**Tarea Individual 25 - Uso de POP3, lectura de correos**

**Objetivo**

El objetivo de esta actividad es desarrollar una aplicación en Java que permita la conexión a un servidor POP3 (como Gmail), la autenticación mediante un usuario y contraseña, la obtención de los mensajes de correo del buzón de entrada, y la descarga de los mensajes en formato adecuado. Se utilizará la biblioteca javax.mail para interactuar con el servidor y realizar estas operaciones.

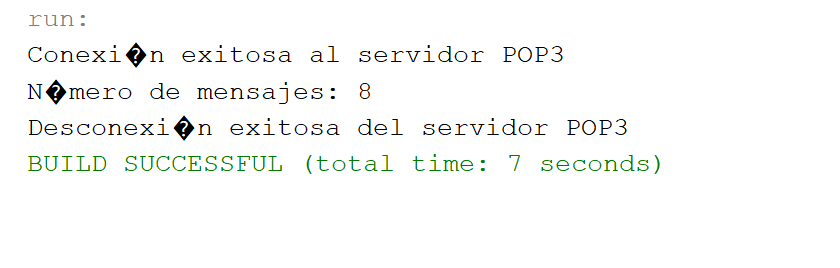
**Descripción de la Tarea**

Con base en el código proporcionado, se solicita a los alumnos realizar las siguientes modificaciones y desarrollos:

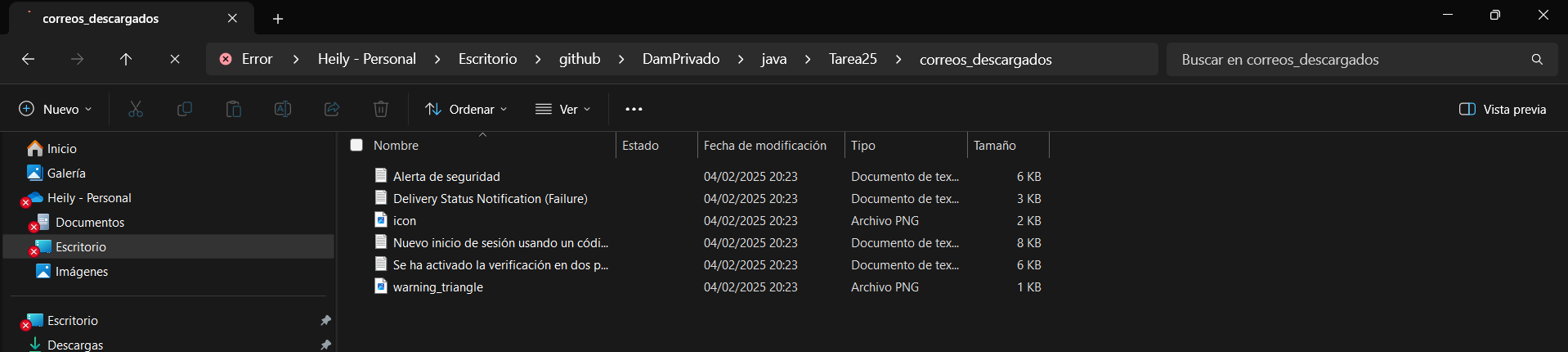
* **Conexión al servidor POP3:**
  + Utilizar la clase Store de la biblioteca javax.mail para establecer la conexión con el servidor POP3 especificado (ej. pop.gmail.com).
  + Configurar las propiedades de la sesión para utilizar POP3S (POP3 seguro) con SSL, utilizando mail.store.protocol y mail.pop3s.ssl.protocols.
* **Autenticación:**
  + Implementar la autenticación con el servidor POP3 utilizando las credenciales de un usuario (nombre de usuario y contraseña).
  + Mostrar un mensaje en la consola indicando si la autenticación fue exitosa o fallida.
* **Obtención y almacenamiento de correos:**
  + Conectar al buzón de entrada del servidor y obtener los mensajes utilizando el método getMessages de la clase Folder.
  + Para cada mensaje, guardar su contenido en un archivo en el sistema de archivos local, con un nombre basado en el asunto del correo. Si el mensaje tiene adjuntos, guardar también esos archivos.
  + Asegurarse de que los mensajes y sus adjuntos sean almacenados correctamente, manejando adecuadamente los tipos de contenido (texto, HTML, etc.).
* **Cierre de sesión y desconexión:**
  + Implementar el cierre de sesión con close para el objeto Folder y store.close() para la desconexión del servidor.
  + Mostrar un mensaje en la consola indicando si la desconexión fue exitosa o no.
* **Uso de TLS 1.2:**
  + props.put("mail.smtp.starttls.enable", "true");
  + props.put("mail.smtp.ssl.protocols", "TLSv1.2");
* **Requisitos adicionales:**
  + Gestionar las excepciones posibles (MessagingException, IOException) mostrando mensajes claros en la consola.
  + Comentar el código explicando las funcionalidades principales de cada bloque de código.
  + Asegurarse de que los archivos adjuntos se guarden correctamente en el sistema de archivos, incluyendo la gestión de las extensiones de los archivos.

**Entregable**

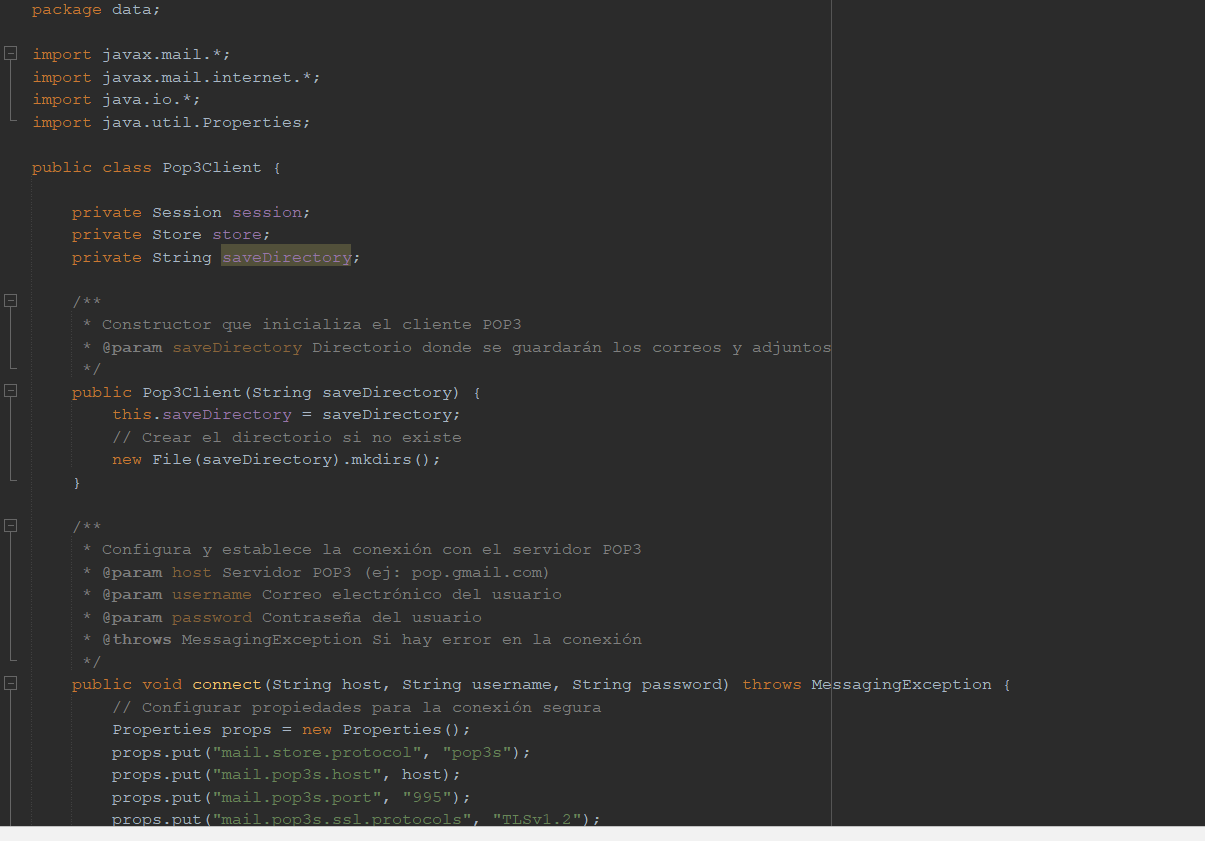
* Capturas de pantalla del código y de la ejecución del programa mostrando:
  + La conexión al servidor POP3 y la autenticación y el cierre de sesión y desconexión del servidor.

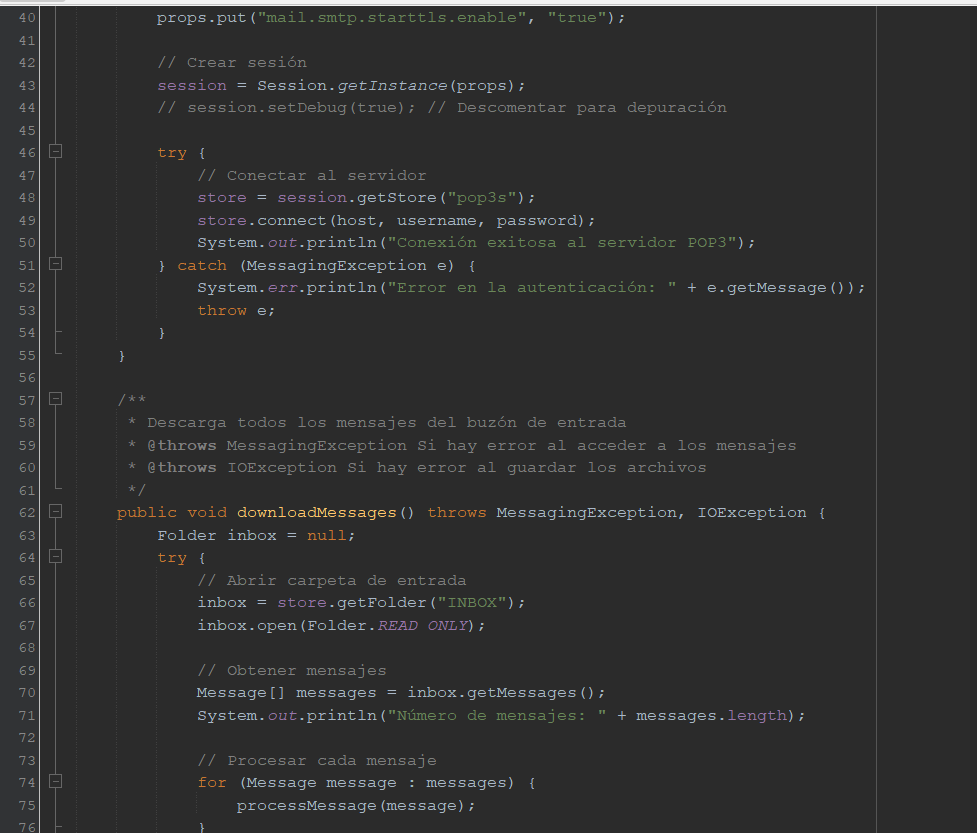


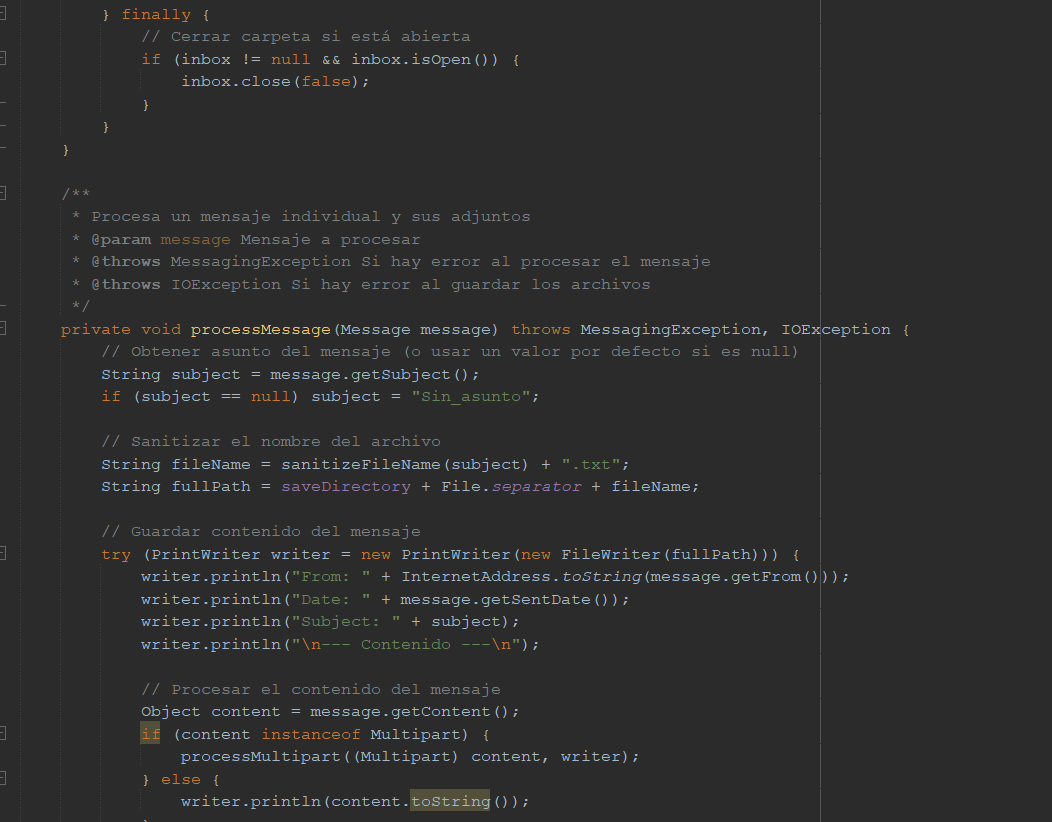
* + La lista de mensajes obtenidos del buzón de entrada y el proceso de almacenamiento de los correos en archivos.

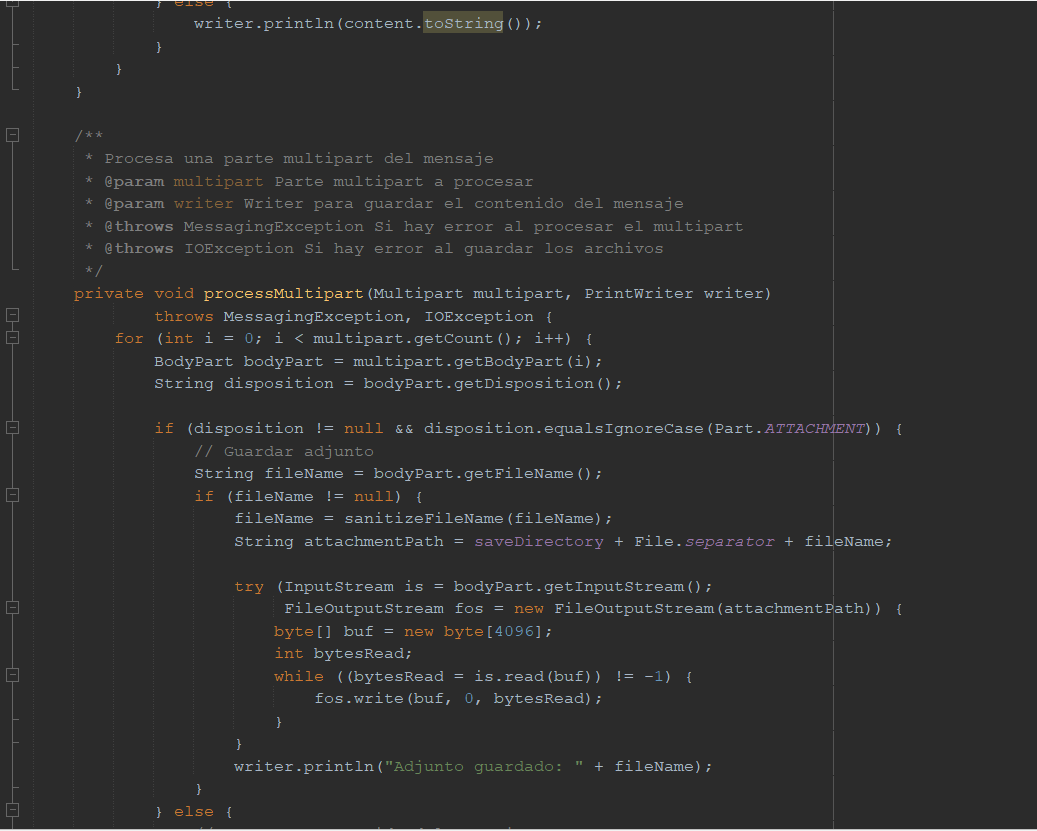


* Capturas de pantalla del código:









} else {

// Procesar contenido del mensaje

Object content = bodyPart.getContent();

if (content instanceof String) {

writer.println(content);

} else if (content instanceof Multipart) {

processMultipart((Multipart) content, writer);

}

}

}

}

/\*\*

\* Sanitiza el nombre del archivo para evitar caracteres inválidos

\* @param fileName Nombre del archivo a sanitizar

\* @return Nombre del archivo sanitizado

\*/

private String sanitizeFileName(String fileName) {

return fileName.replaceAll("[\\\\/:\*?\"<>|]", "\_");

}

/\*\*

\* Cierra la conexión con el servidor

\*/

public void disconnect() {

try {

if (store != null && store.isConnected()) {

store.close();

System.out.println("Desconexión exitosa del servidor POP3");

}

} catch (MessagingException e) {

System.err.println("Error al desconectar: " + e.getMessage());

}

}

/\*\*

\* Método principal para probar la funcionalidad

\*/

public static void main(String[] args) {

Pop3Client client = new Pop3Client("correos\_descargados");

try {

// Conectar al servidor (ejemplo con Gmail)

client.connect("pop.gmail.com", "womens.are.women@gmail.com", "ypuz bafp enyr kchb");

// Descargar mensajes

client.downloadMessages();

} catch (MessagingException e) {

System.err.println("Error de mensajería: " + e.getMessage());

} catch (IOException e) {

System.err.println("Error de E/S: " + e.getMessage());

} finally {

// Desconectar

client.disconnect();

}

}

}