Brian Camilo Fajardo Sánchez

**Recuperación Parcial Corte 1**

Para la explicación, se van a detallar paso a paso como se solucionó cada punto. Tener en cuenta que a medida que avanza el parcial se modificaron métodos y los atributos. Por lo que si desde el punto uno se desea validar las instrucciones con el código no se verá acorde a lo explicado ya que en el repositorio se encuentra la versión final.

1. Cree una clase de tipo Thread que represente el ciclo de vida de un hilo que calcule una parte de los dígitos requeridos.

* Para solucionar este punto, inicialmente se creó una clase llamada *PiDigitsThread.java,* que heredaba de la clase *Thread* y se agregó toda la lógica referente al calculo de la fórmula *BBP.*
* En la nueva clase, en vez de tener un método llamado *getDigits*, hay un método llamado *calculate* que ejecuta la misma función.
* Se añadió un atributo de clase llamado *answer* que es un arreglo de bytes destinado a almacenar el resultado del cálculo de la función *calculate*
* Como *PiDigitsThread* es un thread, se sobrescribió el método *run* y en su contenido solo tiene el llamado al método *calculate* que se almacenará en la variable *answer,* el resto de los métodos se dejaron igual*.*
* En la clase *PiDigits* se creó un nuevo método para reemplazar el antiguo getDigits, que inicialmente (antes de la implementación del punto 2 y 3) únicamente contenía un llamado a la clase *PiDigitsThread* y funcionaba como si contara con un solo hilo en la ejecución.

1. Haga que la función *PiDigits.getDigits()* reciba como parámetro adicional un valor N, correspondiente al número de hilos entre los que se va a paralelizar la solución. Haga que dicha función espere hasta que los N hilos terminen de resolver el problema para combinar las respuestas y entonces retornar el resultado. Para esto, puede utilizar el método *Join()* del API de concurrencia de Java.

* Para este punto, primero se añadió el parámetro N al método *getDigits* de la clase *PiDigits*.
* Se creó un arreglo de bytes llamado *answer* que almacenará el resultado de los threads y un arreglo de *PiDigitsThread* que almacenará los threads
* Se creó un método llamado *createThreads* el cual le entra como atributo el arreglo de los threads, el punto de inicio general (es decir desde donde se van a iniciar a calcular los núemros), la cantidad de números que se desea calcular y la cantidad de hilos. Este método inicialmente calcula el módulo de la cantidad de número que se desea calcular y el número de hilos que se especificaron, esto con el fin de saber cuántos números sobran al hacer la división entera y poder asignarlos más adelante. Luego se calcula el rango inicial que será la división entera de la cantidad de números a calcular entre la cantidad de hilos. Se asigna el valor de inicio y se crea un bucle for que iterará por el número de hilos. Seguido se realiza una verificación para saber si ya se va a calcular el rango de último hilo, si sí, al rango se le añade el residuo de la división que almacenamos al inicio del método, si no, continua y agrega un nuevo thread al arreglo de hilos que entró por parámetro. Por último, se le agrega el rango del último hilo al punto de inicio del siguiente hilo. Al salir del bucle, se hace otro bucle inicializando todos los hilos.
* Se creó un método llamado *joinThreads* al cual le entran únicamente el arreglo de hilos y su función es recorrer el arreglo y realizar un *join()* para que todos los hilos termines su ejecución antes de que intente obtener el resultado.
* Para finalizar, se creó un método llamado *getAnswer* que le entra por parámetro el arreglo de hilos y el arreglo vacío de donde se almacenarán las respuestas. Para obtener las respuestas, en la clase *PiDigitsThread* se creó un método llamado *getAnswer.* En este método se hace uso de un bucle para recorrer los hilos, obtener sus respuestas y copiarlas al arreglo

1. Ajuste la implementación para que cada 5 segundos los hilos se detengan e impriman el número de digitos que han procesado y una vez se presione la tecla enter que los hilos continúen su proceso.