Laboratorio S3:
Clases e Interacción entre Objetos (parte 1)

Objetivos:

- Entender que la definición de una clase puede definir un nuevo tipo de dato
- Entender los conceptos de abstracción y modularización
- Ser capaz de instanciar o crear múltiples objetos en Java: el operador new
  - Ser capaz de definir variables de tipo referencia
  - Ser capaz de construir objetos utilizando distintos constructores
  - Ser capaz de asignar objetos a variables
  - Entender que la asignación de objetos es copiar referencias y no objetos
- Entender que cada objeto es independiente y cada uno tiene su propio estado
- Entender que los valores de las variables de tipo clase son referencias
- Invocar métodos de un objeto
  - Ser capaz de invocar métodos de un objeto concreto
  - Ser capaz de pasar parámetros a los métodos
- Uso del debugger de Eclipse
  - Ser capaz de analizar el valor de las variables y los estados de los objetos de un programa

Herramientas que vamos a utilizar:

- Entorno de desarrollo Eclipse
- Depurador (*debugger*) de Eclipse

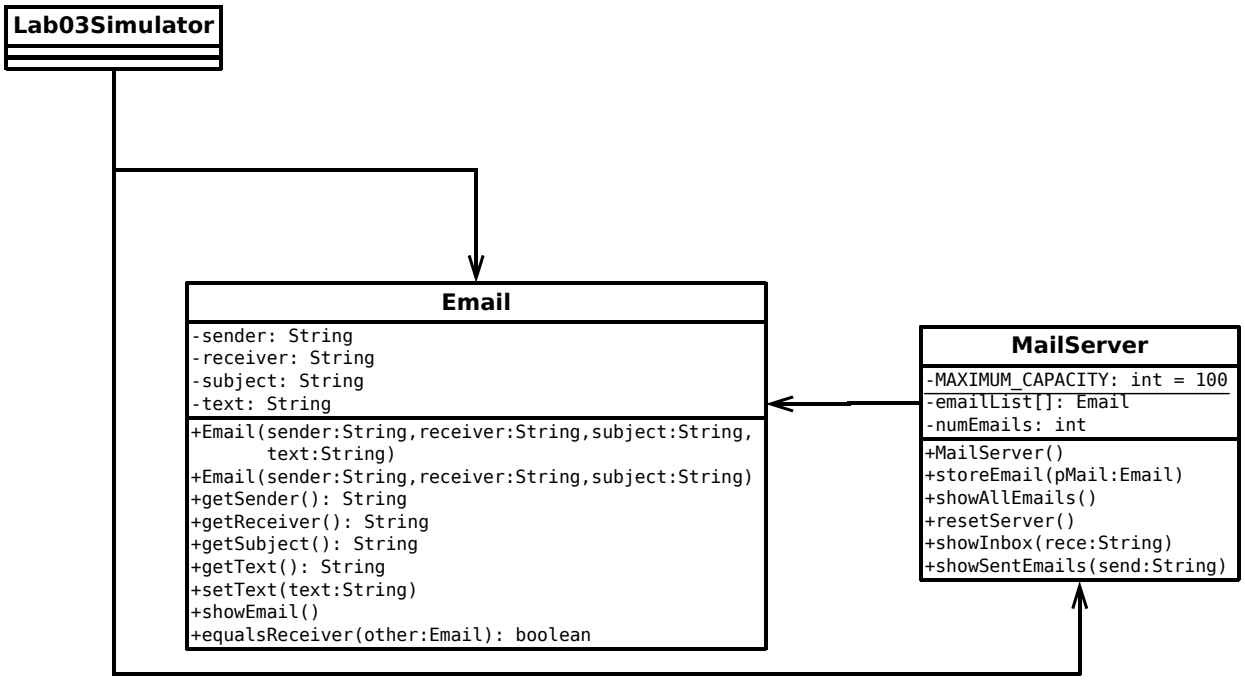
Entregables

Se debe entregar el proyecto de Eclipse exportado.

- Nombre fichero: apellido_nombre.zip
- La entrega es individual y se subirá a eGela
- Fecha límite de entrega: viernes 12 de febrero a las 23:55

Contexto:

El proyecto a desarrollar simulará un sistema de correo electrónico sencillo. Los usuarios pueden enviarse (unos a otros) mensajes de correo. El sistema de envío de correos electrónicos entre usuarios se gestiona mediante un servidor de correo. *Obsérvese que se trata de una simple simulación, esto es, los mensajes se intercambian entre objetos de este proyecto, no son mensajes que van por la red.*



Tareas a realizar en el laboratorio

1. Crea un nuevo proyecto Java en Eclipse, llámalo S3*Mail.* A continuación crea un paquete denominado *mailsystem*.
2. Crea una clase nueva, denominada *Email*, dentro del paquete *mailsystem*. Esta clase encapsula el remitente, destinatario, asunto y contenido de un mensaje. Para ello, un email tendrá los siguientes atributos: `sender (String), receiver (String), subject (String), text (String)`. Implementa las siguientes constructoras y métodos de la clase:
   a. **Constructoras:** Intentar reutilizar el código cuando se pueda
      - Recibe un parámetro por cada atributo de la clase `(sender, receiver, subject` y `text)`
      - Recibe parámetros para los atributos `sender`, `receiver` y `subject.` El texto será un String vacío ("")
   b. Un ***getter*** para cada atributo privado que se ha definido en la clase
   c. El ***setter*** del atributo `text`
   d. El método **showEmail**: Muestra por pantalla el contenido de un Email, mostrando por pantalla el valor que tienen todos los atributos del objeto. Ejemplo:

```
Sender: maite.urretavizcaya@ehu.eus
Receiver:julian.gutierrez@ehu.eus
Subject:lab S3 y S4
Text: Ya está en eGela el enunciado del laboratorio de la semana 3
```

3. Descárgate los ficheros *MailServer.java* y *LabS03Simulator.java* y cópialos en la carpeta *src/mailsystem* del proyecto que acabas de crear.

4. Para analizar el estado de los objetos utilizaremos el Debugger. Para ello, abre el fichero *LabS03Simulator.java* en el editor de código. Pon un punto de interrupción (*Toggle Breakpoint*) en la primera línea de código del método **testEmail**. En vez de ejecutar el programa, depúralo paso a paso empleando el debugger (*Run→ Debug As*) para analizar el estado de los objetos en cada instrucción del método de prueba. Recuerda que sobre el uso del Debugger en Eclipse tienes un documento en eGela.
5. Modifica la clase *MailServer* que has descargado programando el cuerpo de los métodos:
   a. La **constructora** de la clase, de tal forma que inicialice la lista de emails del servidor.
   b. **storeEmail**: Método que si es posible añada el *Email* pasado como parámetro al final de la lista de los ya existentes en la lista de emails del servidor.
   c. **showAllEmails**: Método que muestra por pantalla todos los emails en la lista de emails del servidor.
6. Añade a la clase *LabS03Simulator* un nuevo método testMailServer que realizará las tareas que se indican a continuación. Cuando lo termines, haz una llamada al mismo desde el main de la clase.
   a. Crea una instancia de la clase *MailServer* (*insServer*)
   b. Crea tres emails desde *user1* a user2 y envía otro email desde *user2* a *user1*. Guárdalos en la lista de emails de *MailServer.*
   c. Muestra por pantalla todos los emails que hay en el servidor

Tareas complementarias

7. Crea en la clase Email el método **`equalsReceiver`**: Recibe como parámetro un objeto de tipo Email y devuelve si el objeto actual y el pasado por parámetro tienen el mismo receptor o no.
8. Para probarlo, incluye en el método **testEmail** de la clase *Lab03Simulator* dos llamadas, la primera que compruebe si dos de los mensajes de *user1* a user2 tienen el mismo receiver y otra que compruebe si un mensaje de *user1* a *user2* tiene el mismo receiver que el de *user2* a *user1*.
9. En la clase *MailServer*, crea los siguientes métodos con sus cabeceras y su cuerpo:
    a. **resetServer**: Vacía la lista de emails del servidor
    b. **showInbox**: Dado el nombre de un usuario del sistema, muestra por pantalla todos los emails guardados en el servidor que se hayan enviado a dicho usuario.
    c. **showSentEmails**: Dado el nombre de un usuario del sistema, muestra por pantalla todos los emails guardados en el servidor que haya enviado dicho usuario.
10. Escribe las instrucciones que se indican a continuación dentro el método **testMailserver** de la clase *Lab03Simulator* y prueba el mismo :
    a. Muestra los mensajes enviados por *user1* (utilizando el método de la clase *MailServer* que permita hacerlo)
    b. Muestra los mensajes recibidos por *user1* (utilizando el método de la clase *MailServer* que permita hacerlo)