**ACTIVIDAD 4**

**Enunciado**

Realizar un programa con sockets seguros que establezca un intercambio de información entre un servidor y un único cliente. El intercambio consistirá en que el cliente le enviará al servidor un número y el servidor le devolverá al cliente dicho número elevado al cuadrado.

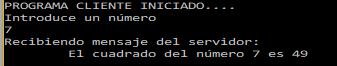
Con respecto al **servidor**:

* El nombre de su clase principal será *SocketServidorSeguro*
* El puerto por el que escuchará las peticiones del cliente será el 6020
* Con respecto al establecimiento de propiedades JSSE, usar el método adecuado de la clase Java *System,* que permita simplificar al máximo la sintaxis de la instrucción a usar desde la línea de comandos, para la ejecución del servidor
* Visualizará un mensaje de estado antes de crear el flujo de entrada del cliente
* Recibirá del cliente el valor de un número y lo elevará al cuadrado
* Devolverá al cliente el valor del número elevado al cuadrado, pero dentro de un stream cuyo texto completo aparece en la última línea que visualiza el cliente
* Cerrará los streams y sockets en el orden recomendado
* Realizará la necesaria gestión de excepciones
* El resultado esperado de su ejecución desde la línea de comandos será el siguiente:



Con respecto al **cliente**:

* El nombre de su clase principal será *SocketClienteSeguro*
* Con respecto al establecimiento de propiedades JSSE, usar el método adecuado de la clase Java *System,* que permita simplificar al máximo la sintaxis de la instrucción a usar desde la línea de comandos, para la ejecución del cliente
* Visualizará un mensaje de estado antes de capturar por el teclado un número de tipo primitivo *int*
* Enviará al puerto 6020 del servidor el número capturado por el teclado
* Recibirá del servidor el valor del número enviado, pero elevado al cuadrado
* Cerrará los streams y sockets en el orden recomendado
* Realizará la necesaria gestión de excepciones
* Suponiendo que el número enviado al servidor sea el 7, el resultado esperado de su ejecución desde la línea de comandos será el siguiente:



Con respecto a la **nomenclatura de los ficheros relacionados con el certificado** que usan el servidor y el cliente:

* Certificado: *Certificado.Apellido.cer*, donde Apellido es el primer Apellido de la persona que hace la actividad.
* Almacén de claves del servidor: *AlmacenSeguroApellido*, donde Apellido es el primer apellido de la persona que hace la actividad.
* Almacén de claves del cliente: *UsuarioAlmacenSeguroApellido*, donde Apellido es el primer apellido de la persona que hace la actividad.

Con respecto a las **contraseñas** relacionadas con el certificado:

* Password del Servidor: *dm2server*
* Password del Cliente: *dm2client*

Con respecto al **alias** a utilizar en la gestión del certificado desde la línea de comandos:

* *claveApellido*, donde Apellido es el primer Apellido de la persona que hace la actividad.

Recomendaciones a la hora de realizar las **pruebas** de esta actividad:

* Si los programas Java están incluidos en algún paquete, al ejecutarse desde la línea de comandos (con permisos de administrador), es preciso indicar el nombre del paquete más un punto, antes del nombre del archivo que contiene los bytecodes (\*.class) del programa a ejecutar.
* Iniciar la ejecución del servidor antes que la del cliente.
* Si se producen errores durante la ejecución de los programas desde la línea de comandos, activar el debug de Java con la opción –Djavax.net.debug=ssl

**EVIDENCIAS**

Incluye a continuación evidencias de cómo has conseguido mediante la herramienta keytool de Java lo siguiente:

Incluye a continuación evidencias de cómo has conseguido lo siguiente:

1. Comprobar que el IDE Eclipse ha generado los bytecodes (\*.class) del Servidor y del Cliente en el correspondiente directorio del workspace de Eclipse.
2. Crear el almacén de claves del servidor desde la línea de comandos en el directorio en el que están los bytecodes de Java.
3. Exportar a un fichero el almacén de claves del servidor.
4. Importar el fichero en un almacén de claves del cliente.
5. Ejecutar el socket seguro de servidor desde la línea de comandos.
6. Ejecutar el socket seguro de cliente desde la línea de comandos.

**Resolución de problemas**

**REQUISITOS**

1. Entregar la actividad en un archivo comprimido que contenga lo siguiente:
   1. Los ficheros \*.java empleados para el servidor y el cliente.
   2. Los archivos que contengan el almacén de claves del servidor, el almacén de claves del cliente y el certificado a usar entre el servidor y el cliente.
   3. Fichero original del enunciado de la actividad con el apartado de Evidencias cumplimentado
2. Los archivos comprimidos seguirán la siguiente nomenclatura:

*UD5\_Actividadn\_apellido*

donde:

*n*: valdrá 4, representando el número de la actividad

*apellido*: será el primer apellido del alumno

**CRITERIOS DE CORRECCIÓN**

1. Cumplimiento de los requisitos (15%).
2. Hacer lo que se indica en el enunciado (30%).
3. Evidencias (25%).
4. Claridad del código Java (15%).
5. Documentación del código Java (15%).
6. Explicación detallada de cómo se han resuelto problemas que hayan podido surgir en la realización de la actividad (hasta un 15% adicional sobre la nota obtenida). Dicha explicación se incluirá al principio de los archivos de código fuente Java mediante líneas de comentarios. La primera línea de comentarios sólo incluirá lo siguiente: // Resolución de problemas.
7. Variables no usadas (-15%).
8. Librerías no usadas (-15%).