Laboratorio S04: Clases e Interacción entre Objetos

Proyecto EHUMail

(PARTE 2) Genericidad de ArrayList

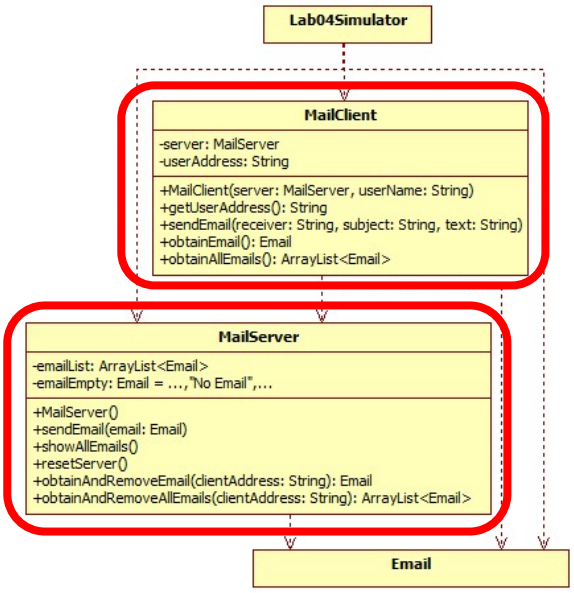
Objetivos:

- Además de los objetivos del laboratorio anterior entender las características y uso de la estructura de datos ArrayList

Proyecto y herramientas que vamos a utilizar:

- Proyecto: *EHUMail*
- Herramientas: entorno de desarrollo Eclipse, depurador (*debugger*) de Eclipse y JavaDoc

- **Contexto del proyecto:** Seguiremos completando el proyecto sin perder de vista todo lo aprendido en el laboratorio anterior. El sistema de correo electrónico permite que sus clientes se envíen mensaje como los descritos en el laboratorio anterior. El sistema de envíos entre clientes se gestiona mediante un servidor de correo que lo implementaremos en este laboratorio (no es un verdadero servidor, sino una simulación de su comportamiento). También implementaremos a los clientes del servidor de correo (los usuarios).



1. Domumenta e implementa la clase **`MailServer`.** Crea la clase **`MailServer`** en el package *mailsystem.* Esta clase se caracteriza por la lista de los mensajes que se van recibiendo/enviando. Además, deberá incluir los comportamientos que se indican:
   a. El **constructor** de la clase, de tal forma que inicialice la lista del servidor como vacía.
   b. **`sendEmail`**: Método que, dado un *Email* como parámetro, lo añade 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. Utiliza for-each para implementarlo.
   d. **`resetServer`**: vacía la lista de emails.
   e. **`obtainAndRemoveEmails`**: Dado el nombre de un usuario del sistema, devuelve y borra del servidor el primer email almacenado para ese usuario. ¡OJO! Es necesario buscar el mensaje y eliminarlo (si es que se encuentra), pero una vez que se encuentra se debe asegurar que no se sigue recorriendo la lista. Si no existiera ningún mensaje para ese usuario se devolverá un mensaje que tenga en el subject "No Email" (implementa una constante en la clase que contenga un objeto `Email` con esa información). Para hacer un uso adecuado de las operaciones de `ArrayList` sobre `Email`, sobreescribe el método **`equals`** de la clase **`Email`**.
   f. **`obtainAndRemoveAllEmail`**: Dado el nombre de un usuario del sistema, obtiene todos los mensajes recibidos (`ArrayList<Email>`) para el dicho usuario y los elimina del servidor. En este caso no utilices el método anterior, sino que debes implementarlo utilizando la clase `Iterator` para resolverlo.

2. Crea la clase **`Lab04Simulator`** con **`main`** (como la del laboratorio de la semana pasada) en el package

`simulators`, e implementa el método **`testMailServer`**. Programaremos las instrucciones necesarias para probar que los métodos implementados para la clase **`MailServer`** funcionan correctamente. Para ello, dentro de ese método:

- Crea la instancia de la clase **`MailServer`** (*mailServer1*) (o define un atributo estático de la clase para poder utilizarlo también en el siguiente método de prueba).
- Crea dos instancias de la clase **`Email`** (*email1* y *email2*), con los valores de atributos que estimes oportunos y envíalos al servidor.
- Haz que se muestren por pantalla todos los mails del servidor, **`showAllEmails`**.
- Antes de continuar, incluye en el **`main`** una llamada al subprograma **`testMailServer`**. Ejecútalo y comprueba que todo funciona como se espera. Comprueba que en el método **`testMailServer`** se han hecho llamadas a todos los métodos implementados en **`MailServer`** y si no es así, haz llamadas adecuadas a los mismos, para comprobar el correcto funcionamiento de la clase.
- Pon un punto de interrupción en la primera línea de código del método **`testMailServer`**. Ejecútalo paso a paso empleando el debugger para analizar el estado de los objetos en cada instrucción del código.

3. Documenta e implementa la clase **`MailClient`** tal y como se indica. 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 los siguientes constructores y métodos de la clase:
   a. **Constructor** que recibe un parámetro por cada atributo de la clase (server, clientName) inicializando cada uno de ellos.
   b. `El` getter de la dirección del cliente.
   c. **`sendEmail`**: Método que, dados el receptor, asunto (subject) y texto (todos ellos String) envía el mensaje de correo electrónico empleando el método **`sendEmail`** del servidor.
   d. **`obtainEmail`**: Método que obtiene y elimina del servidor el primer email recibido para ese usuario. También en este caso deberás hacer uso del método pertinente de la clase **`MailServer`**.
   e. **`obtainAllEmails`**: Obtiene (`ArrayList<Email>`) del servidor todos los mensajes recibidos para el cliente actual. Se habrán eliminado del servidor todos sus mensajes recibidos.

4. Amplía la clase **`Lab04Simulator`** e implementa el método **`testMailClient.`** Programaremos las instrucciones necesarias para comprobar que los métodos de **`MailClient`** funcionan correctamente. Para ello, crea dentro de ese método:
   - Una instancia de la clase **`MailServer`** (*mailServer1*) (o nada si anteriormente lo has definido como atributo `static`).
   - Dos instancias de la clase **`MailClient`** (*mailClient1* y *mailClient2*), con los valores de atributos que estimes oportunos (o defínelos como atributos `static` de la clase **`Lab04Simulator`**).
   - Haz que se envíen 3 mensajes con valores de subject y textos cualesquiera: (1) desde *mailClient1* a *mailClient2*, (2) desde *mailClient2 a mailClient1*, y (3) de *mailClient1* a *mailClient2*.
   - Haz que se muestren todos los mensajes del servidor, **`showAllEmails`**.
   - Añade al **`main`** una llamada al método **`testMailClient.`** Ejecútalo y comprueba el correcto funcionamiento de todo.
   - Obtén y elimina el primer mensaje del servidor para *mailClient1* (usa la operación de **`MailClient`** adecuada), y visualiza ese email por pantalla.
   - Vuelve a mostrar los emails del servidor para comprobar que se ha eliminado adecuadamente.
   - Comprueba que **`testMailClient`** ejecuta todos los métodos implementados en la clase **`MailClient`** y si no es así, haz llamadas adecuadas a los mismos, para comprobar su correcto funcionamiento.

5. Tienes en eGela un documento a parte con **tareas complementarias** a este laboratorio. Anímate a hacer todas las que puedas.

6. **Termina todo el Laboratorio y súbelo a eGela.**

   Terminadas todas las tareas debes exportar y subir el proyecto a eGela. Para exportar el proyecto, recuerda que tienes disponible en la pestaña **Sotfware** del curso de eGela información de cómo hacerlo.

   Tienes en eGela la tarea correspondiente para subir tu trabajo (**fecha límite de entrega viernes 19 de febrero de 2021**).