Nombre	Hora de entrega

Realizar una aplicación Java que simule un sistema de mensajería instantánea «Guasa». Las funcionalidades a implementar son:

public int registerUser(String phone, String email)

Este método añade un nuevo usuario al sistema «Guasa».
Hay que tener en cuenta que un usuario puede tener registrados muchos teléfonos. Todo usuario puede ser referenciado tanto por el email (dos usuarios no pueden tener el mismo email) como por el teléfono (un mismo teléfono sólo puede pertenecer a un único usuario)

Es necesario comprobar las siguientes condiciones:

- Si el teléfono ya estuviera registrado en el sistema, se devolvería el valor -1 (ver tabla)
- En caso contrario, si en el sistema existiera el usuario identificado con el email, a él se le añadiría el teléfono
- En caso contrario, si en el sistema no existiera un usuario con dicho email, se crearía un nuevo usuario y se le agregaría el teléfono.

El método devolverá un valor entero según la siguiente tabla:

Valor	Descripción
0	OK: Se ha creado el usuario y se le ha añadido el teléfono ó se ha añadido el teléfono a un usuario existente
-1	ERROR: el teléfono ya estaba registrado

public int sendMessageByPhone(String fromPhone, String toPhone, String text)

A través de este método un usuario envía un mensaje a otro. Se emplea los teléfonos tanto para identificar al usuario emisor (fromPhone) como al usuario receptor del mensaje (toPhone).

La condición indispensable para poder enviar un mensaje es que estén registrados tanto el teléfono del emisor como el del receptor.

Todo mensaje se compone de las siguientes propiedades: idMessage (identificador del mensaje), fromPhone (teléfono del emisor del mensaje) toPhone (teléfono del receptor del mensaje) isRead (dato boolean que indica si el mensaje ha sido o no leído por el receptor) text (texto del mensaje) y type.

El identificador del mensaje (idMessage) es un entero. El valor de ese entero lo calcula el sistema «Guasa» y representa el número de orden del mensaje enviado (el valor 1 para el primer mensaje, el valor 2 para el segundo, etc.). Este valor será independiente de quién sea el emisor o receptor del mensaje.

Cuando se envía un mensaje, su propiedad isRead tomará el valor false indicando de este modo que el mensaje ha sido enviado pero todavía no ha sido leído por el usuario al que se dirige.

El type del mensaje puede ser REGULAR o CONTROL. Un mensaje con type REGULAR es un mensaje que un usuario envía a otro. Un mensaje con type CONTROL es un mensaje enviado por el sistema «Guasa» para indicar la confirmación de lectura de un mensaje. Por defecto todos los mensajes son REGULAR a excepción de los enviados a través del método sendControl().

Cuando se envía un mensaje, éste se incluirá tanto en los mensajes recibidos por el destinatario (toPhone) como en los mensajes enviados por el emisor (fromPhone).

El método devolverá un entero cuyo valor será:

Valor	Descripción
idMessage	OK: se devuelve el idMessage del mensaje enviado
-1	ERROR: El usuario identificado por fromPhone no existe
-2	ERROR: El usuario identificado por toPhone no existe

public int sendMessageByEmail(String fromPhone, String toEmail, String text)

A través de este método un usuario envía un mensaje a cada uno de los teléfonos que tiene registrado el usuario receptor. Identificamos al usuario emisor por su teléfono (fromPhone) e identificamos al usuario receptor por su email (toEmail). Si, por ejemplo, el usuario receptor tuviera 4 teléfonos, se enviarán cuatro mensajes distintos, uno por cada teléfono.

El método devolverá un entero cuyo valor estará condicionado por:

Valor	Descripción
0	OK: Se ha enviado un mensaje a cada uno de los teléfonos del usuario receptor
-1	ERROR: El usuario identificado por fromPhone no existe
-2	ERROR: El usuario identificado por toEmail no existe

public Collection<Message> receiveMessages(String userPhone, String friendPhone)

Mediante esta funcionalidad el usuario identificado por el teléfono userPhone obtiene todos los mensajes intercambiados con un amigo (identificado por el teléfono friendPhone)

Si el usuario identificado por userPhone o bien el usuario identificado por friendPhone no existieran, el método devolvería null.

La colección devuelta incluirá tanto los mensajes enviados por el amigo al usuario (friendPhone → userPhone) como los mensajes enviados por el usuario a su amigo (userPhone → friendPhone). Todos estos mensajes estarán ordenados dentro de la Collection por el idMessage, garantizando de esta forma la secuenciación de los mensajes intercambiados entre ellos.

Si un mensaje REGULAR enviado por friendPhone y recibido por userPhone estuviese marcado como no leído, se marcaría el mensaje como leído y el sistema automáticamente enviaría un mensaje de control al amigo indicando que el mensaje acaba de leerse (este mensaje se enviará a través del método sendControlMessage()).

Si un mensaje CONTROL enviado por friendPhone y recibido por userPhone estuviese marcado como no leído, simplememnte se marcaría como leído y NO se enviaría mensaje de control.

La función main() deberá mostrar en pantalla cada uno de los mensajes (por cada uno se visualizará el valor de sus propiedades)

public void sendControlMessage(String fromPhone, String toPhone, int idMessage)

Este método será invocado automáticamente desde receiveMessages(). Su misión es la de enviar un mensaje con tipo CONTROL al usuario identificado por el teléfono toPhone por parte del usuario con teléfono fromControl indicaándole que acaba de leer el mensaje con identificador idMessage.
Este mensaje tendrá en la propiedad text el siguiente texto: "El mensaje [<valor idMessage>] ha sido leido".

Para entregar la aplicación:

Debes comprimir tu aplicación NetBeans que obligatoriamente tendrá como nombre tus apellidos y nombre (si te llamas Juan Carlos García Vaquero, a tu proyecto le deberás nombrar como GarciaVaqueroJuanCarlos). El fichero ZIP que subirás al Moodle también tendrá el mismo nombre que tu proyecto.

Calificación:

Aprobado: debes implementar y comprobar el correcto funcionamiento de las funcionalidades `registerUser`, `sendMessageByPhone y receiveMessages (en esta funcionalidad no se tendrá en cuenta el envío de mensajes de control).` Además, mediante comentarios al comienzo de cada clase, deberás justificar razonadamente la elección del contenedor que hayas utilizado para almacenar objetos.

Notable: además de lo anterior, debes implementar y comprobar el correcto funcionamiento del envío de mensajes de control en `receiveMessages` así como la implementación de `sendControlMessage.`

Sobresaliente: además de lo anterior, debes implementar y comprobar el correcto funcionamiento de `sendMessageByEmail`.

Se entiende que algo funciona cuando la ejecución del programa así lo demuestra