Práctica 7. Creación de una aplicación con hilos para resolver problemas.

Competencia a desarrollar

Crea subprogramas para resolver problemas concurrentes utilizando Multihilos.

Introducción

La mejor forma de apreciar la concurrencia es aplicándola en la solución de problemas. Aquí se hace alusión a un tipo de problema sobre datos compartidos y operaciones concurrentes sobre ellos. El caso presentado trata sobre cómo agilizar una manipulación de datos que se pretenden copiar cuando se trabaja con varios hilos. En este caso se manifiestan diversos tipos de problemas, uno es referente al cómo realizar la copia de los datos con varios hilos, otro a cómo mantener la consistencia en las operaciones realizadas y otro que se presenta cuando se requiere obtener algún resultado a través de un método de los datos compartidos cuando los hilos los están utilizando.

Correlación con los temas y aplicación en el contexto

Aquí se ponen en práctica los temas 3.3 y 3.4 en la solución de los casos que se pretenden resolver, donde también se requiere de la comprensión del tema 3.1 y se resalta aún más el tema 3.2 en las ventajas del uso de varios flujos en lugar de solo uno.

El contexto a donde se aplica es en el ambiente de concurrencia a nivel de programación a través de java, utilizando datos en memoria para su acceso y manipulación, los resultados se muestran en consola.

Material y equipo necesario

Equipo de cómputo: Laptop o PC

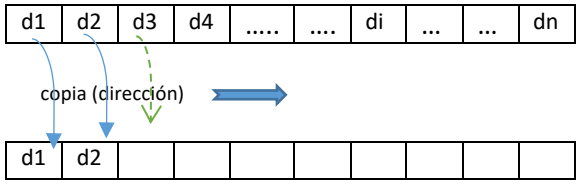
Software: Aquí se emplea el “IDE” NetBeans pero puedes utilizar cualquier otro que maneje java con un entorno de diseño de GUI-Java, versión del jdk 1.7.0 o posterior.

Metodología

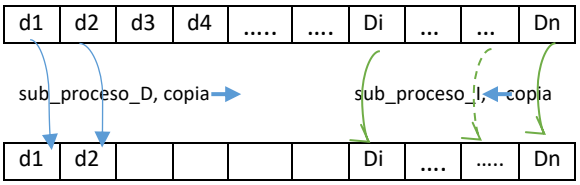
Se presenta la descripción del caso a resolver, se parte de la forma de hacer la copia de datos con un solo hilo y se expone la estrategia a utilizar para aplicar más de un hilo. Luego durante el desarrollo de la práctica se realiza la implementación de la solución, mediante la cual se hace participar al estudiante a través de complementación del código, de análisis de los resultados y del código en sí con preguntas que tiene que responder y anotar en su bitácora de seguimiento. Lo indicado a realizar por estudiante va precedido por el símbolo ´?´ o bien en forma explícita como en el punto número nueve.

<u>Descripción</u>

Esta aplicación surge de una de las tareas más comunes para grandes conjuntos de datos en memoria y persistentes, dicha tarea es “copiar los datos” a fin de no modificar los datos originales en ciertas operaciones, hacer un respaldo o para otros fines. Generalmente la copia se hace en forma lineal, es decir en una sola dirección usando un solo flujo(hilo), como se muestra a continuación:

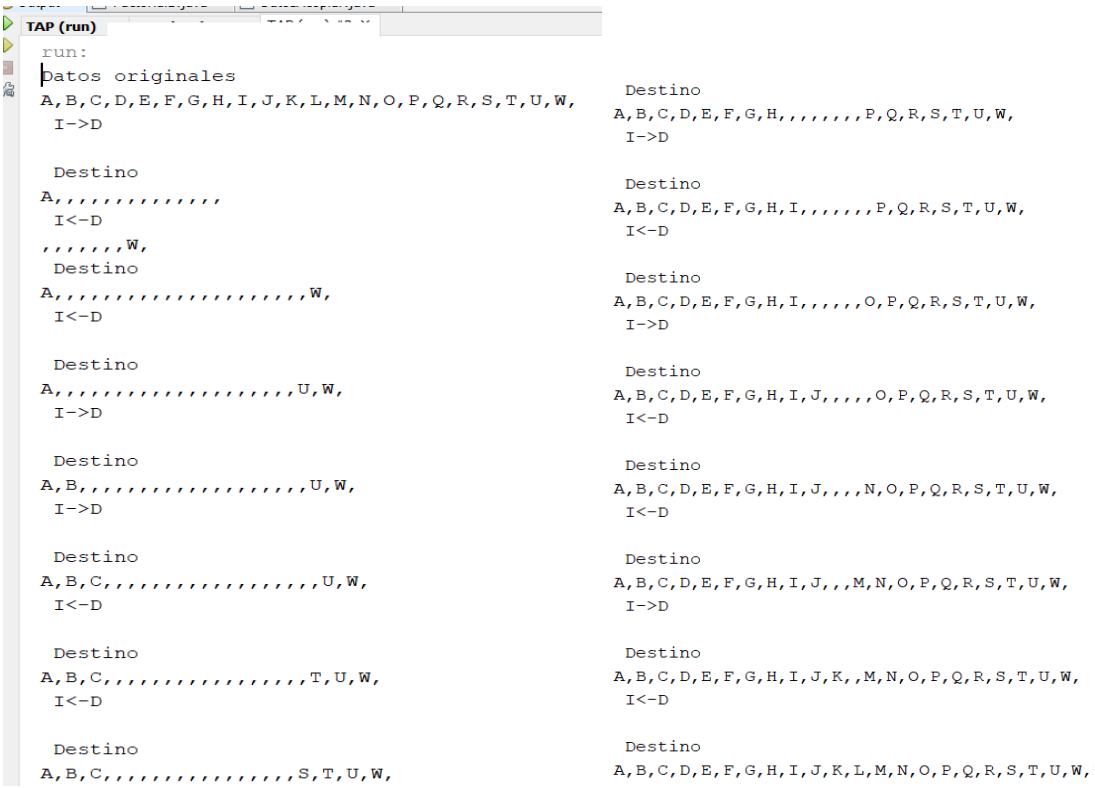


La intención de usar hilos es hacer la copia en ambas direcciones, esto hará repartir la tarea en dos sub-procesos, el sub_proceso_D copiará hacia la derecha, mientras que el sub_proceso_I lo hará hacia la izquierda.



Uno de los problemas a revolver al copiar los datos con estos dos sub-procesos utilizando varios hilos es, determinar en qué momento uno de los procesos debe terminar de copiar, pues ambos realizan la misma tarea con el mismo conjunto de datos pero en sentido inverso. Esto debe soportar el uso de varios hilos (por los menos de dos).

Abajo se muestra el resultado de una ejecución con un conjunto de datos pequeño para que se pueda apreciar y comprender mejor lo que se debe hacer. Nota que aunque aquí se hace una copia en memoria, esto puede ser extendido para el caso de archivos lo cual da pie para un pequeño proyecto.



. . . . . .

Desarrollo

Las clases involucradas son:

DatosAcopiar. Recibe los datos originales y regresa los datos copiados, contiene las operaciones de copia en ambos sentidos y otras necesarias para hacer descriptivo y didáctico su comportamiento.

DatosMovDer. Subproceso que realiza la copia de izquierda a derecha

DatosMovIzq. Subproceso que realiza la copia de derecha a izquierda.

PruebaCopia. Realiza la prueba con un conjunto de datos,

1. Preparativos: Crea un nuevo proyecto de java simple, puedes llamarle practica_07, dentro de él crea un paquete fuente, puedes llamarle 'copiar'. Dentro del paquete 'copiar' crea las clases involucradas con sus nombres arriba propuestos.
2. Clase "DatosCopiar"
   a) Agrega los tributos de la clase DatosCopiar:

```
private char [] datosOrigen;
private char [] datosCopiados;
private int ili; // indice de lectura izquierdo
private int ild; // indice de lectura derecho
private int iei; // indice de escritura izquierdo
private int ied; // indice de escritura derecha
private boolean parar=false;
```

b) ? Crea un constructor public DatosAcopiar(char []datos) que:
   i. Asigne los datos a los datos origen los valores a 'datosOrigen'
   ii. Asigne a los índices de lectura y escritura izquierdos el valor de cero
   iii. Asigna a los índices de lectura y escritura derechos el valor máximo que pueden llegar a tener, lo cual está en función del tamaño de los datos originales a copiar.
   iv. Crea la referencia de 'datosCopiados' del tamaño de los datos originales.

c) Crea el método copiar hacia la izquierda con el siguiente código"

```
public synchronized void copiai()
{//copia de derecha a izquierda
    if(ili+1<=ild && ili<datosOrigen.length && iei+1<=ied && !parar )
{
    datosCopiados[iei++]= datosOrigen[ili++];
}
  else
{  datosCopiados[iei++]= datosOrigen[ili++];
    parar=true;
}
}
```

d) Te corresponde a ti crear el método copiar a la derecha 'void copiad()',para esto tienes que analizar el método copiari(), solo que ahora el sentido es inverso.

e) Crea el método que imprime como va formándose la copia (destino)

```
public synchronized void imprimir()
{  System.out.println("\n Destino");
   for(int i=0; i< datosOrigen.length; i++)
     System.out.print(datosCopiados[i] +",");
}
```

f) ? Te corresponde crear el método para imprimir los datos originales

g) Los métodos que regresan el valor de parar() y de los datos copiados son triviales como puedes observar:

```
public boolean parar ()
{
    return parar;
}
public char [] datosCopiados()
{
   return datosCopiados;
}
```

3. Clase 'DatosMovDer' que realiza la tarea de copiar hacia la derecha

a) Has que la clase 'DatosMovDer', previamente creada implemente a 'Runnable'

b) Crea el constructor

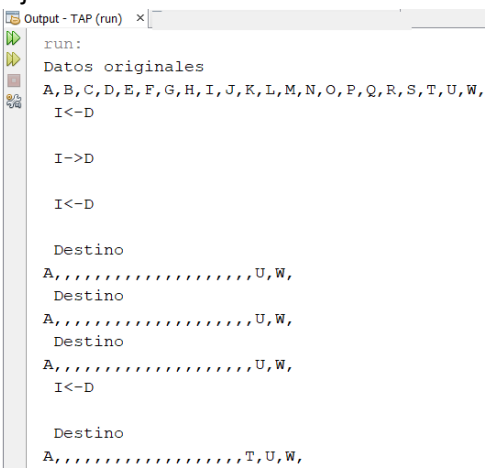
```
public DatosMovDer(DatosAcopiar d)
{
  datos=d;
}
```

c) Implemente el método 'runnable()' con el código

```
public void run() {
      while( ?? ){
        datos.copiad();
        System.out.println("\n I<-D");
        datos.imprimir();
    try {
        Thread.sleep(r.nextInt(500));
    } catch (InterruptedException ex) {
        Logger.getLogger(DatosMovIzq.class.getName()).log(Level.SEVERE, null, ex);
    }
      }
    }
```

d) ? Te corresponde escribir la condición que va dentro del 'while()'

4. Clase 'DatosMovIzq' que realiza la tarea de copiar hacia la izquierda.
   ? Te corresponde crear su código completo, tomando como referencia la clase previamente creada 'DatosMovDer'
5. Clase prueba. Dentro del método 'main', crea el siguiente código
   a) Crea un arreglo de caracteres: ''char datos[]" y has que contenga las letras de la 'A' a la "W", semejante a la muestra.
   b) Crea un objeto de la clase 'DatosAcopiar', pasándole a su constructor el arreglo de datos
   c) Ejecuta el método que imprime los datos originales del objeto creado de 'DatosAcopiar'
   d) Crea un objeto de la clase 'DatosMovIzq' y otro de la 'DatosMovDer'
   e) Crea dos hilos, uno que use un proceso de DatosMovIzq y el otro que use
6. Realiza varias ejecuciones de prueba y graba las imágenes de salida
7. Agrega más hilos que incluyan ya sea un proceso DatosMovIzq o bien de DatosMovDer
   Vuelve hacer otras ejecuciones de prueba y graba las imágenes de salida, las cuales deben ser semejantes a:



   ¿En dónde se nota que hay más hilos y de qué tipo?
8. Has que en la ejecución de prueba el objeto de la clase 'DatosAcopiar' ejecute el método imprimir(), directamente, es decir en forma explícita, agrega el código al final.
   ¿Qué sucede, es decir qué se muestra en la ejecución?
   Has que cada hilo ejecute su método join().

Vuelve a ejecutar otra prueba y ve lo que ocurre ahora

9. Contesta las siguientes preguntas y anótalas en tu bitácora de seguimiento:
   a) ¿Cuáles son las similitudes y las diferencias entre este caso y el de productor consumidor visto en la práctica seis?
   b) Transcribe la parte del código que crees que hace posible un balance de asignación de tiempo para cada hilo, ¿por qué crees que se logra con ello?
   c) ¿Qué pasa cuando aumentas más hilos?, ¿Sigue funcionando bien el código, por qué si o no?
   d) ¿Por qué se requiere usar la cláusula 'synchronized' en el método imprimir?
   e) Realiza una ejecución manual que indique cómo funciona el código.
   f) Realiza un esquema de cómo funciona con el ´join´

Sugerencias didácticas

Documenta tu código.

Trabajo en equipo de 2 a 4 estudiantes

Uso de una bitácora de seguimiento

Reporte del estudiante

Código fuente de las clases participantes

Imágenes de salida de las pruebas realizadas

Bitácora de seguimiento

Bibliografía

1. Deitel y Deitel. Como programar en Java. Prentice Hall.Septima Edición, 2008
2. Oracle Documentación, The Java™ Tutorials. https://docs.oracle.com/javase/tutorial/uiswing/components/index.html. Último acceso 21/11/18