EXAMEN RA3

CE-A-B-C-D-E-F-G-H

TABLA DE CONTENIDO

NOTAS Y CONSEJOS

1) Antes de hacer algo, **lee el examen atentamente**.
2) Es un **examen individual**, no preguntes a tus compañeros para que te ayuden.
3) **Revisar el programa es tu responsabilidad,** si no se ejecuta correctamente cuando lo pruebas, no se ejecutará cuando tu profesor lo revise.
4) **Crea un Proyecto Maven** para una aplicación de Java usando el IDE Netbeans 15 y el **jdk17** o **jdk19**. ¡¡si usas otro tendrás problemas con las bibliotecas y no corregiré tu código!!
5) **Nombre del proyecto**:
    a. Client: *Apellido1Apellido2Nombre_Ex_RA3_Client_20240115*
    b. Server: *Apellido1Apellido2Nombre_Ex_RA3_Server_20230115*

Recuerda que vamos a implementar:

- Crear hilos.
- Crear múltiples hilos.
- Crear sockets TCP.
- Documentación.

ENTREGAS

Los proyectos **maven** (antes de entregarlos, comprímelos en un archive .zip con el mismo nombre del proyecto!).

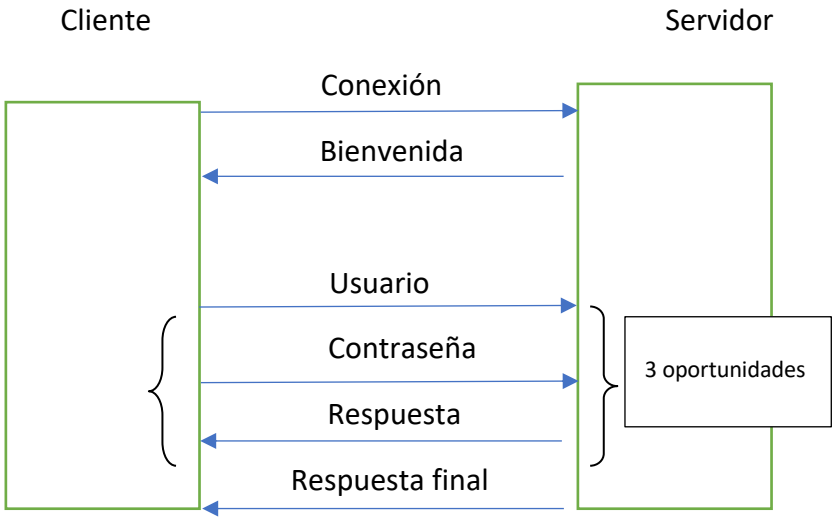
PROBLEMA

CONCEPTO

Hay que implementar la simulación de un *login* a un servidor.

Proceso:

1. Primero el cliente inicia conexión y recibe la bienvenida del servidor.
2. A continuación, el cliente le mandará al servidor un usuario y una contraseña.
    a. El usuario correcto para entrar será DAM**Nombre** (donde **Nombre** es tu nombre real) y la contraseña será **Apellido (**donde **Apellido** es tu apellido real).
3. Posteriormente el servidor comprobará si son correctos el usuario y la contraseña enviados por el cliente. Si el usuario o la contraseña no son correctos, el servidor le responderá al cliente que el login no ha sido incorrecto y además, el número de **oportunidades** que le quedan.
    a. El cliente tendrá **3** oportunidades para acertar el usuario y la contraseña.
4. En el caso de que las credenciales sean correctas o se acaben las oportunidades se cerrará la conexión enviando el servidor una respuesta final.



- **El servidor se ejecuta continuamente sirviendo a múltiples clientes simultáneamente usando subprocesos.**
- La conexión se acaba cuando:
    - Se acaban las oportunidades o las credenciales son correctas.

FUNCIONALIDAD

SERVIDOR

- **Puerto** de escucha: 9666.
- **Protocolo** de la capa de transporte: TCP.
- Maneja múltiples clientes simultáneamente **multihilo**.
- **Maneja** el número de clientes gestionados.
- Manda el mensaje de **bienvenida** al cliente:

```
*******************************
*        SERVIDOR DE LOGIN
*.    por "Nombre del estudiante"     *
*******************************
    - Eres el cliente número
    - Por favor, introduce un usuario y
      contraseña para entrar.
```

- **Procesa** el usuario y contraseña enviados por el cliente:
    - Si son correctos:
        - El servidor le responde al cliente con el mensaje: "Credenciales correctas".
        - Se envía el mensaje "BIENVENIDO AL SERVIDOR" y finaliza la conexión.
    - Si son incorrectos y no ha gastado las **3** oportunidades:
        - El servidor le responde al cliente con el mensaje: "Credenciales incorrectas, vuelve a intentarlo te quedan **X** oportunidades". Donde X es el número de oportunidades o intentos que le quedan al cliente para intentar conectarse.
    - Si son incorrectos y ha gastado las **3** oportunidades:
        - "Credenciales incorrectas, ya no te quedan oportunidades".
        - Se envía el mensaje "NO SE PUDO CONECTAR AL SERVIDOR" y finaliza la conexión.

CLIENTE

1. Pregunta por la IP del servidor o FQDN y el puerto de escucha del servidor.
2. Se conecta al servidor.
3. Recibe e imprime por pantalla el mensaje de bienvenida del servidor.
4. Pregunta al usuario que introduzca un usuario.
5. Pregunta al usuario que introduzca su contraseña.
6. Recibe la respuesta del servidor de si sus credenciales son correctas o no y las oportunidades que le quedan.
7. Recibe el mensaje final del servidor.

ESTRUCTURA

SERVIDOR

El programa debe contener al menos las siguientes clases:

- Clase **Main** (la que contiene el método main).
- Clase **Servidor**.
- Clase para manejar a un cliente.

CLIENTE

El programa debe contener al menos las siguientes clases:

- Clase **principal** (la que contiene el método main).
- Clase **Cliente**.

El nombre de las clases depende de ti, pero debe tener sentido…

DOCUMENTACIÓN

Debes crear un comentario encima de cada declaración de clase, cada declaración de método, explicando lo que hace.

EJEMPLO DE EJECUCIÓN

CAPTURA DE PANTALLA

Con credenciales correctas:

```
Por favor, introduce la IP o FQDN del servidor:
localhost
Por favor, introduce el puerto de escucha del servidor:
9666
*********************************
    Examen RA3 - SERVIDOR Login
    por Joaquín Franco
*********************************

Eres el cliente número: 1
Por favor introduce un usuario y una contraseña para entrar


Por favor, introduce el usuario:
DAMJoaquin
Por favor, introduce la contraseña:
Franco
Credenciales correctas
BIENVENIDO AL SERVIDOR

Process finished with exit code 0
```

Fallando tres veces las credenciales:

```
Eres el cliente número: 2
Por favor introduce un usuario y una contraseña para entrar


Por favor, introduce el usuario:
erroneo
Por favor, introduce la contraseña:
erronea
Credenciales incorrectas, vuelve a intentarlo te quedan: 2 oportunidades
Por favor, introduce el usuario:
erroneo
Por favor, introduce la contraseña:
erronea
Credenciales incorrectas, vuelve a intentarlo te quedan: 1 oportunidades
Por favor, introduce el usuario:
erroneo
Por favor, introduce la contraseña:
erronea
Credenciales incorrectas, ya no te quedan oportunidades
NO SE PUDO CONECTAR AL SERVIDOR

Process finished with exit code 0
```

CORRECCIÓN – RA3

a) Se han identificado escenarios que precisan establecer comunicación en red entre varias aplicaciones.		
Logro	Peso %	Valoración
El estudiante ha sido capaz de identificar correctamente que tenía que establecer una comunicación de red entre varias aplicaciones implementando correctamente el modelo cliente-servidor a la hora de crear la aplicación. Además, la aplicación ejecuta correctamente la comunicación.	100	Sí/No

b) Se han identificado los roles de cliente y de servidor y sus funciones asociadas.		
Logro	Peso %	Valoración
Se han utilizado los roles correctos creando una aplicación independiente para cliente y para servidor utilizando las clases correctas.	100	Sí/No

c) Se han reconocido librerías y mecanismos del lenguaje de programación que permiten programar aplicaciones en red.		
Logro	Peso %	Valoración
El estudiante ha utilizado la clase correcta para crear el socket cliente.	33	Sí/No
El estudiante ha utilizado la clase correcta para crear el socket de escucha del servidor.	34	Sí/No
El estudiante ha utilizado las clases correctas para crear los flujos	33	Sí/No

d) Se ha analizado el concepto de socket, sus tipos y características		
Logro	Peso %	Valoración
El estudiante ha utilizado las clases correctas y los métodos adecuados para crear e interactuar con el socket de conexión.	50	Sí/No
El estudiante ha obtenido correctamente el socket de conexión a partir del socket de escucha del servidor.	50	Sí/No

e) Se han utilizado sockets para programar una aplicación cliente que se comunique con un servidor.		
Logro	Peso %	Valoración
Se ha creado la aplicación cliente implementando correctamente un socket TCP con las características indicadas y que funciona correctamente.	60	Sí/No
Pide correctamente al usuario la IP del servidor y su puerto, y la aplicación funciona correctamente.	20	Sí/No
Imprime por pantalla la ayuda de usuario funcionando correctamente la aplicación.	20	Sí/No

f) Se ha desarrollado una aplicación servidor en red verificando su funcionamiento.		
Logro	Peso %	Valoración
Se ha creado la aplicación servidor implementando correctamente un socket TCP con las características indicadas y que funciona correctamente.	40	Sí/No
Utiliza el puerto de escucha indicado en el enunciado.	10	Sí/No
Gestiona correctamente de múltiples clientes.	20	Sí/No
Gestiona correctamente la conexión persistente con el cliente hasta que éste la termina.	30	Sí/No

g) Se han desarrollado aplicaciones que utilizan sockets para intercambiar información.		
Logro	Peso %	Valoración
Se establece correctamente el mecanismo de transmisión de información a través de sockets utilizando flujos.	10	Sí/No
Se utiliza el flujo más óptimo para la transferencia de la información.	10	Sí/No
Se cumple con el requisito de la información transmitida mandando el usuario y contraseña y el servidor y el cliente actuando en consecuencia.	30	Sí/No
Gestiona correctamente el bucle de petición y terminando éste cuando se acaban las oportunidades o se ha *logeado* correctamente.	20	Sí/No
Imprime por pantalla correctamente la respuesta del servidor después de realizar la conexión y transferencia correctamente.	20	Sí/No
Imprime por pantalla correctamente la bienvenida del servidor.	10	Sí/No

h) Se han utilizado hilos para implementar los procedimientos de las aplicaciones relativos a la comunicación en red		
Logro	Peso %	Valoración
Se crea correctamente una clase que permite crear un hilo independiente para la gestión de cada cliente y la aplicación funciona correctamente.	30	Sí/No
Se transfiere correctamente el socket a la clase que gestiona los clientes individualmente y la aplicación funciona correctamente.	70	Sí/No