

## PDOO. Examen Febrero 13/14. Primer parcial Resuelto

**1. (4 puntos)** Indica si las siguientes afirmaciones son verdaderas o falsas. Dos preguntas incorrectas anulan una correcta.

Dos objetos con el mismo estado pueden tener distinta identidad.	V
Si hay mil objetos de una clase X habrá mil copias de su variable de clase x1.	F
El código: <b>MuertoViviente vampiro;</b> crea en Java un objeto de la clase MuertoViviente.	F
El código: <b>attr_writer :color</b> crea el get y el set del atributo color en Ruby.	F
En Ruby los métodos de instancia son públicos y los atributos de instancia son privados, por defecto.	V
La ligadura estática de un mensaje a un método es menos flexible que la ligadura dinámica, pero más eficiente.	V
La identidad de un objeto en programación orientada a objetos nos la da su dirección de memoria.	V
En el código: <b>a.m(b)</b> , a es el objeto emisor del mensaje.	F
Un mensaje es la implementación de una responsabilidad de una determinada clase de objetos.	F
En Java y Ruby, gracias a la reflexión, es posible modificar las clases en tiempo de ejecución.	F

Teniendo en cuenta el diagrama de clases:

Desde la clase AyudaSolicitada se puede acceder a todos los elementos públicos del paquete GestorProgramas.	V
Un voluntario puede participar en cualquier acción de un programa sin ningún tipo de restricción.	F
El estado de un objeto de la clase Auditor viene determinado por el estado de un objeto de la clase ONG.	F
Un voluntario podría participar en acciones de distintos programas.	V
Un voluntario puede pertenecer a varias ONG.	F
Cuando se define un objeto de la clase Acción, éste tiene que asociarse a un determinado objeto de la clase Programa.	V
En una acción puede participar más de un voluntario como especialista.	V
Desde un objeto de la clase ONG se puede llegar a conocer a todos los especialistas de una determinada acción en un programa.	V
El estado de un objeto Voluntario está exclusivamente determinado por su dni, nombre y especialidad.	F
Todos los métodos de la clase Acción pueden ser accedidos desde la clases AyudaConcedida.	F

Teniendo en cuenta el diagrama de comunicación:

En el envío de mensaje 1.2 el objeto receptor es self/this.	V
En envío de mensaje 1.3.1 significa que a todas las acciones del programa le vamos a adscribir un voluntario.	F
En el método crear de la clase Participa (1.3.3.1) se construye un enlace entre el objeto Participa y el objeto Voluntario.	V
El enlace entre el objeto Acción y el multiobjeto de la clase Participa estereotipado como <<A>> significa que el objeto Acción conoce al multiobjeto sólo para esta operación.	F
El multiobjeto misParticipaciones enlazado con voluntario es un subconjunto del multiobjeto participantes enlazado con accion.	F
El envío de mensaje 1.3.3 se lleva a cabo sólo si adscrito es verdadero.	F

## PDOO. Examen Febrero 13/14. Primer parcial Resuelto

2. (1 punto) Usando la siguiente nomenclatura:

AS = Asociación.

CO = Composición.

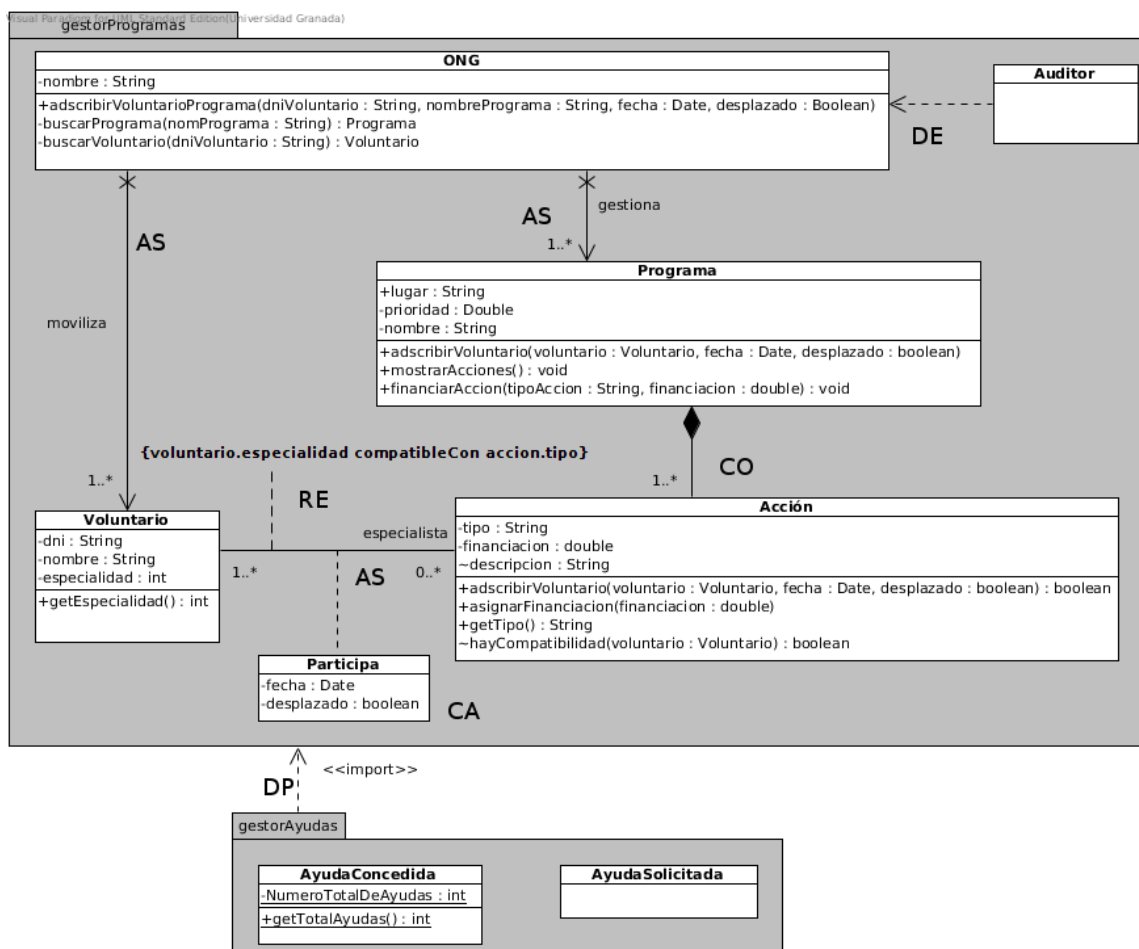
AG = Agregación.

DE = Dependencia.

CA = Clase Asociación.

RE = Restricción.

Etiqueta los elementos correspondientes en el propio diagrama de clases.



## PDOO. Examen Febrero 13/14. Primer parcial Resuelto

3. (1.5 puntos) Implementa en Java y Ruby las clases Accion y AyudaConcedida.

### Java: Accion

```
package gestorProgramas;

import java.util.ArrayList;
import java.util.Date;

public class Accion {

    public String tipo;
    private double financiacion;
    String descripcion;
    private ArrayList<Participa> especialistas = new ArrayList();
    private Programa miPrograma;
    public boolean adscribirVoluntario(Voluntario voluntario, Date fecha, boolean desplazado){return false;}
    public void asignarFinanciacion(double financiacion){}
    public String getTipo(){return tipo;}
    boolean hayCompatibilidad(Voluntario voluntario){return false;}
}
```

### Java: AyudaConcedida

```
package gestorAyudas;
import gestorProgramas.*;
public class AyudasConcedidas {
    private static int NumeroTotalDeAyudas;
    public static int getTotalAyudas(){return NumeroTotalDeAyudas;}
}
```

### Ruby : Accion

# definida en el archivo gestorProgramas

```
class Accion
  @tipo
  @financiacion
  @descripcion
  @especialistas =Array.new
  @miPrograma
  attr_reader :tipo

  def adscribirVoluntario(voluntario,fecha,desplazado)
  end
  def asignarFinanciacion(financiacion)
  end
  def hayCompatibilidad(voluntario)
  end
end
```

## PDOO. Examen Febrero 13/14. Primer parcial Resuelto

### Ruby: AyudaConcedida

```
require_relative 'gestorProgramas'
class AyudaConcedida
  @@NumeroTotalDeAyudas
  def self.getTotalAyudas

    end
end
```

**4. (1.5 puntos)** Implementa en Java y en Ruby el método `adscribirVoluntario(...)` de la clase Programa.

JAVA:

```
class Programa {
    private ArrayList<Accion> acciones = new ArrayList();

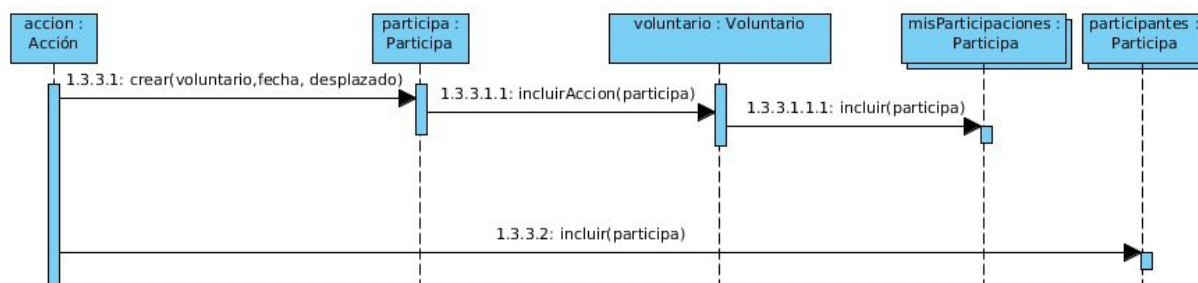
    public void adscribirVoluntario(Voluntario voluntario, Date fecha, boolean despalzado)
    throws Exception
    {
        boolean adscrito=true;
        for (Accion accion:acciones){
            if(accion.hayCompatibilidad(voluntario))
                adscrito = accion.adscribirVoluntario(voluntario, fecha, despalzado);
        }
        if(!adscrito)
            throw new Exception("el voluntario no ha podido adscribirse a ese programa");
    }
}
```

RUBY:

```
class Programa
  @acciones=Array.new()
  def adscribirVoluntario(voluntario,fecha,desplazado)
    acciones.each do |accion|
      if accion.hayCompatibilidad
        adscrito = accion.adscribirVoluntario(voluntario, fecha, desplazado)
      end
    end
    if !adscrito
      raise 'el voluntario no se ha podido adscribir a ese programa'
    end
  end
end
```

## PDOO. Examen Febrero 13/14. Primer parcial Resuelto

5. (2 puntos) Obtén el Diagrama de secuencia de la operación `adscribirVoluntario(...)` de la clase `Acción`, incluyendo todos los envíos de mensaje subordinados al 1.3.3.



Reservados todos los derechos. No se permite la explotación económica ni la transformación de esta obra. Queda permitida la impresión en su totalidad.