Programación 4

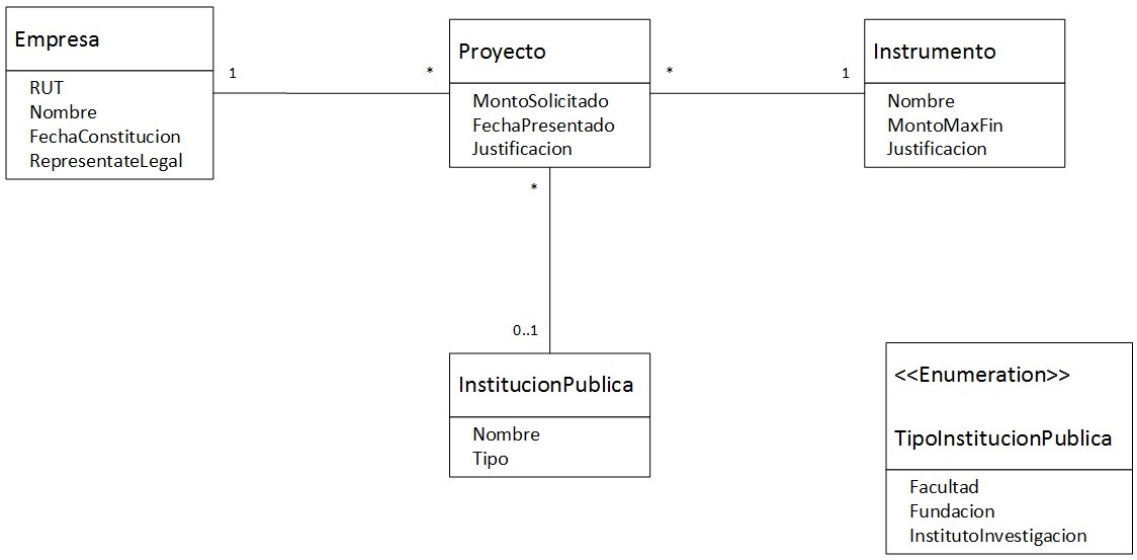
EXAMEN DICIEMBRE 2013

SOLUCIÒN

Problema 1 (30 puntos)

a)
Modelar el Dominio del Problema y Especificar el Comportamiento del Sistema.

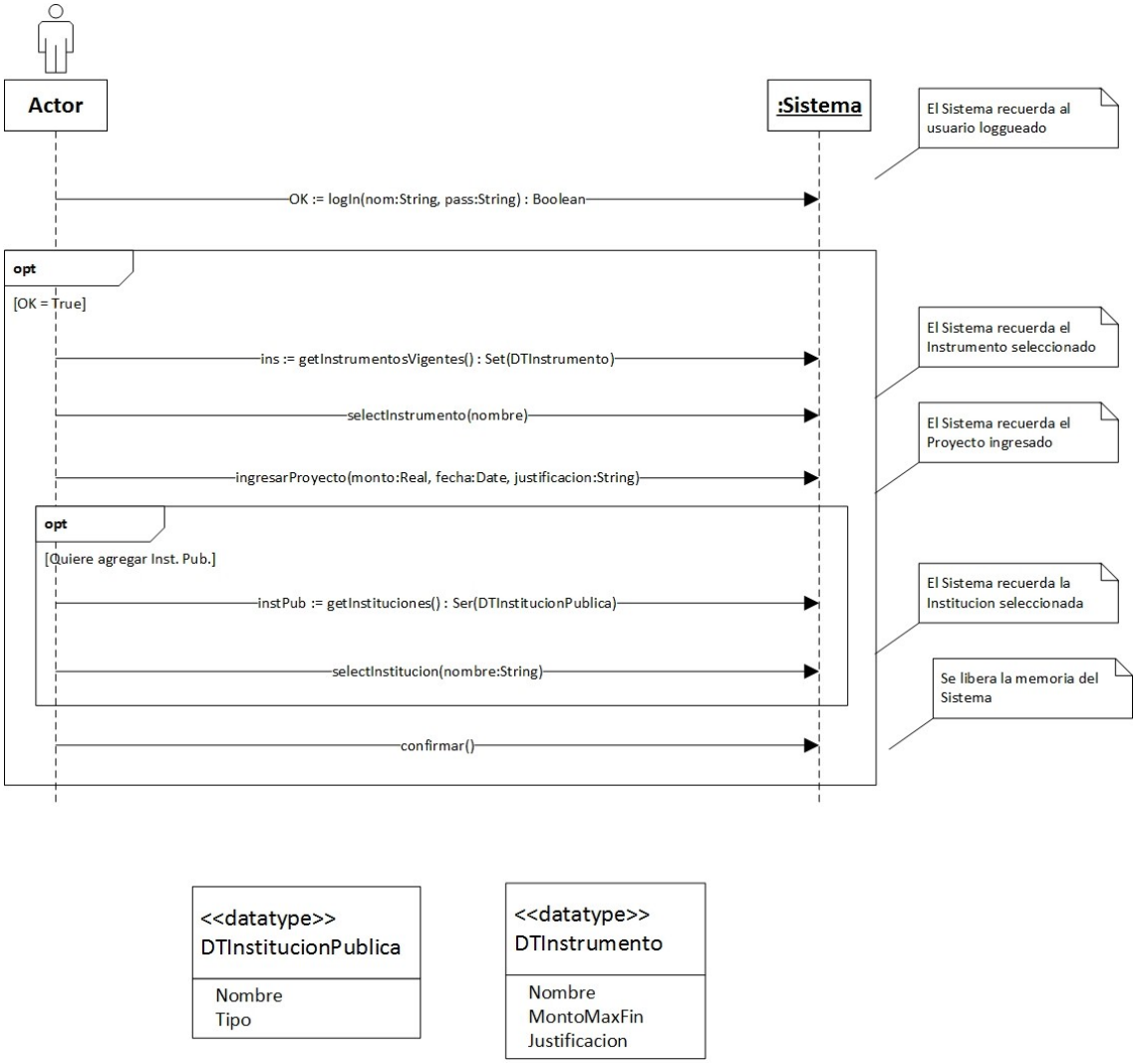
b) i.



Restricciones:

- El RUT identifica a la empresa.
- El Nombre identifica a la institución pública.
- El Nombre identifica al instrumento.
- El MontoSolicitado del Proyecto debe ser menor o igual al MontoMaxFin del Instrumento.

ii.



Problema 2 (35 puntos)

a)

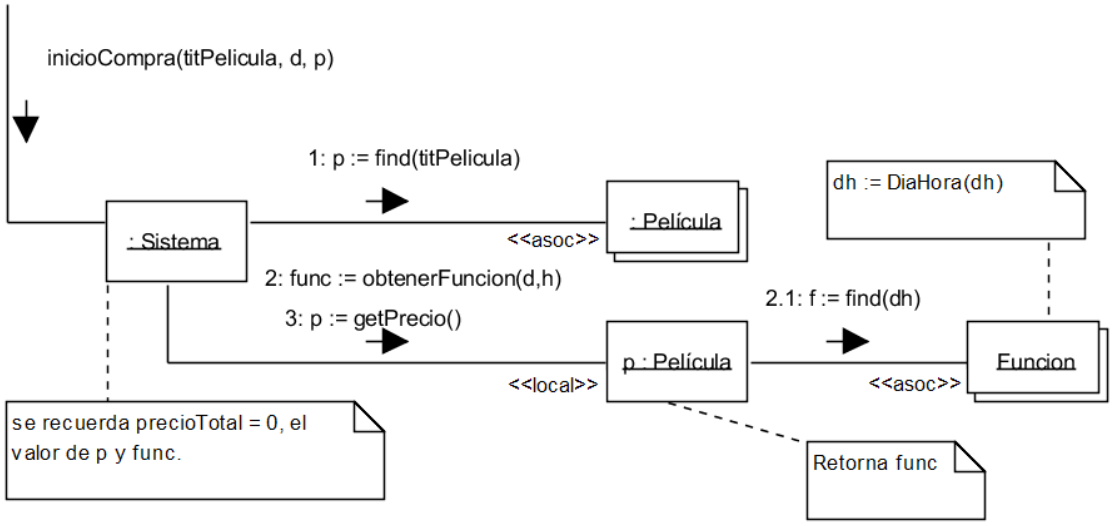
i.

Los GRASP son criterios que ayudan a resolver el problema de asignar responsabilidades durante la etapa de Diseño. (Ver teórico)

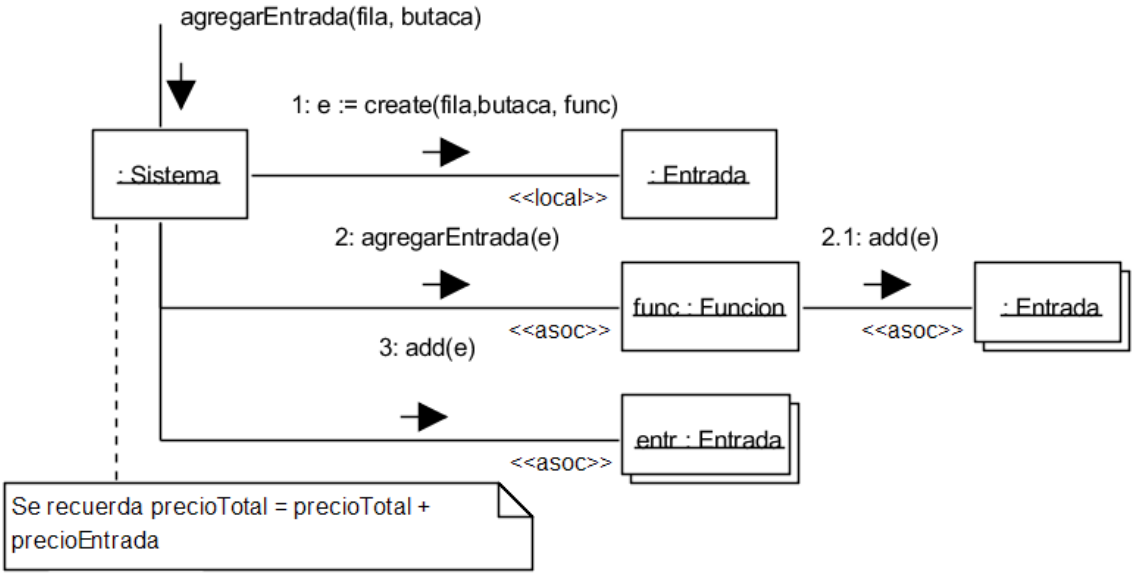
ii. Bastaría con nombrar 3. (Ver teórico)

i. No hables con extraños
ii. Expert
iii. Creator
iv. Controller
v. Bajo Acoplamiento
vi. Alta Cohesión

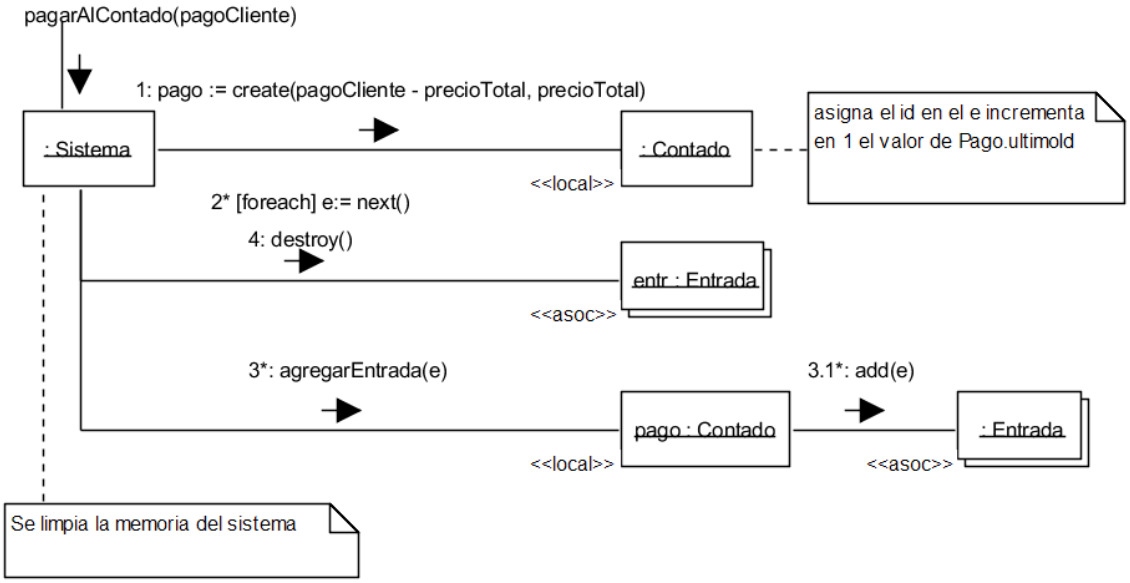
b) i.



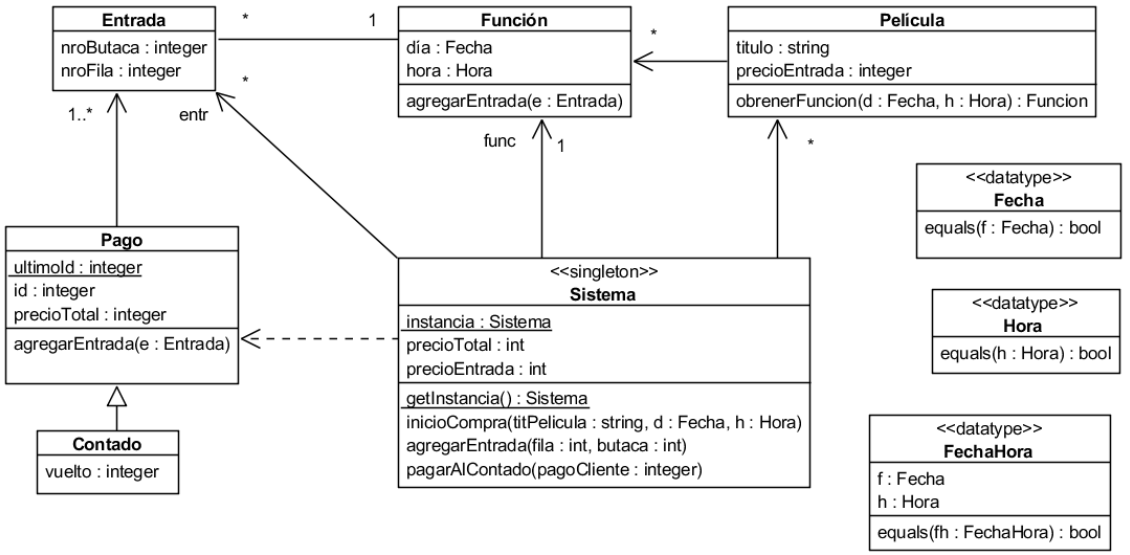
Nota: En el diagrama de `inicioCompra()` el mensaje 2.1 se puede reemplazar por una iteración en la colección de funciones para hacer la búsqueda de la entrada



Se eligió hacer links en ambos sentidos entre `e` y `func`, sin embargo basta la asociación de en un solo sentido para cumplir las postcondiciones de la operación `agregarEntrada()`



ii.



c) El patrón más conveniente es *Strategy* porque permite intercambiar formas de calcular descuentos implementando la interfaz *CalculadorDescuento*, así como crear nuevas clases sin modificar el código existente. Los roles serían

Estrategia: CalculadorDescuento

Estrategia Concreta: Las clases que implementen la interfaz CalculadorDescuento

Contexto: Puede ser la clase Sistema o cualquier clase que utilice a la estrategia, como la clase Pago.

Problema 3 (35 puntos)

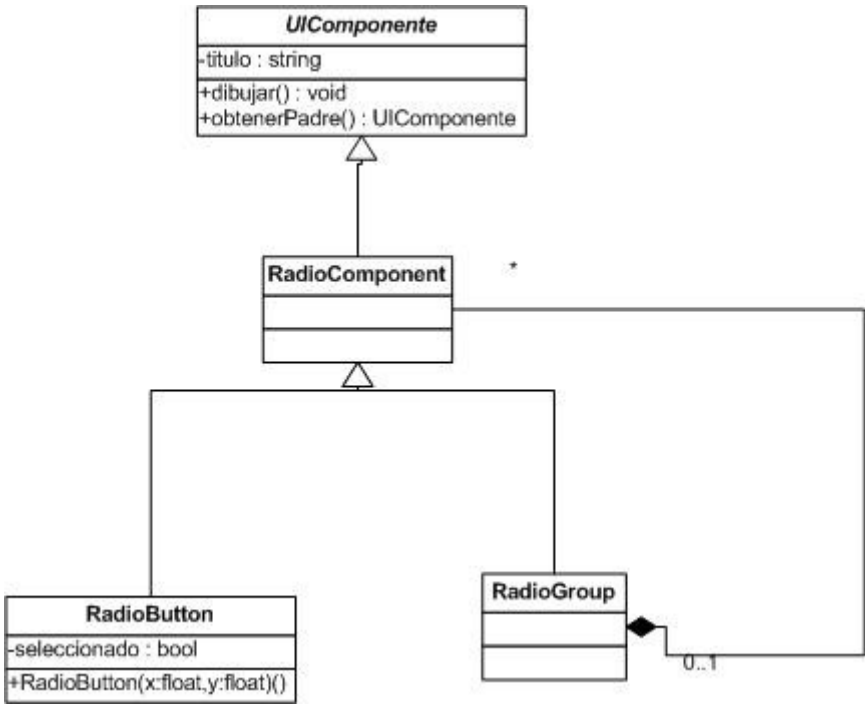
Parte A

```
 class UIComponent: public XComponenteGrafico{

public:
        virtual dibujar()=0:
        virtual UIComponente*  obtenerPadre()=0;
        virtual ~UIComponente() {};
private:

        String titulo;
        UIComponent*  padre;

};




class Boton : public UIComponent{

public:
        Boton(x:float,y:float, ancho:float,alto:float);
         ~Boton() {};
prívate:
         XFormaGeometrica* forma;


};

Boton::Boton(x:float,y:float, ancho:float,alto:float){
  forma=new XRectanculo();
  forma->x=x;
  forma->y=y;
 forma->ancho=ancho;
 forma->alto=alto;
}

Void Boton::dibujar(){
  this->dibujar(forma);
};

Boton::~Boton(){
 delete forma;
};


class RadioButton : public UIComponent{

public:
        RadioButton(x:float,y:float);
         ~ RadioButton () {};

prívate:
  bool seleccionado;
  XFormaGeometrica* forma;

};
```

```
RadioButton::RadioButton(x:float,y:float){
    forma=new XCirculo();
   forma->x=x;
   forma->y=y;
   forma->radio=N;
  //seleccionado es false por defecto
}

Void RadioButton::dibujar(){
 if seleccionado
   this->forma->relleno=true;
else
 this->forma->relleno=false;

  this->dibujar(forma);
};

RadioButon::~RadioButon(){
 delete forma;
};

class Contenedor : public UIComponent{
public:
          agregarComponente(UIComponent* c)
          ~ Contenedor () {};

Private:
  Set<UIComponent *> hijos;
};

Void Contenedor::agregarComponente(UIComponent* c){
 Hijos.insert(c);
}

Contenedor::~Contenedor(){
 set<UIComponent *>::iterator it;
for(it = hijos.begin(); it != hijos.end(); ++it){
 delete (*it);
}
 hijos.clear();
}

Contenedor::dibujar(){
 set<UIComponent *>::iterator it;
for(it = hijos.begin(); it != hijos.end(); ++it){
 (*it).>dibujar();
}

};
```

Parte B

i.




```
class RadioButton : public RadioComponent{

public:
        RadioButton(x:float,y:float);
         ~ RadioButton () {};

Prívate:
  Bool seleccionado;

};

class RadioComponent: public UIComponent{

public:
        virtual ~RadioComponente() {};
private:

        RadioComponent*  padre;

};

class RadioGroup : public RadioComponente{

public:
        agregarRadio(RadioComponent* c)
        ~ RadioGroup () {};

Prívate:
  Set<RadioComponent *> hijos;
```

```
};

Void RadioGroup::agregarRadio(RadioComponent* c){
 Hijos.insert(c);
}

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

ii.

El patrón utilizado es el composite
RadioComponent cumple el rol de componente
RadioButton cumple el rol de hoja
RadioGroup cumple el rol de compuesto.