Descripción del proyecto:

El proyecto consiste en un pequeño programa cliente-servidor para gestionar una biblioteca virtual donde los usuarios pueden realizar operaciones como:

- Listar libros (se pueden aplicar filtros de búsqueda)
- Alquilar/Devolver un libro

El sistema usa una estructura cliente-servidor utlizando JAVA RMI lo que permite la comunicación remota entre los dos agentes.

La clase ServidorRMI gestiona la lógica de negocio a través de la clase BibliotecaRemota que es la que implementa y expone los métodos definidos por la interfaz BibliotecaInterfaceRMI a los clientes que conecten al servidor.

Dicha interfaz a su vez hereda de UnicastRemoteObject que es la clase que le permite que sus métodos sean accesibles para clientes remotos.

Justificación soluciones adoptadas:

La implementación del sistema RMI viene dada por requisitos del enunciado. Aunque se pide usar como base el código ofrecido por el profesor en el repositorio de GitHub yo he decidido no usar una clase Hilo que maneje las conexiones de los clientes al servidor.

Esto ha sido así porque al usar la tecnología RMI podemos abstraer la comunicación entre el cliente y el servidor ya que RMI maneja automáticamente la concurrencia y la creación de hilos por cada conexión al servidor.

En mi proyecto solo he sido capaz de detectar un punto donde podría haber conflicto entre varios hilos produciéndose la llamada carrera de hilos. Este punto es en los métodos que actualizan el estado de los objetos Libro (el atributo de clase ‘disponible’).

La solución adoptada es incluir el comando ‘synchronized’ solo en el caso donde se va a producir el cambio de estado del objeto. Esto es más interesante que declararlo al método entero porque esto solo bloquea el cambio de estado de dicho objeto evitando que varios hilos puedan modificarlo a la vez ocasionando resultados no deseados.

Si lo hubiera declarado en la cabecera del método este quedaría bloqueado por completo a un hilo creando un cuello de botella mientras que la solucion adoptada deja al resto de hilos usar el método menos cuando lleguen a intentar modificar el estado del mismo objeto.

Finalmente comentar el por qué los métodos de la BibliotecaRemota devuelven objetos de clase Libro o List<Libro> en lugar de imprimir en pantalla directamente el resultado. Esto es así porque esta clase trabaja del lado del servidor por lo tanto todo lo que imprima se hará en la consola que pertenezca al servidor dejando a ciegas al cliente. Por ello he decidido que sus métodos devuelvan por un lado los objetos de las clases que antes he mencionado y por otro lado impriman mensajes que sirvan de ‘log’ en la consola del servidor.