11 Ejercicios paradigma cliente/servidor.

0. Calculadora distribuida

- concurrente
- sin estado
- protocolo: una serie de operaciones disponibles y el número de operandos de cada una de ellas.
- proceso: cliente pide protocolo, recibe operaciones, envía operación y lista operandos, recibe resultado.

---

1. Servicio “echo”

- iterativo
- sin estado
- sin protocolo
- proceso: cliente realiza una conexión, recibe del servidor un mensaje con un número aleatorio y termina la conexión.

---

2 Formateador de textos

- iterativo
- sin estado
- protocolo: operaciones de conversión de textos (capitalizar, a mayúsculas, a minúsculas …)
- proceso: cliente pide protocolo, recibe operaciones, envía operación y texto, recibe resultado.

---

3. Acumulador de valores

- iterativo
- con estado
- protocolo: conjunto de operaciones, entre las que se encuentra enviar valor (o valores) a distintos acumuladores cuyo nombre elige el cliente y operaciones sobre esos acumuladores (suma, media, máximo, mínimo)
- proceso: cliente pide protocolo, recibe operaciones, envía operación y texto, recibe bien un aceptado, bien resultado.

ejemplo:

>> acumular enteros 1 2 3 4

<< ok

>> acumular decimales 1.5

<<ok

>> media enteros

<< 2.5

4. Servicio mensajería

Deseamos implementar un servidor para una **sala de chat** (parecido a un grupo de Whatsapp).

Para ello reutilizaremos un paquete **SistemaVentanas**, proporcionado para el ejercicio y que viene apropiadamente comentado (carpeta doc dentro del jar). Para usarlo solo hay que heredar de la clase **SistemaVentanas** de dicho paquete en la parte cliente.

Para resolver la tarea la parte cliente y el servidor deben estar en proyectos distintos y comunicarse mediante **conexión**.

En cuanto al **cliente** hay que implementar las funcionalidades de dicha clase heredada y las necesarias para comunicarse con el servidor (enviar y recibir mensajes).

Para la parte **servidor** hay que implementar un servidor concurrente con las siguientes funcionalidades:

1. ***Estructuras de datos:***
    - El servidor contiene una estructura de datos donde se almacena parejas de elementos **usuario-contraseñ**a.
    - El servidor contiene una estructura de datos que almacena, para cada usuario registrado, una **lista de los últimos 50 mensajes de ese usuario**.
    - El servidor contiene una estructura de datos con los **usuarios que están conectados** en ese momento.
    - **No** es necesario implementar persistencia, los usuarios pueden ser leídos de fichero, desde consola, o estar predefinidos en el propio código.

2. ***Comportamiento***:
    - Cuando el servidor **recibe un mensaje** desde cualquier usuario lo **interpreta**, lo **almacena** en la estructura apropiada y, **ejecuta** la acción adecuada.
    - Los mensajes pueden ser de varios tipos, pero deben tener un formato y comportamiento prefijado. Para esta parte pueden ser de los siguientes tipos, pero en un fututo pueden añadirse más:
        i. *Mensaje de un usuario*: "**msg:<nick>:<texto>**". Envía dicho mensaje a todos los usuarios conectados.
        ii. *Login de un usuario*: "**login:<nick>:<pass>**". Si la pareja clave-contraseña es válida, notifica al cliente que tiene acceso al sistema.
        iii. *Logout de un usuario*: "**logout:<nick>**". Elimina al usuario de los usuarios conectados.

        Por ejemplo "*msg:pepe:hola a todos!*" es un formato de mensaje de usuario.