Ejemplo de acceso sincronizado a un contador en Kotlin utilizando corrutinas

[crayon-6a9d60bb4738c398920370/]

Explicación línea a línea:

[crayon-6a9d60bb47393671944846/]

En esta sección se importan las bibliotecas necesarias. kotlinx.coroutines es la biblioteca que proporciona soporte para la programación asíncrona en Kotlin utilizando corrutinas. kotlin.concurrent.thread es una biblioteca estándar de Kotlin que permite crear hilos para ejecutar tareas de forma concurrente.

[crayon-6a9d60bb47395008396185/]

Aquí se define la clase Counter, que tiene un contador privado llamado count. Los métodos increment y getCount están sincronizados utilizando synchronized(this) para asegurar el acceso seguro al contador desde hilos múltiples. El método increment incrementa el contador en uno, mientras que el método getCount devuelve el valor actual del contador.

[crayon-6a9d60bb47397101544494/]

En la función main, se crea una instancia de la clase Counter llamada counter. A continuación, se crea un hilo adicional utilizando thread para ejecutar un bucle que incrementa el contador en uno y espera 10 milisegundos entre cada incremento.

Dentro de runBlocking, se utiliza withContext(Dispatchers.IO)

para crear un contexto en el que se ejecutará el siguiente bloque de código de manera asíncrona en el hilo de E/S. Esto es necesario para evitar bloquear el hilo principal mientras se accede al contador.

Dentro de withContext, se ejecuta otro bucle que muestra el valor actual del contador utilizando counter.getCount() y lo imprime en la consola. También se introduce un retraso de 10 milisegundos entre cada iteración del bucle.

En resumen, este código crea un contador sincronizado que puede ser accedido de forma segura por hilos múltiples. Muestra cómo el acceso sincronizado puede ralentizar la ejecución del programa al detener brevemente la ejecución en cada acceso al contador sincronizado.