Desarrollo de software

Programa de la Unidad didáctica:
Programación orientada a objetos III

Unidad 2. Hilos

Clave:
15142423/ 16142524

Ciudad de México, julio del 2025

Universidad Abierta y a Distancia de México

Índice

Presentación de la unidad

Existen ocasiones en las que la ejecución de un programa requiere de procesamiento paralelo, es decir que el programa realice diferentes tareas al mismo tiempo, para ello existen hilos de ejecución, los cuales permiten ese procesamiento multitarea (o multi-hilo) del que hablamos.

En esta segunda unidad de la materia Programación orientada a objetos III (POO3), revisarás la diferencia entre programas con flujo único y múltiple, así como el uso de hilos para crear programas con múltiples flujos de procesamiento. Estos temas serán tratados para adentrarte en la programación multi-hilos y con ello logres crear programas con funciones que se ejecuten de forma paralela.

Logros de la unidad

En esta unidad lograrás:

- Distinguir el flujo de ejecución de un programa
- Diferenciar el flujo único del flujo múltiple de ejecución de un programa
- Crear y manipular hilos

Competencia específica

- Crear programas mediante la utilización de hilos para el manejo de flujos múltiples de información.

Temario de la unidad

2. Hilos
2.1. Programas con flujo único
    2.1.1. Flujo normal
    2.1.2. Ejemplo de un programa con flujo único
2.2. Programas de flujo múltiple
    2.2.1. Creación de hilos
    2.2.2. Estados de hilos
    2.2.3. Control de hilos

2. Hilos

Los hilos en programación, básicamente, son procesos de ejecución, de tal manera que si se programan varios hilos en una sola aplicación; ésta será capaz de realizar varias tareas de manera paralela, por lo que este tema resulta de sobremanera útil cuando se deban realizar programas con una gran carga de procesamiento de información. A lo largo de esta unidad siguiendo las lecturas y ejercicios propuestos analizarás el manejo de hilos en Java.

Por lo anterior, es importarte que atiendas a las indicaciones posteriores, pues, tendrás que revisar textos, ejecutar tareas y, al final, realizar tu evidencia de aprendizaje, misma que dará cuenta de la obtención de la competencia de la unidad. Dicho lo anterior, entrarás en materia con el tema *2.1. Programas con flujo único*.

2.1. Programas con flujo único

Todos los programas, que hasta el momento se han realizado a lo largo de las materias de POO, han sido de flujo único, pues las instrucciones de éstos se van ejecutando conforme se van llamando; estas instrucciones pueden estar en diferentes métodos, o clases, pero sólo se van ejecutando de manera única conforme son llamadas.

“Un programa de flujo único, tarea única o mono-hilo utiliza un único flujo de control para controlar su ejecución. Muchos programas no necesitan la potencia o utilidad de múltiples tareas. Sin necesidad de especificar explícitamente que se quiere un único flujo de control.” (Froufe, 2009, pág. 203)

Para conocer más a fondo sobre programas de flujo múltiple de ejecución de un programa **revisa**[1] los siguientes textos:

- En *Java 2 Manual de usuario y tutorial* de (Froufe 2009, p. 202), podrás encontrar una descripción de lo que son los programas de flujo único, así como un pequeño código que ejemplifica este tipo de programas.
- En *Java 2* de (Sánchez 2004, p.185) se presenta una introducción al tema de hilos, donde podrás encontrar una descripción sobre el manejo de programas con un solo hilo, para que lo distingas de programas con la aplicación de varios hilos.
- En *El Lenguaje de Programación Java TM* (2011) encontrarás una descripción sobre lo que son los hilos, en él se menciona que un hilo es un flujo simple de ejecución dentro de un programa, de esta misma manera encontrarás un ejemplo de un programa con un solo hilo, para que lo comprendas mejor.

[1] Los textos los encontrarás en la sección **descargables** de la unidad.

Como ya se mencionó, todos los programas realizados anteriormente caen en esta clasificación de flujo único de ejecución. A continuación, se tiene un ejemplo de programa con flujo único. Este ejemplo realiza la impresión de números del 0 al 99 mediante un *for*, la interfaz gráfica sugerida se puede ver en la Figura. Programa de flujo único con impresión de números.

Ejemplo de código 1. Impresión de números en programa de flujo único.

```
private void btnEjecutarActionPerformed(java.awt.event.ActionEvent evt) {
        // TODO add your handling code here:
        String texto="";

        for(int i=0; i<100; i++){
            texto+=i+"\n";
            txaImpresion.setText(texto);
        }

    }
```

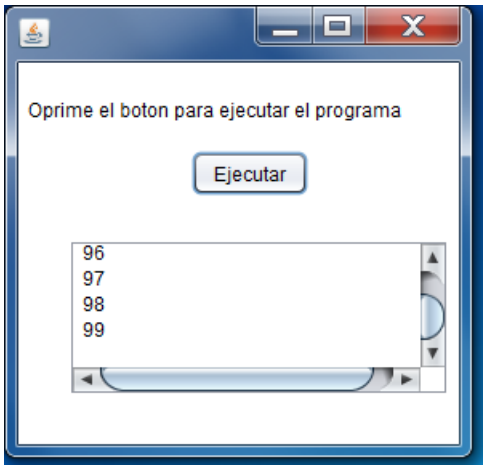

Figura. Programa de flujo único con impresión de números.

* Nota: Se presenta sólo el método que realiza la impresión de los números, la interfaz gráfica que manejará la información será omitida, pues el código para realizarlo ya se revisó en POO2.

En este tema lo importante es que comprendas que: todos los programas realizados con anterioridad manejan un flujo único de ejecución, para ello puedes consultar cualquiera de los códigos proporcionados en las tres entregas de la materia de POO.

2.2. Programas de flujo múltiple

El flujo múltiple de ejecución de programas permite que se estén realizando diferentes tareas en un programa al mismo tiempo; es decir, que las tareas se ejecuten de forma paralela, para ello Java utiliza unos elementos llamados hilos (*thread*).

Cada hilo realiza una tarea en específico, al tener varios hilos ejecutándose, se tendrán varias tareas corriendo en el mismo programa. Lo que permitirá que en el mismo programa se estén realizando diferentes actividades al mismo tiempo. Por ejemplo, si se tiene un sistema en red dentro de una empresa y se requiere que, al realizar una venta, se actualice la cuenta del cliente, las ventas del vendedor y el inventario un programa mono-hilo se deberá ir realizando cada tarea una a una, de manera que hasta que se realice un registro podrá procederse a realizar otro, a diferencia de un programa multi-hilo, donde podrán registrarse todas las tareas al mismo tiempo.

Mientras que los programas de flujo único pueden realizar su tarea ejecutando las sub-tareas secuencialmente, un programa multitarea permite que cada tarea comience y termine tan pronto como sea posible. Este comportamiento presenta una mejor respuesta a la entrada en tiempo real. Para enriquecer la información **consulta** en el *Java 2 Manual de usuario y tutorial* de (Froufe, 2009, p. 215); en que se muestran los diferentes estados en los que puede estar una tarea (hilo), así como la comunicación que se puede generar entre diferentes hilos para lograr una aplicación más robusta, en este texto también encontrarás bloques de código aplicando los conceptos que se presentan para que comprendas como utilizarlos en un programa.

Los hilos de ejecución en Java están implementados en la clase *thread*, que forma parte del paquete java.lang. Cada hilo (*thread*) tiene un principio, un flujo de ejecución y un fin definidos, pero no es una entidad independiente sino que debe ejecutarse en el contexto de un programa (Joyanes, 2002). Por lo tanto, **revisa** el *tema 12.5. Hilos de tipo demonio* en Joyanes (2002, p. 317), en dicho texto encontrarás como crear y manejar hilos demonio, los cuales permiten ofrecer servicios a otros hilos de ejecución existentes dentro del mismo proceso, la obra te ofrece un pequeño ejemplo para que aprendas a crear hilos demonio y con ello logres desarrollar programas con hilos de este tipo.

Para conocer más a fondo el manejo de hilos y las clases y métodos que ayudan con su manipulación y utilización, **abre** el texto Sanchez, J. (2004, p. 185) y consulta una breve introducción al tema de hilos, el cual también se conoce como multitarea, que es la posibilidad de que una computadora realice varias tareas a la vez. Esa introducción te ayudará a comprender mejor el concepto de hilo.

Complementa tu información ingresando a la siguiente liga: http://docs.oracle.com/javase/tutorial/essential/concurrency/index.html de Oracle (2022), donde encontrarás el tema de concurrencia que abarca el manejo de hilos, así como la especificación de la clase *thread* y su uso. En ese apartado del tutorial de Java, podrás encontrar todo el aspecto técnico con respecto al manejo de hilos, sólo deberás moverte en el menú para revisar temas más avanzados.

Ahora, **revisa** el *Tema 9.2 Extensión de la clase thread de Programador Certificado Java 2*, en Martin, (2010, p. 397) donde podrás ver cómo crear un hilo, así como los métodos más utilizados para manejarlos, estudiando este texto y analizando el código que presentan lograrás tener una visión más concreta sobre cómo programar con hilos; se recomienda que crees los programas con los bloques de código presentados en esta obra para que vayas comprendiendo y practicando al mismo tiempo.

Para finalizar y después de que revisaste los textos sugeridos sobre el tema de hilos, se presenta un ejemplo de cómo crear y manipular hilos en un programa Java, es recomendable que realices tu propio programa basándote en la sintaxis que se muestra en el ejemplo, todo ello para que analices a profundidad el código y logres aplicarlo en las actividades que realizarás para este tema.

El código está comentado, sigue los comentarios cuidadosamente para que comprendas el código presentado.

Ejemplo de código 1. Clase para el uso de hilos

```
1  /**
2   *       Esta clase representa un hilo.
3   *       Hereda de la clase Thread y sobreescribe el
4   *       método run.
5   */
6  public class HiloMensaje extends Thread
7  {
8           //Esta variable sirve para dar nombre al hilo y
9           //se imprimirá cada vez que el hilo esté en ejecución.
10          private String nombreHilo;
11
```

```
12          //Esta variable sirve para detener la ejecución
13          //en cualquier momento que el usuario desee.
14          private boolean ejecutando;
15
16          /**
17           *      Constructor vacío para la clase HiloMensaje.
18           *      Asigna por default el nombre "Hilo sin nombre".
19           */
20          public HiloMensaje()
21          {
22                  this ("Hilo sin nombre");
23          }
24
25          /**
26           *      Constructor para la clase HiloMensaje.
27           *      Recibe como parámetro el nombre que recibirá
28           *      el hilo.
29           */
30          public HiloMensaje(String nombre)
31                  {
                            nombreHilo = nombre;
32                          ejecutando = false;
                    }
33
34                  /**
                     *      Devuelve el valor de la variable
35                   *      nombreHilo.
                     */
36                  public String getNombre()
                    {
```

```
37                  return nombreHilo;
            }
38
            /**
39           *      Este método se manda llamar de forma automática cuando se
             *      inicia la ejecución del hilo. En otras palabras, en este método
40           *      se deben escribir las instrucciones que se desean ejecutar
             *      como un hilo paralelo.
41           */
            @Override
42          public void run()
            {
43                  //Esta variable servirá para llevar un conteo de las veces que se itera
                    //el ciclo while más abajo.
44                  int i = 0;

45                  //Se establece la variable "ejecutando" en true para indicar que el hilo
                    //esta en ejecución.
46                  ejecutando = true;

47                  //Mientras la variable "ejecutando" tenga el valor verdadero,
                    //se imprimirá un mensaje mostrando el nombre del hilo y el
48  número de veces
                    //que se ha iterado en el ciclo.
49                  while (ejecutando)
                        System.out.println("Hola, soy el hilo " + nombreHilo + " en
50  iteración " + (++i));
            }

51          /**
             *      Sirve para saber si actualmente el hilo está en ejecución.
52           */
            public boolean estaEjecutando()
53          {
                    return ejecutando;
54          }

55          /**
             *      Sirve para forzar un hilo a detenerse.
56           */
            public void detener()
57          {
                    ejecutando= true;
58
59
60
61
```

```
62                  }
            }
63
64
65
66
67
68
69
70
71
72
73
74
75
76
77
78
79
80
81
82
83
84
```

Ejemplo de código 2. Clase principal

```
1  /**
2   *       Clase principal del programa.
3   *
4   */
5  public class Main
6  {
7          /**
8           *       Este método inicia la ejecución del programa.
9           *
10          *       Crea dos objetos de tipo HiloMensaje (hilos) y los inicia.
11          */
12         public static void main(String[] args) throws Exception
13         {
14                 // Objeto que representa el primer hilo.
15                 HiloMensaje hilo1 = new HiloMensaje("Hilo 1");
16
17                 // Objeto que representa el segundo hilo.
18                 HiloMensaje hilo2 = new HiloMensaje("Hilo 2");
19
20                 hilo1.start(); //Se inicia la ejecución del primer hilo.
21                 hilo2.start(); //Se inicia la ejecución del segundo hilo.
22
23                 //Esta instrucción detiene el hilo principal del programa
24                 //por dos segundos. Sin embargo, los dos hilos anteriores
```

```
25                  //continúan su ejecución.
26                  Thread.currentThread().sleep(2000);
27
28                  System.exit(0);
29              }
30          }
```

Cierre de la unidad

Has concluido la segunda unidad, y a lo largo de ella te has introducido en los conceptos de flujo único y múltiple de ejecución de tal manera que has distinguido el uso de hilos para generar programas que manejen varios procesos de ejecución de forma paralela, también has estudiado las clases que Java proporciona para ese manejo de flujos múltiples mediante hilos. Por último, has estudiado la manera de lograr la manipulación de hilos de ejecución. En la siguiente unidad aprenderás a realizar programas que trabajen en red (es decir, en más de una computadora), estos programas pueden requerir de utilizar diferentes hilos de ejecución en su codificación.

Es aconsejable que revises nuevamente si los temas mencionados no te son familiares. De no ser este tu caso, ya estás preparado (a) para seguir con la unidad tres, en donde continuarás con la revisión del uso de *sockets*.

Para saber más…

Para que puedas ejecutar los programas que se te presentan, así como las actividades es importante que instales un IDE en tu computadora, se recomienda NetBeans, puedes descargarlo de forma gratuita de la siguiente liga: http://netbeans.org/downloads/

Es recomendable que pruebes los códigos que se presentan en los ejemplos que se encuentran en cada fuente de consulta mencionada.

Fuentes de consulta

Bibliografía Básica

Froufe, A. (2009). *Java 2: Manual de usuario y tutorial*. Alfaomega.

Joyanes, L. (2002). *Java 2: Manual de programación*. McGraw-Hill.

Martín, A. (2010). *Programador certificado Java 2: Curso práctico*. Alfaomega.

Oracle Corporation. (2022). *The Java tutorials*. Oracle. https://docs.oracle.com/javase/tutorial/

Sánchez, J. (2004). *Java 2*. [Editorial no especificada].

Solano, J. A. (2020). *Hilos*. Unidades de Apoyo para el Aprendizaje, CUAIEED, Facultad de Ingeniería, Universidad Nacional Autónoma de México. https://uapa.cuaed.unam.mx/node/1077