ESCUELA DE
INGENIERÍA EN CIENCIAS Y SISTEMAS
FACULTAD DE INGENIERÍA
UNIVERSIDAD DE SAN CARLOS DE GUATEMALA

ECYS | DTT
DESARROLLO DE TRANSFERENCIA TECNOLÓGICA

Día, Fecha:	Miércoles, 18/03/2023
Hora de inicio:	19:00

Introducción a la programación y computación 1 [D]

José Eduardo Morales García

Manejo de hilos

- El término "manejo de hijo" en programación se refiere a la capacidad de un proceso o programa para crear y controlar otros procesos o programas secundarios, conocidos como "hijos".

- En los sistemas operativos modernos, un proceso padre puede crear uno o varios procesos hijos, que se ejecutan como subprocesos separados y pueden realizar tareas específicas en paralelo con el proceso padre. El proceso padre puede comunicarse con sus procesos hijos a través de señales, intercambio de mensajes, variables compartidas y otros mecanismos de sincronización.

Hilos en Java

- Los hilos en Java son subprocesos de ejecución que permiten que un programa realice varias tareas al mismo tiempo. Los hilos son una forma de programación concurrente en la que se pueden ejecutar varias tareas simultáneamente en un solo proceso.

- Java es un lenguaje de programación que tiene soporte nativo para hilos. La clase Thread es la base para trabajar con hilos en Java. La creación de hilos en Java se puede hacer de dos maneras

Creación de hilos en Java

Extender la clase Thread

- Se puede crear una clase que extienda la clase Thread, y sobrescribir el método run() para indicar las tareas que se ejecutarán en el hilo. Luego, se puede crear una instancia de la clase creada y llamar al método start() para iniciar el hilo.

```java
public class MiHilo extends Thread {
    public void run() {
        // Código que se ejecutará en el hilo
        System.out.println("Hola desde el hilo!");
    }

    public static void main(String[] args) {
        // Crear una instancia del hilo
        MiHilo hilo = new MiHilo();

        // Iniciar el hilo
        hilo.start();

        // Código que se ejecutará en el hilo principal
        System.out.println("Hola desde el hilo principal!");
    }
}
```

Creación de hilos en Java

Implementar la interfaz Runnable

- Se puede crear una clase que implemente la interfaz Runnable, y sobrescribir el método run() para indicar las tareas que se ejecutarán en el hilo. Luego, se puede crear una instancia de la clase Thread, pasando la instancia de Runnable como argumento al constructor, y llamar al método start() para iniciar el hilo.

```java
public class MiHilo implements Runnable {
    public void run() {
        // Código que se ejecutará en el hilo
        System.out.println("Hola desde el hilo!");
    }

    public static void main(String[] args) {
        // Crear una instancia del objeto que implementa Runnable
        MiHilo miHilo = new MiHilo();

        // Crear una instancia del hilo y pasar el objeto Runnable como argumento
        Thread hilo = new Thread(miHilo);

        // Iniciar el hilo
        hilo.start();

        // Código que se ejecutará en el hilo principal
        System.out.println("Hola desde el hilo principal!");
    }
}
```

Métodos de los hilos

start()

- este método inicia la ejecución del hilo. Cuando se llama a este método, el hilo comienza a ejecutar su método run().

run()

- este método contiene el código que se ejecutará en el hilo. Este método se sobrescribe en una clase que extiende la clase Thread o se implementa en una clase que implementa la interfaz Runnable.

```java
public class MiHilo extends Thread {
    public void run() {
        System.out.println("Hola desde el hilo!");
    }

    public static void main(String[] args) {
        MiHilo miHilo = new MiHilo();
        miHilo.start();
        System.out.println("Hola desde el hilo principal!");
    }
}
```

```
Hola desde el hilo principal!
Hola desde el hilo!
```

Métodos de los hilos

join()

- Este método hace que el hilo que lo llama espere hasta que el hilo en el que se llama termine su ejecución.

```java
public class MiHilo extends Thread {
    public void run() {
        try {
            Thread.sleep(5000); // el hilo se suspende durante 5 segundos
            System.out.println("Hola desde el hilo!");
        } catch (InterruptedException e) {
            e.printStackTrace();
        }
    }

    public static void main(String[] args) {
        MiHilo miHilo = new MiHilo();
        miHilo.start();
        System.out.println("Hola desde el hilo principal!");
        try {
            miHilo.join(); // el hilo principal espera a que el hilo secundario ter
        } catch (InterruptedException e) {
            e.printStackTrace();
        }
        System.out.println("Fin del programa.");
    }
}
```

```
Hola desde el hilo principal!
Hola desde el hilo!
Fin del programa.
```

Métodos de los hilos

sleep()

- Este método hace que el hilo que lo llama se suspenda durante un período de tiempo determinado, especificado en milisegundos.

```
class MiHilo extends Thread {
   public void run() {
      try {
         System.out.println("El hilo se está ejecutando");
         Thread.sleep(5000);
         System.out.println("El hilo ha terminado de ejecutarse");
      } catch (InterruptedException e) {
         System.out.println("El hilo ha sido interrumpido");
      }
   }
}

public class Ejemplo {
   public static void main(String[] args) {
      MiHilo hilo = new MiHilo();
      hilo.start();
   }
}
```

Métodos de los hilos

yield()

- este método hace que el hilo que lo llama se suspenda temporalmente para permitir que otros hilos en el sistema tengan la oportunidad de ejecutarse.

```java
class MiHilo extends Thread {
    public void run() {
        for (int i = 0; i < 5; i++) {
            System.out.println("El hilo está en ejecución");
            Thread.yield();
        }
    }
}

public class Ejemplo {
    public static void main(String[] args) {
        MiHilo hilo = new MiHilo();
        hilo.start();
        for (int i = 0; i < 5; i++) {
            System.out.println("El hilo principal está en ejecución");
        }
    }
}
```

Métodos de los hilos

isAlive()

- Este método devuelve true si el hilo en el que se llama aún está en ejecución.

```java
public class MiHilo implements Runnable {
    private Thread hilo;

    public void run() {
        // código a ejecutar en el hilo
        // ...
    }

    public void iniciarHilo() {
        if (hilo == null) {
            hilo = new Thread(this);
            hilo.start();
        }
    }

    public boolean hiloEstaActivo() {
        if (hilo != null) {
            return hilo.isAlive();
        }
        return false;
    }
}

public class Ejemplo {
    public static void main(String[] args) {
        MiHilo miHilo = new MiHilo();
        miHilo.iniciarHilo();

        if (miHilo.hiloEstaActivo()) {
            System.out.println("El hilo está en ejecución");
        } else {
            System.out.println("El hilo no está en ejecución");
        }
    }
}
```

Métodos de los hilos

interrupt()

- Este método interrumpe el hilo en el que se llama, lo que puede hacer que se produzca una excepción.

```java
class MiHilo extends Thread {
   public void run() {
      try {
         System.out.println("El hilo se está ejecutando");
         Thread.sleep(5000);
      } catch (InterruptedException e) {
         System.out.println("El hilo ha sido interrumpido");
         return;
      }
      System.out.println("El hilo ha terminado de ejecutarse");
   }
}

public class Ejemplo {
   public static void main(String[] args) {
      MiHilo hilo = new MiHilo();
      hilo.start();
      try {
         Thread.sleep(2000);
      } catch (InterruptedException e) {
         System.out.println("El hilo principal ha sido interrumpido");
      }
      hilo.interrupt();
   }
}
```

Métodos de los hilos

wait()

- Es utilizado en Java para hacer que un hilo espere hasta que se cumpla alguna condición

```java
public class MiHilo implements Runnable {
    private boolean bandera = false;

    public void run() {
        synchronized(this) {
            while (!bandera) {
                try {
                    wait();
                } catch (InterruptedException e) {
                    // manejo de la excepción
                }
            }
            // código a ejecutar después de que la bandera sea verdadera
            System.out.println("La bandera es verdadera");
        }
    }

    public void cambiarBandera(boolean nuevaBandera) {
        synchronized(this) {
            bandera = nuevaBandera;
            notifyAll();
        }
    }
}
```

```java
public class Ejemplo {
    public static void main(String[] args) {
        MiHilo miHilo = new MiHilo();
        Thread hilo = new Thread(miHilo);
        hilo.start();

        // esperar unos segundos antes de cambiar la bandera
        try {
            Thread.sleep(2000);
        } catch (InterruptedException e) {
            // manejo de la excepción
        }

        // cambiar la bandera para que el hilo continúe su ejecución
        miHilo.cambiarBandera(true);
    }
}
```

Métodos de los hilos

notify()

- Este método notifica a otro hilo que se ha producido un cambio de estado y que puede continuar la ejecución.

notifyAll()

- Este método notifica a todos los hilos en espera que se ha producido un cambio de estado y que pueden continuar la ejecución.

```java
public class MiHilo implements Runnable {
    private boolean bandera = false;

    public void run() {
        synchronized(this) {
            while (!bandera) {
                try {
                    wait();
                } catch (InterruptedException e) {
                    // manejo de la excepción
                }
            }
            // código a ejecutar después de que la bandera sea verdadera
            System.out.println("La bandera es verdadera");
        }
    }

    public void cambiarBandera(boolean nuevaBandera) {
        synchronized(this) {
            bandera = nuevaBandera;
            notifyAll();
        }
    }
}
```

Métodos de los hilos

notify()

- Este método notifica a otro hilo que se ha producido un cambio de estado y que puede continuar la ejecución.

notifyAll()

- Este método notifica a todos los hilos en espera que se ha producido un cambio de estado y que pueden continuar la ejecución.

```java
public class Ejemplo {
    public static void main(String[] args) {
        MiHilo miHilo = new MiHilo();
        Thread hilo = new Thread(miHilo);
        hilo.start();

        // esperar unos segundos antes de cambiar la bandera
        try {
            Thread.sleep(2000);
        } catch (InterruptedException e) {
            // manejo de la excepción
        }

        // cambiar la bandera para que el hilo continúe su ejecución
        miHilo.cambiarBandera(true);
    }
}
```

Parte práctica