    void responderPregunta(int preg_id, string resp);
    void realizarPregunta(string preg);
    virtual FormaPago* obtenerFormaDePago(string preg) = 0;
};

// Publicacion.CPP
void Publicacion::realizarPregunta(string preg)
  {
    AdminPublicaciones *ap = AdminPublicaciones.getInstance();
    int preg_id = ap->obtenerIdNuevaPregunta(pub_id);
    Pregunta *p = new Pregunta( preg_id, fecha, preg);
    Pendientes[preg_id] = p;
  };

void Publicacion::responderPregunta(int preg_id, string resp){
    std::map<int,Preguntas >::iterator it = Pendientes.find(preg_id);
    Pendientes.erase(it->second);
    Preguntas[it->first]=it->second;
    Respuesta *r = new Respuesta(resp);
    it->second->addRespuesta(r);
}

Publicacion::~Publicacion(){
    for (map<int, Preguntas*>::iterator it=Preguntas.begin(); it!
=Preguntas.end(); ++it)
      delete it->second;
    for (map<int, Pendientes*>::iterator it=Pendientes.begin(); it!
=Pendientes.end(); ++it)
      delete it->second;
  };

// Usuario.CPP
void Usuario::destruirPublicacion(int pub_id)
  {
    AdminPublicaciones.getInstance()->borrarPublicacion(pub_id);
    std::map<int,Publicacion >::iterator it =
Publicaciones.find(pub_id);
    Publicaciones.erase(it);
    delete it->second;
  };
```

```
// Pregunta.H
class Pregunta
{
  private:
    int id;
    Date fecha;
    string pregunta;
    Respuest* resp;
  public:
    ~Pregunta;
    void addRespuesta(Respuesta *);
}

// Pregunta.CPP
Pregunta::Pregunta(int pub_id)
{
      this.id = pub_id;
//fecha queda con el valor correcto usando el constructor por defecto.
};

void Pregunta::addRespuesta(Respuesta * r)
{
      this->resp = r;
};

Pregunta::~Pregunta(int pub_id)
{
  if this->respuesta != NULL
    delete this->respuesta;
};
```