Laboratorio S4
Clases e Interacción entre Objetos (parte 2)

Objetivos:
- Todos los objetivos del laboratorio S3 y además:
- Invocar métodos de un objeto
  - Ser capaz de diferenciar entre llamadas internas y externas a métodos
- Entender las características y uso de la estructura de datos ArrayList
- Documentación de clases y métodos en Java con Javadoc

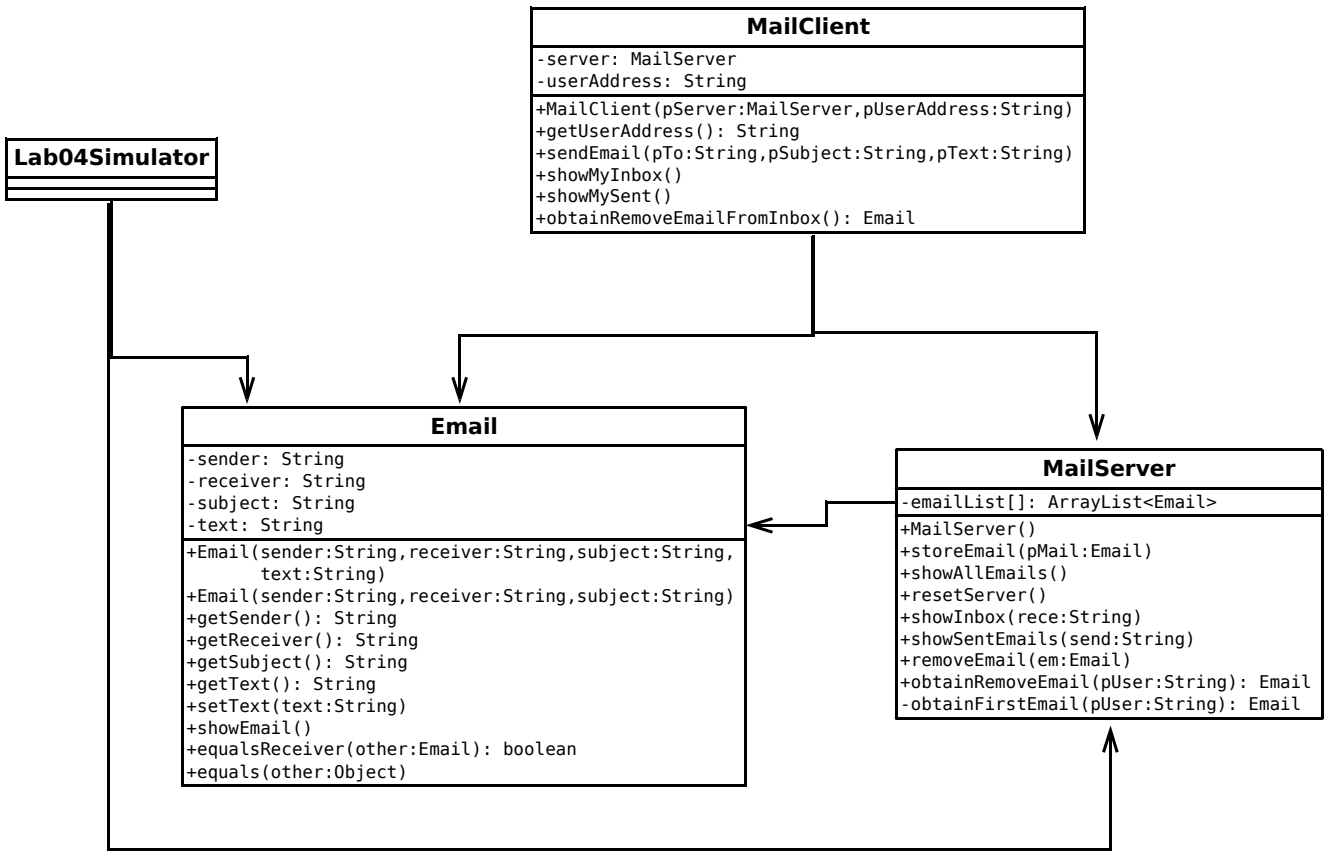
Herramientas que vamos a utilizar:
- Entorno de desarrollo Eclipse
- Depurador (*debugger*) de Eclipse
- JavaDoc

Entregable

Se debe entregar un fichero **ZIP** con el **Proyecto Exportado** del laboratorio S4. Este proyecto debe contener el **código** desarrollado **y** la **documentación** generada.

- Nombre del fichero: *apellido_nombre*.zip.
- La **entrega** es **individual** y se subirá a eGela
- **Fecha límite de entrega**: Viernes 19 de febrero a las 23:55

Nuevo modelo del proyecto



Tareas para realizar en el laboratorio

1. Carga el proyecto de la semana pasada
2. Cambia el nombre del proyecto a S4*Mail* (utiliza para ello el menú *File-→ Rename*)
3. Cambia el nombre de la clase *Lab03Simulator* a *Lab04Simulator* (utiliza para ello el menú *File-→ Rename*)
4. Se debe documentar adecuadamente cada clase y operación que desarrolles. Recuerda que tienes un documento en eGela sobre la documentación en Java.
5. Modifica la clase *MailServer* del laboratorio anterior para que trabaje con ArrayList en vez de con Arrays
6. En la clase *MailServer*, modifica o añade los siguientes métodos con sus cabeceras y 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.
7. Vuelve a ejecutar el método testMailServer de la clase *Lab04Simulator* para probar que la nueva implementación de los métodos de la clase *MailServer* funciona correctamente
8. Genera la documentación del proyecto que has creado siguiendo los pasos descritos en la guía disponible en eGela. Esto se debe realizar antes de exportar los proyectos a entregar para que quede todo adecuadamente documentado.
9. Añade en la clase *MailServer* un método remove*Email* que dado un *Email* lo elimine de la lista de mensajes del servidor. Para hacer un uso adecuado de los métodos de la clase *ArrayList*, sobrescribe el método **equals** para la clase *Email*
10. Crea una clase nueva, denominada *MailClient*, dentro del paquete *mailsystem*. Esta clase tendrá el atributo `server (MailServer),` que representa el servidor de correo al que el cliente se conecta, y el atributo `userAddress (String)`. Implementa las siguientes constructoras y métodos de la clase:
    a. **Constructora** que recibe un parámetro por cada atributo de la clase (`server, userAddress`)
    b. **getUserAddress**: getter del atributo `userAddress`.
    c. **sendEmail**: Método que dados un receptor, una cabecera y un texto (todos ellos String) envía el mensaje de correo electrónico empleando el método **storeEmail** del servidor.
    d. **showMyInbox**: Método que muestra por pantalla todos los emails guardados en el servidor que hayan sido enviados al usuario actual.
    e. **showMySent**: Método que muestre por pantalla todos los emails guardados en el servidor que hayan sido enviados por el usuario actual.

11. Programa las instrucciones necesarias para probar que los métodos implementados para la clase *MailClient* funcionan correctamente Para ello, crea en la clase *Lab04Simulator* el método **testMailClient** que contenga las instrucciones que se indican a continuación. Realizar una llamada desde el método main para probarlo.
    a. Crea una instancia de la clase *MailServer* (*insServer*)
    b. Crea dos instancias de la clase *MailClient* (*mailClient1* y *mailClient2*), con los valores de atributos que estimes oportunos.
    c. Haz que se envíen tres Emails desde *mailClient1* y envía otro Email desde *mailClient2* .
    d. Muestra los mensajes enviados por *mailClient1* (utilizando el método de la clase *MailClient* que permita hacerlo).
    e. Muestra los mensajes recibidos por *mailClient1* (utilizando el método de la clase *MailClient* que permita hacerlo) .

Tareas complementarias

12. En la clase *MailServer*, añade los siguientes métodos con sus cabeceras y cuerpo:
    a. **obtainRemoveEmail**: Dada la dirección de un usuario del sistema, devuelve y borra del servidor el primer Email almacenado para ese usuario. Para facilitar la implementación de este método, implementa y utilizar el siguiente método:
        - **obtainFirstEmail**: Método privado que, dado el nombre de un usuario del sistema, obtiene el primer mensaje del Inbox del usuario en la lista de emails del servidor.
13. Añade en la clase *MailClient* el siguiente método:
    a. **obtainRemoveEmailFromInbox**: Método que, obtiene y elimina del servidor el primer email del Inbox de este usuario.
14. Programa las instrucciones necesarias para probar que los métodos implementados en las tareas complementarias funcionan correctamente.