Threads – Java Concurrency Cookbook

Programación Concurrente 2017

Ing. Ventre, Luis O.

Interrupting A Thread

- Un programa JAVA, con mas de un hilo en ejecución solo finaliza, cuando todos sus hilos (no daemons) terminan su ejecución.
- Puede ser necesario finalizar la ejecución de un Hilo. Por ej. Si queremos terminar un programa o finalizar la tarea que el hilo esta haciendo.
- Java provee un mecanismo para indicarle al hilo que queremos detenerlo.
- Una peculiaridad del mecanismo, es que el hilo puede ignorar la petición y continuar trabajando.

Interrupting A Thread

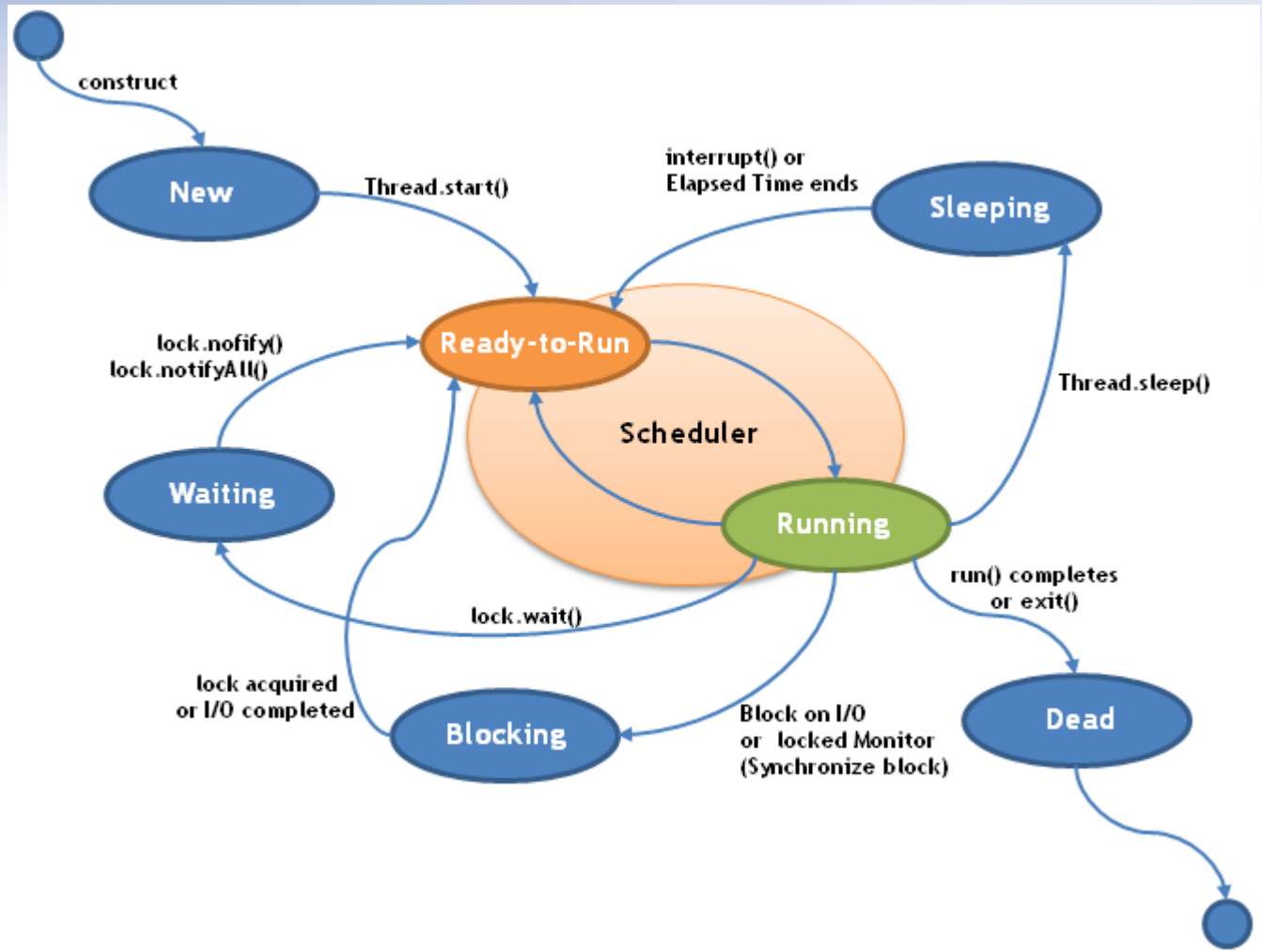
- Se realizara un programa que crea un hilo y 10 segundos después lo finaliza usando el mecanismo de interrupción.

- Clase Main

Main.java | PrimeGenerator.java

```
1  package Pkt;
2
3  public class Main {
4
5      public static void main(String[] args) {
6          Thread task = new PrimeGenerator();
7          task.start();
8
9          try {
10             Thread.sleep(10000);
11         }
12         catch (InterruptedException e) {
13             e.printStackTrace();
14         }
15
16         task.interrupt();
17     }
18 }
19
```

Interrupting A Thread



Interrupting A Thread

- Cree una clase que extiende la clase Thread.

```
public class PrimeGenerator extends Thread {
```

- Sobrescriba el método run, incluyendo un loop que correrá indefinidamente.
  - En este loop se procesaran números comenzando en 1.
  - Para cada numero, se determinara si es primo o no.
  - Si es primo se imprimirá en la consola.
  - Luego de procesar el numero se chequeará si el hilo ha sido interrumpido a través de isInterrupted().
  - El método isPrime determinara si es o no primo.

Clase PrimeGenerator

```
package Pkt;

public class PrimeGenerator extends Thread {

    @Override
    public void run() {

        long number =1L;
        while(true){

            if(isPrime(number)){
                System.out.println("Number" + number + " is Prime");
            }

            if (isInterrupted()){
                System.out.printf("The prime generator has been interrupted\n");
                return;
            }
            number++;
        }
    }
     private boolean isPrime(long number) {
         if(number<=2)
             return true;

         for(long i=2;i<number;i++)
         {
             if((number%i)==0)
                 return false;
         }
         return true;
     }
}
```

Interrupting A Thread

- Ejecute el programa y observe los resultados.
- La salida por consola debería ser similar a (dependiendo del tiempo de sleep)
-

```
Number148199 is Prime
Number148201 is Prime
Number148207 is Prime
Number148229 is Prime
Number148243 is Prime
Number148249 is Prime
Number148279 is Prime
Number148301 is Prime
Number148303 is Prime
Number148331 is Prime
Number148339 is Prime
The prime generator has been interrupted
```

Interrupting A Thread

- El mecanismo visto para la interrupción de un hilo es adecuado para programas simples. Sin recursividad ni complejidad en cantidad de métodos.

- Java provee la interrupción por excepción InterruptedException, esta interrupción puede ser lanzada al detectar la interrupción del hilo y hacer el catch desde el método run().

Interrupting A Thread

- El siguiente ejercicio implementará un hilo que busca archivos con un determinado nombre en un directorio y sus subdirectorios.

- Crear una clase llamada FileSearch que implemente la interfaz runnable.
- Declarar dos atributos privados, uno para el nombre del archivo y el otro para el directorio donde comenzar a buscar.

Interrupting A Thread

- Implemente el método run(), la cual chequea si el atributo fileName es un directorio, y si lo es llama al método directoryProcess().
  - Este método, puede lanzar una excepción InterruptedException por lo que debemos implementar catch.

- Implementar el método directoryProcess(). En cada llamado al método hará un recursión llamando al método fileProcess().

Interrupting A Thread

- Luego de procesar todos los archivos y directorios el hilo chequea si ha sido interrumpido y lanza la excepción correspondiente.

- Implementar el método fileProcess, este método compara el archivo buscado con cada uno, si son iguales, imprime un mensaje en la consola. Luego el hilo chequea si ha sido interrumpido y lanza la excepción correspondiente.

Interrupting A Thread

- Implemente la clase Main.
- Cree e inicialice un objeto de la clase FileSearch.

- Comience la ejecución del hilo.

- Espere 10 segundos

- Interrumpa el hilo.
- Ejecute y vea el resultado.

Interrupting A Thread

- Clase Main

*Main.java | FileSearch.java

```
1  package Pkt;
2
3  import java.util.concurrent.TimeUnit;
4
5  public class Main {
6
7      public static void main(String[] args) {
8
9          FileSearch buscador = new FileSearch("C:\\","log.txt");
10         Thread hilo = new Thread(buscador);
11
12         hilo.start();//si pongo run, el hilo nunca se detiene porque?
13
14         try{
15             TimeUnit.SECONDS.sleep(10);
16         }catch (InterruptedException e){
17             e.printStackTrace();
18         }
19
20         hilo.interrupt();
21     }
22
23 }
```

Interrupting A Thread

- Clase FileSearch

```
import java.io.File;

public class FileSearch implements Runnable {

    private String initPath;
    private String fileName;


    public FileSearch(String initPath, String fileName) {
        super();
        this.initPath = initPath;
        this.fileName = fileName;
    }

    public void run() {
        File file=new File(initPath);
        if(file.isDirectory()){
            try{
                directoryProcess(file);
            }catch (InterruptedException e){
                System.out.printf("%s: La busqueda ha sido interrumpida ",
                        Thread.currentThread().getName());

            }
```

Interrupting A Thread

- Método directoryProcess

```
private void directoryProcess(File file) throws InterruptedException {

    File list[]=file.listFiles();
    if(list != null){
        for(int i=0;i<list.length;i++){
            if(list[i].isDirectory()){
                directoryProcess(list[i]);
            }else{
                fileProcess(list[i]);
            }

        }

    }
    if(Thread.interrupted()){
        throw new InterruptedException();
    }
}
```

Interrupting A Thread

- Método fileProcess

```
private void fileProcess(File file) throws InterruptedException {

    if (file.getName().equals(fileName)) {
        System.out.printf("%s : %s\n", Thread.currentThread().getName(),
                file.getAbsolutePath());
    }

    if(Thread.interrupted())
    {throw new InterruptedException();
    }
}
```

Sleeping & Resuming a Thread

- En diferentes ocasiones es necesario interrumpir la ejecución de un hilo por un determinado tiempo.
- Por Ej. Un hilo que lee un sensor, y luego durante un minuto no realiza tarea alguna.
- Puede utilizar el metodo sleep() de la clase Thread para este propósito.

Sleeping & Resuming a Thread

- Este método recibe un entero como argumento, que representa el número en milisegundos que el hilo suspende su ejecución.

- Cuando el tiempo de sleep finaliza, el hilo continúa con la ejecución de la instruccion siguiente al sleep(). Cuando la JVM le asigne cpu time

- Otra posibilidad es utilizar sleep() de un elemento TimeUnit.

Sleeping & Resuming a Thread

- El siguiente programa utiliza el método sleep() para mostrar por la consola la fecha a cada segundo.

- Paso a paso:

- 1-Crear la clase llamada FileClock y especificar que implementa la interfaz runnable

```
public class FileClock implements Runnable {
```

Sleeping & Resuming a Thread

- 2-Implemente el método run()
  - Cree un bucle con 10 iteraciones. En cada iteración cree un objeto Date.
  - Luego llame al método sleep() de la clase TimeUnit, atributo SECONDS y suspenda la ejecución del hilo por un segundo.
  - Como el método sleep puede arrojar una InterruptedException, se debe incluir el código para hacer el catch correspondiente.

Sleeping & Resuming a Thread

- 2-Implemente el método run()

```
public void run() {
    for(int i=0;i<10;i++){
        System.out.printf("%s\n", new Date());
        try{
            TimeUnit.SECONDS.sleep(1);
        }catch (InterruptedException e){
            System.out.printf("The FileClock has been interrupted");
            return;
        }
    }
}
```

Sleeping & Resuming a Thread

- 3-Hemos creado el hilo. Ahora crearemos la clase main.
  - Crear un objeto de la clase FileClock.
  - Luego un Thread para ejecutarlo. Y start()

```
public class FileMain {

    public static void main(String[] args) {

        FileClock clock= new FileClock();
        Thread thread=new Thread(clock);

        thread.start();
```

Sleeping & Resuming a Thread

- 4-Llamar al método sleep() por 5 segundos, desde la clase main.
  - Luego interrumpir el hilo.
  - Ejecute y vea los resultados!.

```
        try{
            TimeUnit.SECONDS.sleep(5);
        }catch (InterruptedException e){
            e.printStackTrace();
        }

        thread.interrupt();
    }
```