Programación Orientada a Objetos I
Parcial - 2da. Fecha

Cada Miembro es identificado con un e-mail. Cada Miembro puede tener “intereses profesionales”, “novedades recientes” y además “contactos directos”. El e-mail es un string. Los “intereses profesionales” se implementan como una colección con símbolos Smalltalk con los temas que son de interés profesional del usuario. Las “novedades recientes” son noticias que se relacionan con cada usuario. Los “contactos directos” se implementan como una colección de Miembros. Las conexiones entre miembros son bidireccionales, miembros conectados se conocen mutuamente.

Caso 1: Existen cuatro Miembros: Natalia, Diego, Lucas y Xavier. Natalia y Diego son contactos directos. Diego y Lucas son contactos directos. Xavier es contacto directo de Diego y Lucas.
Natalia tiene interés en: #programacion y #educacion. Diego tiene interés en: #formalismos. Lucas tiene interés en: #educacion y #testing. Natalia y Lucas son mutuamente “contacto indirecto”. Natalia y Xavier son “contactos indirectos”.

Las Novedades asociadas a cada Miembro pueden ser de varios tipos: Personal, y Profesional. Una Novedad Personal es comunicable con los contactos directos. Una Novedades Profesional es comunicable a todos los miembros que están conectados de manera directa e indirecta y que tienen intereses en común.

Caso 2: Natalia, Diego, Xavier y Lucas han ingresado datos de sus vacaciones (novedades personales). Natalia además ha ingresado datos de un ascenso a “Arquitecto” (novedad profesional). Lucas ha ingresado datos de una conferencia que está organizando (novedad profesional).
La red social CONECTADOS, presenta a cada uno de sus usuarios una Pizarra de Novedades, con las novedades Personales (de sus contactos directos) y las profesionales (de contactos directos e indirectos que tienen algún interés en común). La PizarraNovedades entiende un mensaje de clase:
`#novedadesPara: unUsuario`
“toma un usuario como parámetro y retorna una nueva instancia de PizarraNovedades”

Las instancias de PizarraNovedades entienden los siguientes mensajes:
`#agregarContacto: unMiembro`
“Crea el contacto directo bidireccional entre unMiembro y el usuario de esta pizarra”
`#contactosDirectos`
“retorna la lista ordenada alfabéticamente de contactos directos del usuario de esta pizarra”
`#novedadesPersonales`
“retorna la lista de novedades personales de cada contacto directo”
`#novedadesProfesionales`
“retorna la lista de novedades profesionales de contactos directos o indirectos con igual interés”

Caso 3: El sistema informático de CONECTADOS construye una instancia de PizarraNovedades para Natalia. La misma tiene dos novedades. Primero, la novedad personal de Diego sobre sus vacaciones, dado que es contacto directo. Segundo, la NovedadProfesional de Lucas sobre la conferencia que está organizando, dado que es

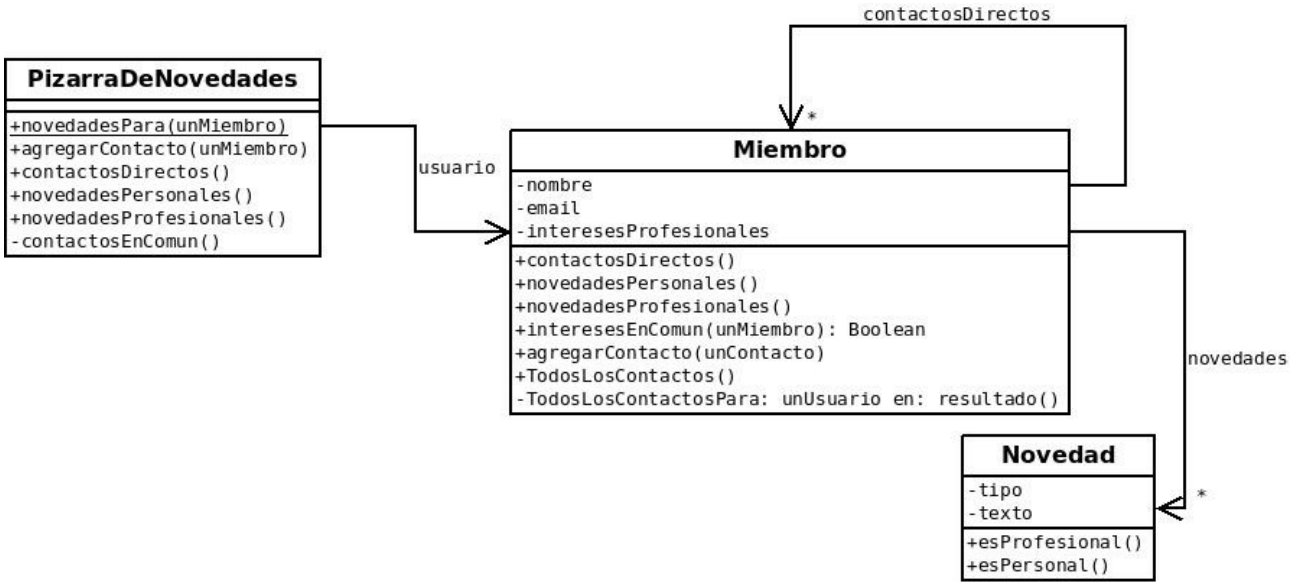
contacto indirecto pero tienen intereses en común (a ambos les interesa #educacion)

1. Documente utilizando Diagrama de Clases UML.
2. Presente el Diagrama de Secuencia UML del envío del mensaje del constructor de PizarraNovedades
3. Considerando los Casos 1, 2 y 3 cuál sería el resultado del siguiente código Smalltalk?

```
(PizarraNovedades novedadesPara: usuarioXavier)
novedadesProfesionales
```

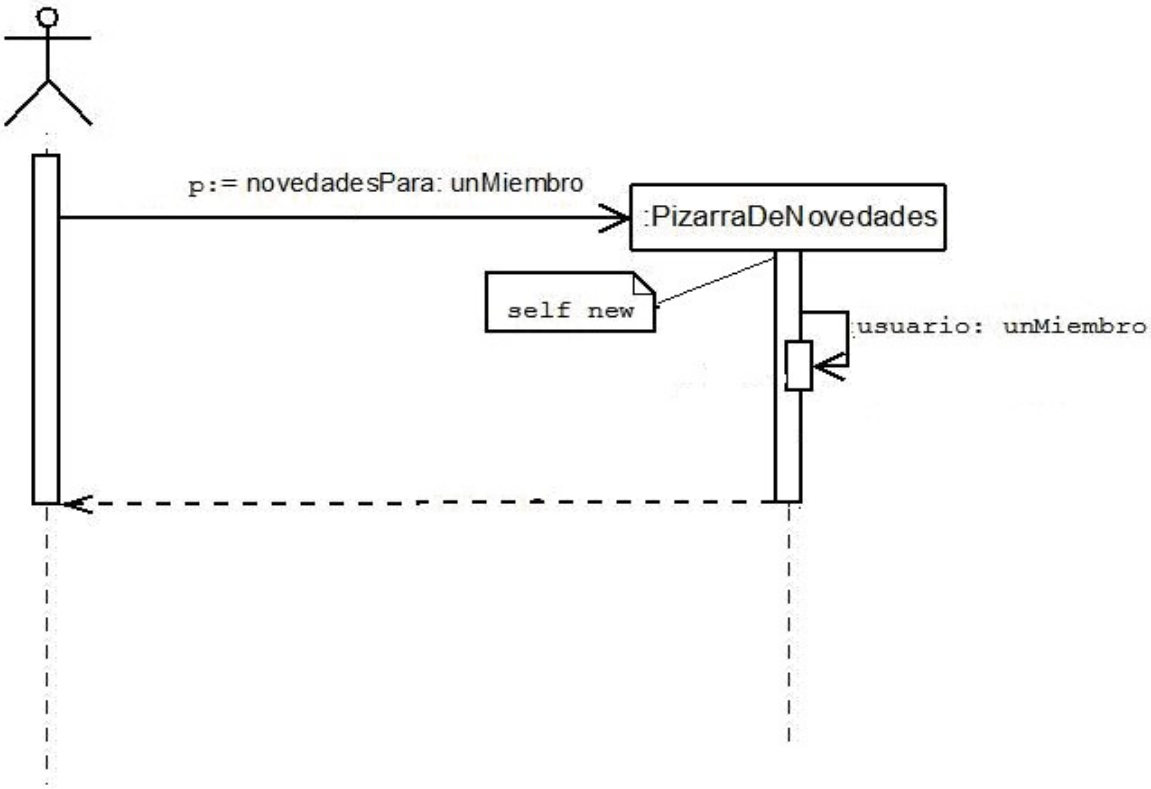
4. Implemente en Smalltalk los métodos de PizarraNovedades y los métodos que estos invocan

Ejercicio 1



Obs: si bien se podría plantear una solución utilizando una jerarquía de Novedad, a los efectos del enunciado, es suficiente la solución planteada.

Ejercicio 2



Ejercicio 3
Devuelve una colección vacía

Ejercicio 4

Pizarra

```
Pizarra class#novedadesPara: unUsuario
```

"toma un usuario como parámetro y retorna una nueva instancia de PizarraNovedades"

```
^ (self new)
     usuario: unUsuario ;
     yourself
```

```
#agregarContacto: unMiembro
```

"Crea el contacto directo bidireccional entre unMiembro y el usuario de esta pizarra"

```
     self usuario agregarContacto: unMiembro.
```

```
#contactosDirectos
```

"retorna la lista ordenada alfabéticamente de contactos directos del usuario de esta pizarra"

```
  ^ self usuario contactosDirectos
```

```
#novedadesPersonales
```
"retorna la lista de novedades personales de cada contacto directo"
```
 |novedades|
   novedades:= OrderedCollection new.
  ^ self usuario contactosDirectos do:[:c| novedades addAll:
                                        (c novedadesPersonales)].
     ^novedades

#novedadesProfesionales
```
"retorna la lista de novedades profesionales de contactos directos o indirectos con igual interés"
```
 | novedades |
     novedades := OrderedCollection new.
     self contactosEnComun do: [ :c | novedades addAll:
                                          c novedadesProfesionales].
     ^ novedades

#contactosEnComun
```
"Retorna una colección con los contactos directos e indirectos con intereses comunes a los del usuario "
```
     ^ (self usuario todosLosContactos)
         select:[:c| c interesesEnComun: usuario]
```

Miembro

```
#initialize: unNombre mail: unMail
     nombre := unNombre.
     mail := unMail.
     intereses:= OrderedCollection new.
     novedades := OrderedCollection new.
     contactosDirectos := SortedCollection sortBlock: [:cont1
                              :cont2| cont1 nombre < cont2 nombre].
     ^self

#agregarContacto: unMiembro.
     self contactosDirectos add: unMiembro.
     unMiembro contactosDirectos add: self.

#contactosDirectos
     ^ contactosDirectos

#novedadesPersonales
   ^ self novedades select:[:unaNovedad| unaNovedad esPersonal]

#novedadesProfesionales
   ^ self novedades select:[:unaNovedad| unaNovedad esProfesional]


#interesesEnComun: otroUsuario
```

```
"Retorna si el usuario tiene intereses en común con otroUsuario"
     ^ self intereses anySatisfy:
          [:interes| otroUsuario intereses includes: interes]

#todosLosContactos
"Retorna una colección con todos los contactos (directos e
indirectos) del usuario"
|contactos|
 contactos := OrderedCollection new.
 self contactosDirectos do:[ :c|
      c todosLosContactosPara: self en: contactos.].
 ^  contactos


#todosLosContactosPara: unUsuario en: resultado
"Recorre la red de contactos guardando en resultado, todos los
contactos directos e indirectos de unUsuario"

 (unUsuario ~= self) &  ((resultado includes: self) not)
     ifTrue: [
         resultado add:self.
         self contactosDirectos do:[ :cd |
             cd todosLosContactosPara: unUsuario en:resultado.]]
```

Novedad

```
#esPersonal
 ^ tipo == #personal

#esProfesional
 ^ tipo == #profesional
```