Programación 4

PARCIAL FINAL EDICIÓN 2010

SOLUCIÓN

Por favor siga las siguientes indicaciones:

- Escriba con lápiz
- Escriba las hojas de un solo lado
- Escriba su nombre y número de documento en todas las hojas que entregue
- Numere las hojas e indique el total de hojas en la primera de ellas
- Recuerde entregar su numero de parcial junto al parcial

Problema 1 (30 puntos)

a) Conteste brevemente: ¿qué es un evento del sistema en el contexto de la etapa de Análisis – Comportamiento del Sistema?

R: Es un estímulo externo, generado por un actor, ante el cual el sistema debe reaccionar.

b) Considere los dos siguientes modelos de dominio parciales. Estos modelan una realidad en donde un país está compuesto administrativamente por departamentos y a su vez éstos por municipios, lo que determina una dependencia jerárquica a nivel administrativo.

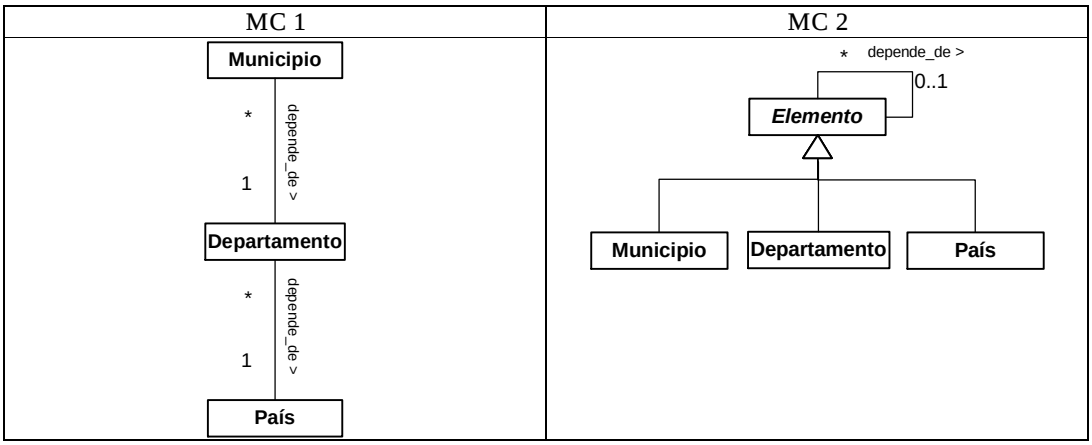
i. Indique las restricciones en lenguaje natural que deben incluirse para representar la realidad descrita.

R:

- MC1: no necesita restricciones.
- MC2 requiere:
  - Los Municipios dependen de un Departamento.
  - Los Departamentos dependen de un País.
  - Los Países no dependen de otro Elemento.

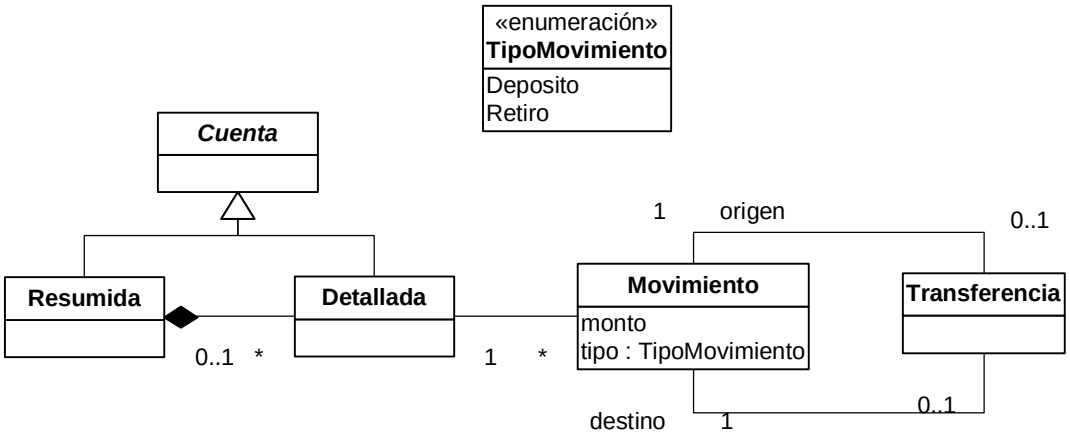
ii. ¿Qué modificaciones haría, en cada caso, para incorporar una nueva relación, análoga a la anterior entre municipios, departamentos y países?

R: en el MC1 habría que duplicar las asociaciones, sin necesidad de agregar nuevas restricciones, mientras que en el MC2 habría que agregar una nueva auto-relación en Elemento y las correspondientes restricciones (análogas a las del MC2 en la pregunta anterior).



c) A partir de la siguiente descripción de una realidad realice el modelo de dominio con restricciones en lenguaje natural.

> "***El banco URUBANC permite manejar cuentas detalladas para retirar o depositar dinero, indicando el monto de cada movimiento. Asimismo, el banco permite tener cuentas resumidas, las cuales resumen la información de cuentas detalladas, por lo que una cuenta resumida no puede contener directamente ningún movimiento (ni depósitos ni retiros). Las transferencias implican un retiro en la cuenta de origen y un depósito (del mismo monto) en la cuenta de destino. Ambas cuentas deben ser diferentes y detalladas.***"

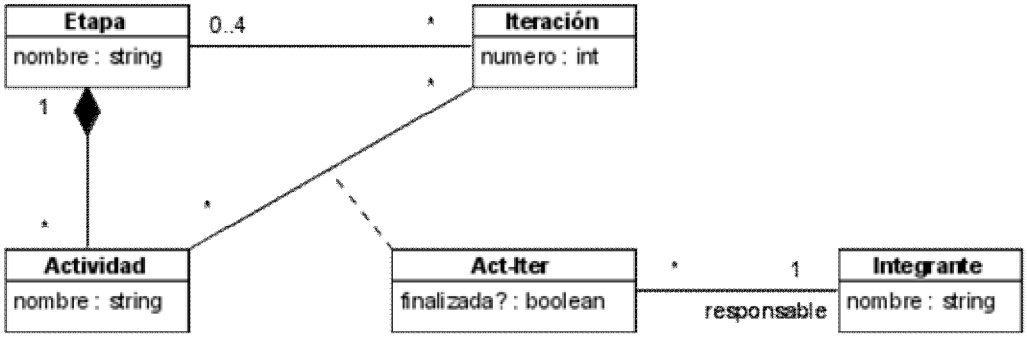


Restricciones:

- Los movimientos de origen y destino de una transferencia poseen igual monto, diferente tipo y está relacionados con diferentes cuentas.
- El movimiento de origen de una transferencia es de tipo Retiro mientras que el de destino es de tipo Deposito.

Problema 2 (30 puntos)

Se está construyendo un sistema de seguimiento de un proyecto de software que siga el modelo de proceso Iterativo e Incremental como el visto en el curso. Durante una primera etapa se analizó el mismo generándose el modelo de dominio que se puede observar en la figura siguiente.



El sistema permite definir las etapas de un proceso (ej: Análisis, Diseño, etc.) y las actividades que se realizan en estas etapas (ej: Modelado de Dominio y Especificación de Comportamiento en la etapa de Análisis). Cabe señalar que el nombre de la actividad la identifica dentro de una etapa, pudiendo haber actividades del mismo nombre en diferentes etapas. Además permite definir un conjunto de iteraciones y asignar las etapas a dichas iteraciones. A su vez, para cada iteración se debe definir qué actividades se realizarán ya que no todas las actividades de una etapa se tienen porqué realizar en cada iteración. Para cada actividad en cada iteración, se conoce si la actividad fue realizada o no y quién es el integrante del equipo del proyecto que es responsable por dicha actividad.

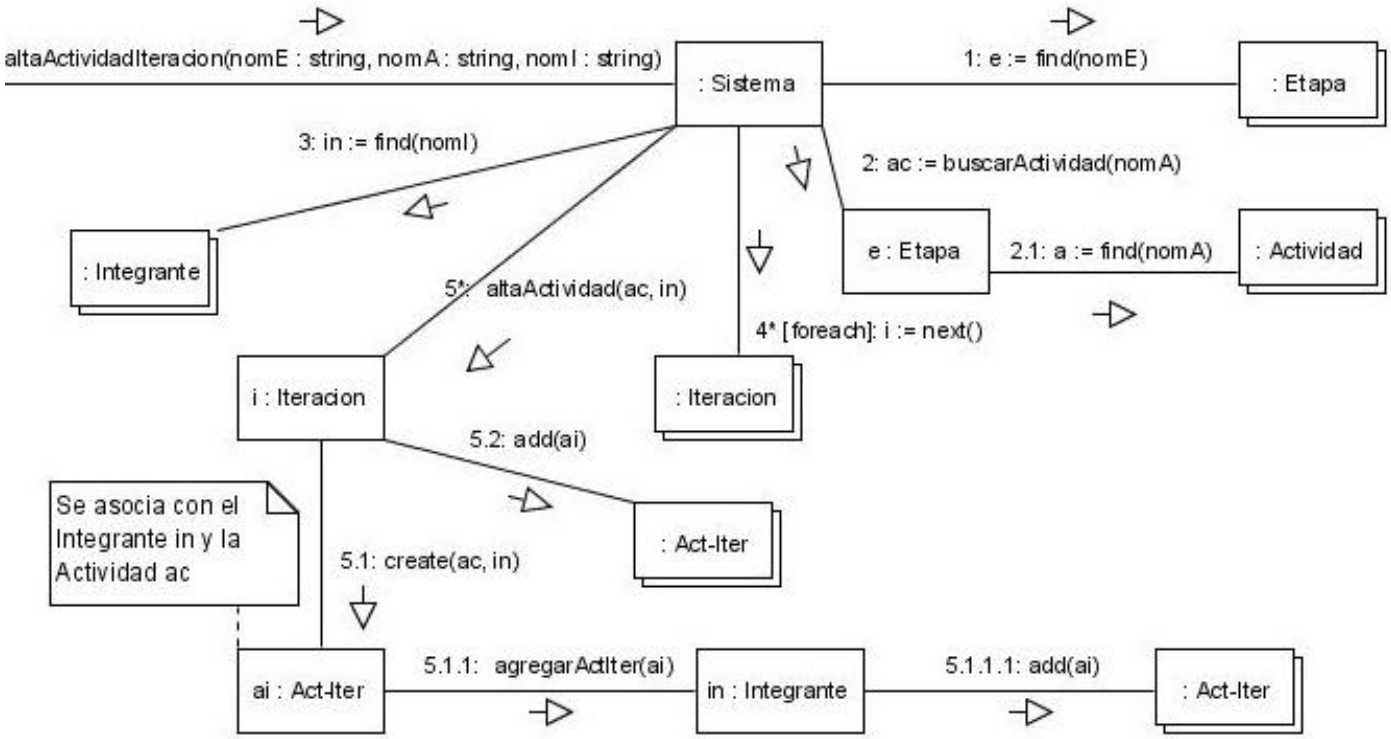
Adicionalmente al Modelo de Dominio, se definieron dos operaciones que están definidas por los siguientes contratos.

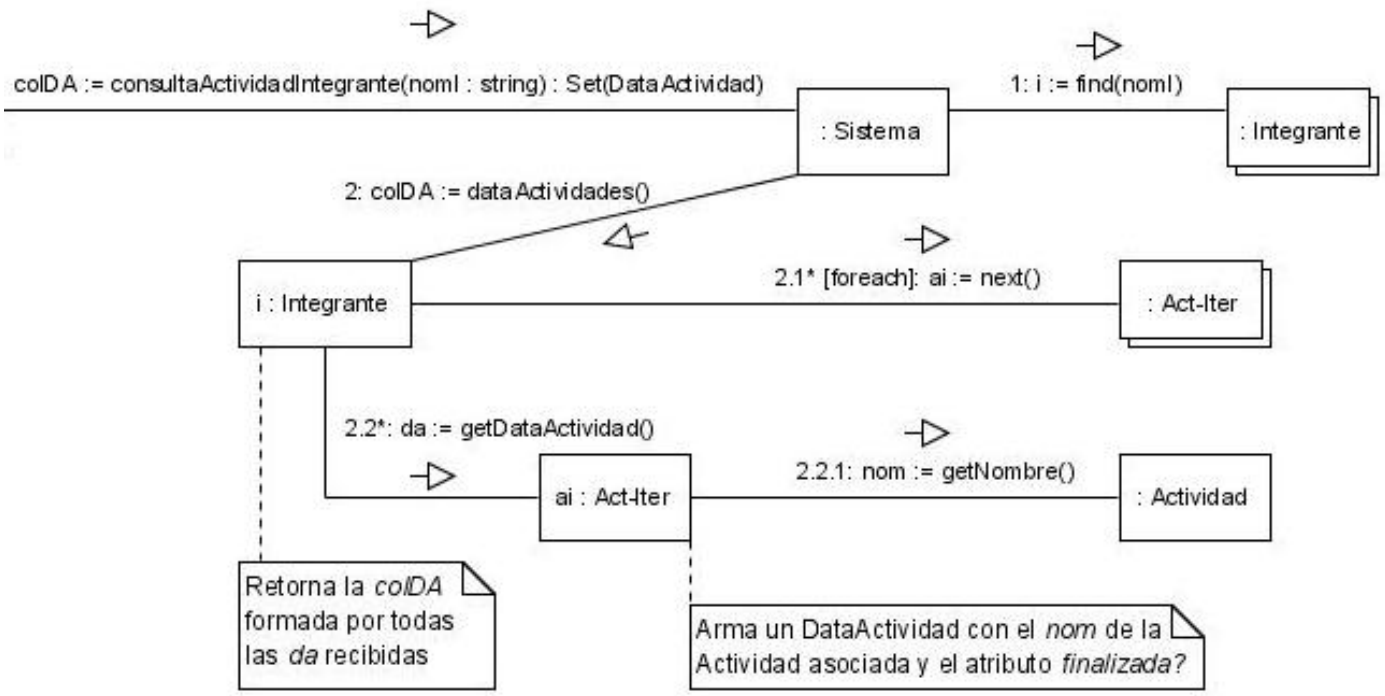
Operación	`altaActividadIteracion(nomE:string, nomA:string, nomI:string)`
Descripción	`Permite asignar una actividad específica a todas las iteraciones del proyecto así como definir el responsable de ésta (que será el mismo en todas las iteraciones)`
Pre- y poscondiciones	
`pre: Existe una instancia de Etapa con nombre igual a nomE.`	
`pre: Existe una instancia de Actividad asociada a la Etapa indicada anteriormente con nombre igual a nomA.`	
`pre: Existe una instancia de Integrante con nombre igual a nomI.`	
`pre: La Etapa de nombre nomE tiene un link con cada instancia de Iteración del sistema.`	
`pre: No existe ninguna instancia del tipo asociativo Act-Iter con links a las instancias de Actividad e Iteración identificadas anteriormente.`	
`post: Se creó una instancia de Act-Iter con finalizada?=FALSE para cada una de las instancias de Iteración, y se asoció a cada una de estas instancias a la correspondiente instancias de Iteración y a la instancia de Actividad identificada anteriormente.`	
`Post: Se asoció la instancia de Integrante identificada en las precondiciones a cada instancia de Act-Iter indicada en la postcondicion anterior.`	

Operación	consultarActividadIntegrante(nomI:string):Set(DataActividad)
Descripción	Permite consultar información de las actividades a las cuales está asignado un integrante del equipo del proyecto.
Pre- y poscondiciones	
pre: Existe una instancia de Integrante con nombre igual a nomI. post: Para cada instancia de Act-Iter con la cual esté linkeado la instancia de Integrante, se define un datavalue de tipo DataActividad. El datavalue contiene el nombre de la instancia de Actividad asociada a la instancia de Act-Iter y el valor del atributo finalizada? de esta última. Se retorna el conjunto de datavalues definidos.	

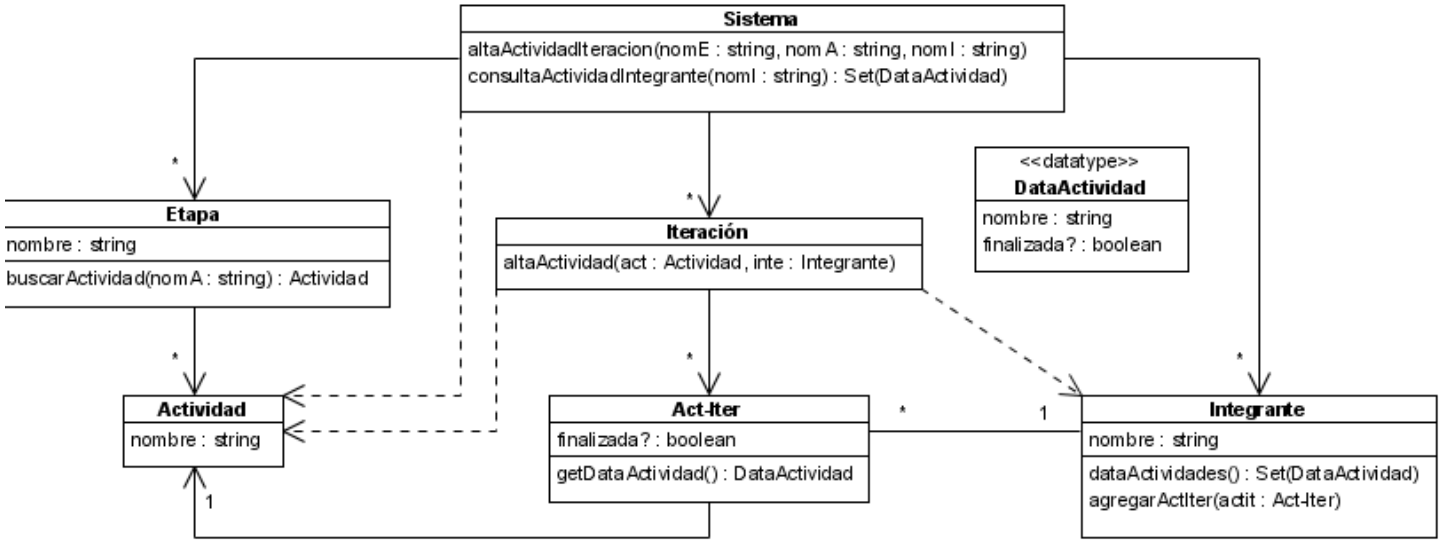
Se pide:

i. Realice los Diagramas de Comunicación para las operaciones especificadas en los contratos. No es necesario indicar las visibilidades.





ii. Realice el Diagrama de Clases de Diseño de la solución.



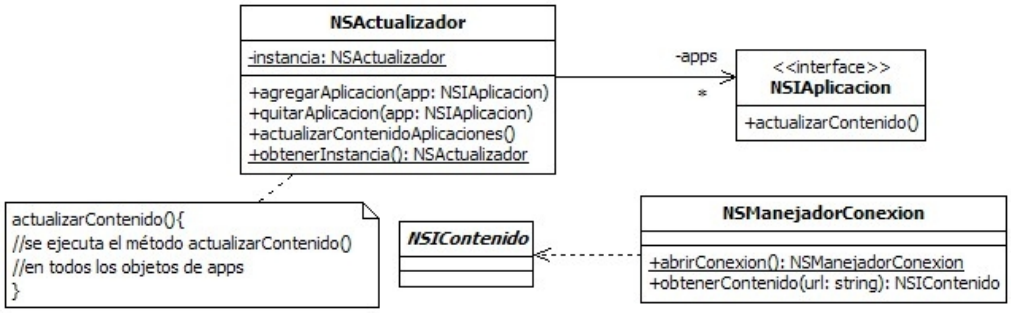
Problema 3 (40 puntos)

Se está realizando la construcción de un dispositivo móvil con capacidades de conexión inalámbrica a una red. El dispositivo tiene un sistema operativo que permite correr varias aplicaciones al mismo tiempo y provee de los servicios necesarios para que las mismas tengan conexión cuando estas lo soliciten. Por ultimo el sistema operativo puede detectar cuando el dispositivo pierde o recupera la conexión inalámbrica, por lo que puede regir que realizan las aplicaciones que corren en él ante estos eventos.

Como política general de funcionamiento, las aplicaciones tienen 2 modos de ejecución: uno en línea (***online***) y otro fuera de línea (***offline***); ocurre que cuando una aplicación está en línea solicita información a través de la conexión inalámbrica a demanda, cuando una aplicación esta ***offline*** solo se muestra el contenido obtenido previamente, y cuando pasa de estar ***offline*** a estar ***online*** el sistema exige que todas las aplicaciones refresquen su contenido.

Con el fin de soportar este comportamiento de manera automática, el sistema operativo instancia y utiliza una clase llamada `NSActualizador` que posee 3 métodos: uno para agregar una aplicación a actualizar cuando se recupera la conexión (se ejecuta cuando se inicia la aplicación), otro para quitar una aplicación del proceso de actualización (utilizado por ejemplo cuando la aplicación se cierra) y uno que se ejecuta cuando se recupera la conexión (el sistema operativo detecta que se entró a una zona WI-FI y posteriormente lo ejecuta). Para homogeneizar el tratamiento de todas las aplicaciones el sistema también define la interfase `NSIAplicacion` que posee un método llamado `actualizarContenido()` que se utiliza durante el proceso ejecutado por `NSActualizador` y también se define la clase abstracta `NSIContenido` que agrupa todos los contenidos que manejan las aplicaciones.

A su vez existe una clase en el sistema operativo llamada `NSManejadorConexion` que provee los métodos necesarios para obtener una conexión y recuperar contenidos a través de ella. En la siguiente figura se ilustran las clases antes mencionadas y sus dependencias.



Se pide:

i. Implementar la clase `NSActualizador` y la interfase `NSIAplicacion`. Considere que existe una implementación de `ICollection` e `IIterator` que proveen todas las funcionalidades necesarias para el manejo de colecciones de objetos. No incluir directivas del preprocesador.

```
/* NSIAPLICACION_H_ */

class NSIAplicacion: public ICollectible {
public:
      NSIAplicacion();
      virtual void actualizarContenido() = 0;
      virtual ~NSIAplicacion();
};
```

```
/* NSIAPLICACION_CC_ */

NSIAplicacion::NSIAplicacion() {}

NSIAplicacion::~NSIAplicacion() {}

/* NSACTUALIZADOR_H_ */

class NSActualizador {
private:
      static NSActualizador * instancia;
      ICollection *apps;
      NSActualizador();
public:
      static NSActualizador * obtenerInstancia();
      void agregarAplicacion(NSIAplicacion*);
      void quitarAplicacion(NSIAplicacion*);
      void actualizarContenidoAplicaciones();
      virtual ~NSActualizador();
};

/* NSACTUALIZADOR_CC_ */

NSActualizador* NSActualizador::instancia = 0;

void NSActualizador::actualizarContenidoAplicaciones() {
      IIterator *it = apps->getIterator();
      NSIAplicacion * app;
      while (it->hasCurrent()) {
            app = (NSIAplicacion*) it->current();
            app->actualizarContenido();
            it->next();
      }
      delete it;
}

void NSActualizador::quitarAplicacion(NSIAplicacion *nSIAplicacion) {
      apps->remove(nSIAplicacion);
}

void NSActualizador::agregarAplicacion(NSIAplicacion *nSIAplicacion) {
      apps->add(nSIAplicacion);
}

NSActualizador *NSActualizador::obtenerInstancia() {
      if (instancia == 0)
            instancia = new NSActualizador();
      return instancia;
}

NSActualizador::NSActualizador() {
      apps = new List();
}

NSActualizador::~NSActualizador() {
      NSActualizador::instancia = 0;
      IIterator *it = apps->getIterator();
      NSIAplicacion * app;
      while (it->hasCurrent()) {
            app = (NSIAplicacion*) it->current();
            delete app;
```

```
            it->next();
        }
        delete it;
        delete apps;
}
```

ii. ¿Se está utilizando algún patrón de diseño en la solución? En caso de que la respuesta sea afirmativa identifique el patrón justificando su respuesta.

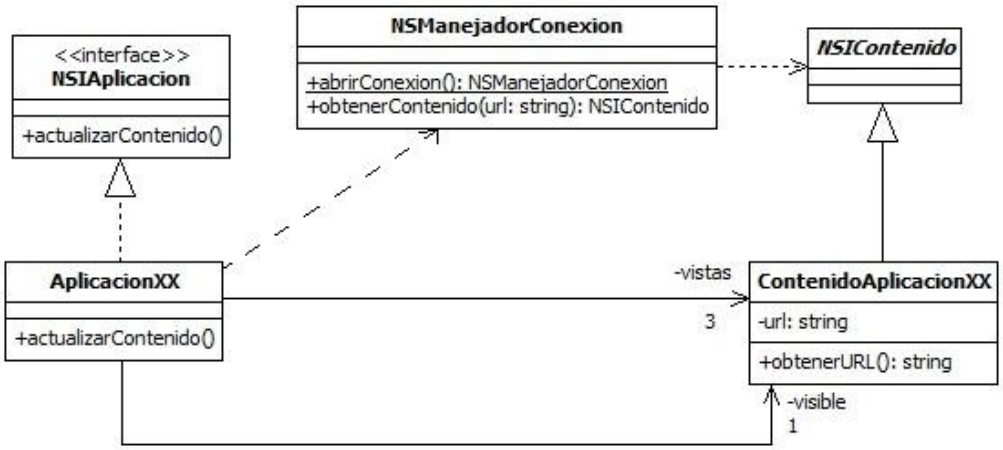
Se está utilizando el patrón ***Singleton*** en la clase NSActualizador.
Se está utilizando el patrón ***Observer*** con los siguientes roles:

- ***Subject*** -> `NSActualizador`
- ***Observer*** -> `NSIAplicacion`
- ***ConcreteObserver*** -> Las aplicaciones que se implementarán para el dispositivo (p.e.: `AplicacionXX`)

En una segunda etapa del desarrollo se quiere permitir que las aplicaciones que corren en el dispositivo puedan mantener varias vistas simultáneas de los contenidos que se pueden acceder. Dado que la cantidad de información que se puede mostrar está limitada por el tamaño de la pantalla, se espera que las vistas se manejen de manera similar a como lo hacen los navegadores Web (por ejemplo mediante el uso de pestañas donde se muestra solo un contenido a la vez). Pensando en el ahorro de batería del dispositivo (uso innecesario de la antena para la conexión) se define la siguiente política: ante la actualización del contenido, si la aplicación tiene varias vistas, solo se refresca la información del contenido visible. El diseño de todas las aplicaciones que corran en el dispositivo debe tomar en cuenta esta política y las definidas en la parte anterior, sino serán rechazadas por el fabricante del dispositivo.

Se pide:

iii. Realizar el DCD y la implementación de la clase `AplicacionXX` que representa una aplicación que maneja 3 vistas simultáneas de contenido representado por la clase `ContenidoAplicacionXX` que almacena la información del contenido en un `string`. Considere que existe una implementación de `ICollection` e `IIterator` que proveen todas las funcionalidades necesarias para el manejo de colecciones de objetos. No incluir directivas del preprocesador.




```
/* CONTENIDOAPLICACIONXX_H_ */

class ContenidoAplicacionXX: public NSIContenido {
private:
      string url;
      string contenido;
public:
      ContenidoAplicacionXX(string, string);
      string obtenerURL();
      virtual ~ContenidoAplicacionXX();
};

/* CONTENIDOAPLICACIONXX_CC_ */

ContenidoAplicacionXX::ContenidoAplicacionXX(string     url,     string
contenido) {
      this->url = url;
      this->contenido = contenido;
}

string ContenidoAplicacionXX::obtenerURL() {
      return url;
}

ContenidoAplicacionXX::~ContenidoAplicacionXX() {
      delete url;
}

/* APLICACIONXX_H_ */

class AplicacionXX: public NSIAplicacion {
private:
      ContenidoAplicacionXX *visible;
      int idxActual;
      ContenidoAplicacionXX *vistas[3];
public:
      AplicacionXX();
      virtual void actualizarContenido();
      virtual ~AplicacionXX();
};

/* APLICACIONNXX_CC_ */
```

```
AplicacionXX::AplicacionXX() {
      idxActual = 0;
      NSActualizador::obtenerInstancia()->agregarAplicacion(this);
}

void AplicacionXX::actualizarContenido() {
      NSManejadorConexion *cnx = NSManejadorConexion::abrirConexion();
      ContenidoAplicacionXX *nuevo =
                  (ContenidoAplicacionXX*) cnx->obtenerContenido(
                              visible->obtenerURL());
      delete visible;
      visible = nuevo;
      vistas[idxActual] = nuevo;
}

AplicacionXX::~AplicacionXX() {
      NSActualizador::obtenerInstancia()->quitarAplicacion(this);
      for (int i = 0; i < 3; i++)
            delete vistas[i];
}
```