Protocolo de Diseño de Monitores en Java con API de Alto Nivel

1. Decidir qué datos encapsular en el monitor.
2. Construir un monitor teórico, utilizando tantas variables de condición como sean necesarias.
3. Usar señalización SC en el monitor teórico.
4. Implementar en java mediante una clase
   - Instanciar un objeto de clase `ReentrantLock, L`
   - Escribir un método por cada procedimiento del monitor teórico. Todo su código irá encapsulado entre `L.lock()` y `L.unlock()`
   - Hacer los datos encapsulados con `private`.
   - Obtener tantos objetos `Condition` como variables de condición tenga el monitor teórico (`c=L.newCondition()`)
   - Sustituir cada `wait(variable_condición)` por una condición de guarda `while(!condicion) try{c.await()}…`
   - Sustituir cada `send(variable de condición)` por una llamada a `c.signal() o c.signalAll()`
   - Escribir el código de inicialización del monitor en el constructor de clase

Técnica de Diseño de Monitores en Java

1. Decidir qué datos encapsular en el monitor.
2. Construir un monitor teórico, utilizando tantas variables de condición como sean necesarias.
3. Usar señalización SC en el monitor teórico.
4. Implementar en java
    1. Escribir un método `synchronized` por cada procedimiento.
    2. Hacer los datos encapsulados con `private`.
    3. Sustituir cada `wait(variable_condición)` por una condición de guarda `while(!condicion) try{wait()}`…
    4. Sustituir cada `send(variable de condición)` por una llamada a `notifyAll()`
    5. Escribir el código de inicialización del monitor en el constructor del mismo