# Ejercicio 1

***Ruby Esencial***

**Inicio: 23 de Enero**
**Entrega: 3 de Febrero a las 11:00 a.m.**

El objetivo de este ejercicio es familiarizarse con el lenguaje Ruby y sus principales construcciones:

- Clases, herencia, control de acceso
- Contenedores, iteradores
- Ficheros

Se pide crear una aplicación de mensajería instantánea para la comunicación entre grupos de usuarios. Los requisitos de la aplicación se presentan en el enunciado de manera incremental, de tal manera que en cada apartado debes hacer un incremento respecto al apartado anterior.

**Nota**: Siempre que sea posible, se favorecerá el uso de iteradores para operar con arrays.

---

## Paso 1: Grupos de usuarios (3 puntos)

La aplicación debe gestionar usuarios y grupos de usuarios. Tanto los usuarios como los grupos tienen un nombre. Un grupo puede contener usuarios y otros grupos, pero no a sí mismo. Además, los grupos tienen un número máximo de miembros que debe quedar fijado en el momento de crear el grupo y no podrá ser modificado posteriormente. El programa no debe permitir que existan grupos con más miembros de los permitidos. De esta manera, si tenemos un grupo de capacidad máxima 2 que contiene 1 usuario, no es posible añadirle otro grupo que contenga 2 usuarios. Un usuario puede pertenecer a varios grupos, pero un grupo no puede contener usuarios ni grupos repetidos. Se considera repetido que un usuario (o grupo) pertenezca dos veces al mismo grupo de manera directa o indirecta a través de algún subgrupo.

Se recomienda crear cada clase (o conjunto de clases relacionadas) en ficheros separados. Si en el fichero-a.rb necesitas usar una clase que está definida en el fichero-b.rb, incluye en el fichero-a.rb la siguiente declaración (donde no necesitas incluir la extensión rb al indicar el nombre del fichero-b):

```
require 'fichero-b'
```

Para probar la gestión de grupos, define un script de Ruby donde se creen algunos usuarios y grupos, y se incorporen miembros a algún grupo existente. El script se entregará junto con las demás clases del ejercicio. No hace falta que los datos de usuarios y grupos (ej. el nombre) se pidan por consola.

---

## Paso 2: Mensajes (2 puntos)

La aplicación debe gestionar mensajes. Un mensaje tiene un *identificador único* que el sistema asigna automáticamente, un tema, una fecha de envío, y el usuario que lo envió. Hay distintos tipos de mensajes:

(a) *noticias*: contienen el texto de la noticia. Las noticias pueden ser (o no) respuesta a otra noticia, y por tanto deben guardar una referencia a la noticia que responden. A partir de una noticia A, se quiere poder acceder a su "hilo", es decir, a la secuencia de todas las noticias a las que la noticia A responde, de manera directa o indirecta.
(b) *eventos*: contienen la fecha del evento.
(c) *reuniones*: contienen la fecha de la reunión y el lugar donde se celebrará.

Para las fechas, puedes usar el tipo Date de Ruby. A continuación tienes un ejemplo de uso. Para más información, consulta la documentación en http://ruby-doc.org/stdlib-2.0.0/libdoc/date/rdoc/Date.html

```
require 'date'

d1 = Date.new(2014,2,3)
puts d1.day    # 3
puts d1.month  # 2
puts d1.year   # 2014
puts Date.new(2014,1,1) <=> Date.new(2014,1,2) # -1 (la primera fecha es menor)
puts Date.new(2014,1,1) <=> Date.new(2014,1,1) # 0  (las dos fechas son iguales)
puts Date.new(2014,1,1) <=> Date.new(2013,1,1) # 1  (la primera fecha es mayor)
```

Actualiza el script realizado en el paso 1 para mostrar la creación de mensajes de cada tipo, y el hilo de una noticia.

---

## Paso 3: Envío y recepción de mensajes (2,5 puntos)

El envío de mensajes se realiza a través de distribuidores de mensajes. Tanto los usuarios como los grupos de usuarios pueden suscribirse a un distribuidor. Cuando un distribuidor recibe un mensaje, lo almacena y además lo distribuye a todos los usuarios y grupos que estén registrados. Un distribuidor de mensajes tiene un nombre, un tema y almacena mensajes hasta un máximo determinado, que se indica al crear el distribuidor. El distribuidor tendrá un método para eliminar los mensajes anteriores a una fecha dada, y debe poder devolver los eventos y reuniones almacenados que estén programados entre dos fechas dadas.

Cuando un grupo recibe un mensaje de un distribuidor, debe retransmitir el mensaje a todos sus miembros. Cuando un usuario recibe un mensaje, simplemente lo imprime por pantalla precedido de su nombre. La aplicación debe prohibir que un usuario o grupo esté suscrito dos veces al mismo distribuidor. No se considera repetido que un usuario y los grupos a los que pertenece estén suscritos al mismo distribuidor.

Añade la posibilidad de configurar una respuesta automática para un usuario. Si un usuario tiene activada la respuesta automática, responderá a todos las noticias que reciba con una noticia con texto predefinido. Para evitar bucles infinitos, la respuesta automática no se efectuará para noticias remitidas por el mismo usuario, o que sean respuesta a noticias enviadas por el mismo usuario.

Actualiza el script realizado en los pasos anteriores para incluir la creación de un distribuidor, el registro de usuarios en el mismo, y la distribución de mensajes a través del distribuidor.

---

## Paso 4: Otros escuchantes (2,5 puntos)

Se desea poder registrar en los distribuidores no sólo usuarios y grupos, sino también los siguientes programas para el procesamiento automático de mensajes:

(a) *loggers*: cuando reciben un mensaje del distribuidor, lo guardan en un fichero con formato HTML. Un *logger* se configura con el nombre del directorio donde se guardarán los ficheros. Si el directorio no existe (`File.exists? nombre-directorio`) deberá crearse (`Dir.mkdir nombre-directorio`). A continuación tienes la estructura de una página que muestra la información de una noticia (fichero msg1.html). Para eventos y reuniones habría que mostrar además la fecha y lugar.

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN">
<html>
  <head>
    <meta http-equiv="content-type" content="text/html; charset=ISO-8859-1">
  </head>
  <body>
  <h1>Noticia #1: Tema del mensaje</h1>
  <h2>De: Nombre del remitente</h2>
  <h3>Respuesta al mensaje <a href="msg0.html">id del mensaje anterior</a></h3>
  Texto del mensaje msg1
  </body>
</html>
```

(b) *analizadores de mensajes*: llevan la cuenta de cuántos mensajes ha mandado un usuario concreto, o bien un determinado grupo. Además, imprime en pantalla esta cantidad cada vez que recibe X mensajes para el mismo usuario o grupo (donde el valor de X se configura en el constructor).
(c) *otros distribuidores de* mensajes: retransmiten el mensaje a los "escuchantes" registrados en ellos.

Actualiza el script realizado en los pasos anteriores para mostrar la nueva funcionalidad: crea dos distribuidores, uno suscrito al otro; crea un *logger* suscrito a cada uno de los distribuidores; crea un analizador de mensajes suscrito a cada uno de los distribuidores; añade código para que se envíen unos cuantos mensajes.

## Normas de Entrega

El código se comprimirá en un único fichero zip/rar, que se entregará a través de moodle antes de la fecha de entrega indicada en la cabecera del enunciado.

El nombre del fichero deberá incluir el nombre y apellido del estudiante.