Recursos compartidos por los hilos en Kotlin (gestión de transacciones bancarias concurrentes)

[crayon-6a9d696fac61c313201049/]
Este código en Kotlin implementa un programa de simulación de gestión de transacciones bancarias concurrentes utilizando hilos y bloqueos (locks) en un contexto de una cuenta bancaria. Aquí está la explicación detallada del código:

1. Se define una clase llamada CuentaBancaria que representa una cuenta bancaria con un saldo inicial. Esta clase tiene dos métodos principales:
    - depositar(cantidad: Int): Este método permite depositar una cantidad específica en la cuenta. Utiliza un bloqueo (lock) mediante ReentrantLock para garantizar que las operaciones de depósito sean seguras en entornos concurrentes. Después de depositar, muestra un mensaje que indica la cantidad depositada y el saldo actual.
    - retirar(cantidad: Int): Este método permite retirar una cantidad específica de la cuenta. También utiliza un bloqueo para garantizar que las operaciones de retiro sean seguras. Verifica si hay suficiente saldo antes de retirar y muestra mensajes apropiados en función del resultado.
2. En la función main, se crea una instancia de CuentaBancaria llamada cuenta con un saldo inicial de 1000.
3. Se crean cuatro hilos (hilo1, hilo2, hilo3, y hilo4) que ejecutan diferentes secuencias de operaciones en la cuenta:
    - hilo1: Realiza 5 depósitos de $100 cada uno, con

un intervalo de 100 ms entre cada depósito.
   - hilo2: Realiza 3 retiros de $200 cada uno, con un intervalo de 150 ms entre cada retiro.
   - hilo3: Realiza 4 depósitos de $50 cada uno, con un intervalo de 200 ms entre cada depósito.
   - hilo4: Realiza 2 retiros de $300 cada uno, con un intervalo de 100 ms entre cada retiro.
4. Se utiliza join() en cada hilo para asegurarse de que todos los hilos completen sus operaciones antes de imprimir el saldo final.
5. Finalmente, se imprime el saldo final de la cuenta después de que todos los hilos hayan completado sus transacciones.

Este código ilustra cómo se pueden utilizar bloqueos (locks) para gestionar transacciones bancarias concurrentes de manera segura, evitando condiciones de carrera y garantizando la consistencia de los datos en una aplicación multi-hilo.

Las llamadas lock.lock() y lock.unlock() se utilizan para crear secciones críticas del código en las funciones depositar y retirar. Aquí está cómo funcionan:

1. lock.lock(): Cuando un hilo llama a lock.lock(), adquiere un bloqueo en el objeto lock, lo que significa que ningún otro hilo puede entrar en una sección crítica protegida por el mismo bloqueo hasta que se libere el bloqueo. En este caso, se usa para asegurar que solo un hilo a la vez puede ejecutar las secciones críticas dentro de las funciones depositar y retirar.
2. lock.unlock(): Después de que un hilo ha terminado de ejecutar las operaciones críticas dentro de una sección protegida por el bloqueo, debe llamar a lock.unlock() para liberar el bloqueo. Esto permite que otros hilos adquieran el bloqueo y ejecuten las secciones críticas de manera concurrente.

El objetivo de utilizar bloqueos es garantizar que las

operaciones de depósito y retiro se realicen de manera segura y eviten condiciones de carrera, donde varios hilos intentan acceder y modificar el saldo al mismo tiempo. Los bloqueos permiten que solo un hilo acceda a la sección crítica a la vez, asegurando la coherencia de los datos.