UNIDAD 3. PROGRAMACIÓN DE COMUNICACIONES EN RED.

PRÁCTICAS

(8 puntos) Servidor de Subastas en Tiempo Real

1. Descripción General

El objetivo es desarrollar una aplicación distribuida Cliente/Servidor en Java que simule una **Casa de Subastas en Tiempo Real**. El sistema debe permitir que múltiples usuarios (clientes) se conecten simultáneamente para pujar por artículos. El servidor debe gestionar el estado de la subasta, cronometrar los tiempos y adjudicar el artículo al mejor postor, garantizando la integridad de los datos en un entorno concurrente.

2. Requisitos Técnicos Obligatorios

A. Comunicación y Arquitectura (Base Unidad 3)

- **Modelo TCP/IP:** La comunicación principal (pujas, mensajes del sistema) debe realizarse mediante **Sockets TCP** (clases Socket y ServerSocket) para garantizar que ningún dato se pierda.
- **Modelo Cliente/Servidor:**
  - **Servidor:** Debe publicar un puerto y quedar a la espera de peticiones accept().
  - **Cliente:** Debe conectarse al servidor mediante IP y Puerto[3].
- **Intercambio de Objetos:** La comunicación no debe ser solo texto plano. Se deben enviar objetos serializados (por ejemplo, clase Apuesta o Articulo que implementen Serializable) utilizando ObjectInputStream y ObjectOutputStream.

B. Concurrencia y Gestión de Hilos

Servidor Concurrente: El servidor no puede ser secuencial. Debe atender a múltiples clientes simultáneamente. Se **exige** el uso de un **Pool de Hilos** mediante la interfaz ExecutorService para gestionar las conexiones entrantes, evitando crear un hilo manual por cada conexión si la carga es alta.

- **Sincronización:** Dado que varios clientes intentarán pujar por el mismo artículo a la vez, se producirá una **condición de carrera** sobre la variable del "precio actual". Es obligatorio proteger el recurso compartido (el objeto Artículo/Subasta) utilizando bloques synchronized o cerrojos (Lock) para garantizar la exclusión mutua y evitar que dos pujas se sobreescriban.

C. Protocolo de Estado

- El servidor debe implementar un autómata o **máquina de estados** para gestionar el flujo, tal y como se explica en el temario (Transiciones).

- Estados mínimos esperados: ESPERANDO_JUGADORES -> SUBASTA_INICIADA -> TIEMPO_FINALIZADO -> ADJUDICADO.

3. El Reto Extra (Dificultad Avanzada)

"Servicio de Descubrimiento Automático (Service Discovery) y Difusión"
En lugar de que el cliente tenga que escribir manualmente la IP del servidor (ej. localhost o 192.168.1.50), se debe implementar un mecanismo híbrido TCP/UDP:

1. **Broadcast/Multicast del Servidor:** El servidor, además de escuchar por TCP, debe tener un hilo independiente que emita periódicamente un **Datagrama UDP** (o use un grupo Multicast) anunciando su presencia (Nombre de la sala de subastas y Puerto TCP de escucha) a toda la red local.
2. **Cliente Inteligente:** Al iniciar, el cliente se queda escuchando por UDP. En cuanto detecta el paquete del servidor, extrae la IP y el puerto automáticamente y lanza la conexión TCP para empezar a pujar.
3. *Justificación:* Esto obliga a mezclar la fiabilidad de TCP (para el dinero/pujas) con la velocidad y capacidad de difusión de UDP (para el descubrimiento), un patrón común en sistemas reales (como impresoras en red o Chromecasts).

Criterios de Calificación (Total: 8 Puntos)

1. Implementación de Sockets TCP y Comunicación (2 Puntos)

- Establecimiento correcto de la conexión Socket y ServerSocket. El servidor queda a la espera y acepta conexiones.
- Uso correcto de flujos de entrada/salida para objetos (ObjectOutputStream), asegurando que las clases sean Serializable.
- Implementación de un protocolo de comunicación claro (el cliente envía una oferta, el servidor responde si es aceptada o rechazada por ser baja).

2. Gestión de la Concurrencia (2 Puntos)

- Implementación correcta del servidor concurrente. Uso obligatorio de ExecutorService (Executors.newCachedThreadPool() o similar) para manejar los clientes.
- Seguridad en Hilos (Thread-Safety). Protección correcta de la variable "precio actual" y "ganador actual" mediante synchronized.

3. Lógica de Negocio y Estados (1.5 Puntos)

- Implementación correcta del diagrama de estados en el servidor. El servidor no debe aceptar pujas si el estado no es SUBASTA_INICIADA. Debe manejar correctamente la transición a ADJUDICADO.

4. Reto Extra: Descubrimiento UDP/Multicast (2.5 Puntos)

- El servidor envía correctamente datagramas (DatagramPacket) a una dirección de difusión o grupo Multicast.
- El cliente es capaz de recibir el paquete, extraer la IP del emisor y conectarse automáticamente por TCP sin intervención del usuario.
- Manejo correcto de excepciones y cierre de recursos (sockets UDP y TCP).

Recomendaciones.

- **Monitorización:** Utiliza System.currentTimeMillis() para medir los tiempos de respuesta entre que llega una puja y se confirma.
- **Interbloqueos:** Cuidado si decides que el servidor notifique a todos los clientes (broadcast) dentro de un bloque sincronizado, ya que podría causar un cuello de botella o bloqueo si un cliente es lento.

1. **(2 puntos)** Crea un manual técnico y de uso del ejercicio realizando capturas del código, comentando qué hace cada parte y por qué usas ese código (método, clase, etc.) y no otro (cuando sea relevante), así como ejemplos de uso con las capturas de los resultados. Recuerda crear una portada, un índice y estructurar el contenido.