Uncheked Exceptions & ThreadFactory – Java Concurrency Cookbook

Programación Concurrente 2017

Ing. Ventre, Luis O.

Processing Uncontrolled Exceptions in a Thread

- Hay dos clases de excepciones en JAVA.
  - Checked exceptions: estas excepciones deben ser especificadas en las clausulas Throw or Catch. Son verificadas en tiempo de compilación.
  - **Unchecked exceptions**: No verificadas en tiempo de compilación. El manejo debe ser implementado por el programador.

Processing Uncontrolled Exceptions in a Thread

- Cuando una excepción checked es lanzada dentro de un método run() de un objeto Thread, debemos tratarla, ya que un método run() no acepta clausula Throw.

- Cuando una excepción unchecked es lanzada dentro de un método run() de un objeto Thread, el comportamiento esperado es imprimir el stackTrace en la consola y finalizar el programa.
  - Afortunadamente, Java provee un mecanismo para tratarlas y evitar la finalización abrupta del prog.

Processing Uncontrolled Exceptions in a Thread

- Se vera un ejemplo sobre como implementar el manejo de excepciones unchecked.
  - Primero se implementa la clase para tratar las excepciones unchecked. Esta clase debe implementar la interface UncaughtExceptionHandler, e implementar el método uncaughtException() declarado en la interfaz.
  - En nuestro caso, llamar a esta clase y hacer que el método imprima en la consola datos.

Processing Uncontrolled Exceptions in a Thread

- Luego implementar la clase que lanzara una excepción no chequeada en tiempo de ejecución.
  - Esta clase será Task.
  - Implementara la interfaz runnable.
  - Forzara la excepción, tratando de convertir un string a un int.
- Implemente la clase Main

- Cree un objeto de la clase Task y un Thread para correrlo. Y setear el handler de las excepciones con el método setUncaughtExceptionHandler().

Processing Uncontrolled Exceptions in a Thread

- Como Funciona?
  - Cuando una excepción es lanzada en un hilo y es unchecked, la JVM chequea si el hilo tiene un handler de este tipo de excepciones. Si lo tiene el método correspondiente es ejecutado.
  - Existe otro metodo para el manejo de excepciones setDefaultUncaughtExceptionHandler(), el cual aplica a todos los hilos.
  - Si no lo tiene, la JVM imprime en la consola el stackTrace y finaliza la ejecución.

Processing Uncontrolled Exceptions in a Thread

- Cuando se produce una excepción unchecked en un hilo, la JVM busca 3 posibles manejadores:
  - Primero busca un handler del objeto Thread que lanzo la excepción como se vio en el ejemplo.
  - Si no existe, la JVM busca por un handler para un grupo de hilos, que es mas general que el anterior.
  - Si tampoco existe, la JVM busca por el default método, otro método para el manejo de excepciones setDefaultUncaughtExceptionHandler(), el cual aplica a todos los hilos.
  - Si ninguno existe, imprime el stackTrace y finaliza.

Processing Uncontrolled Exceptions in a Thread

- Código fuente:

ExceptionHandler.java | Task.java | *Main.java

```
1  package com.packtpub.java7.concurrency.chapter1.recipe8.core;
2  import com.packtpub.java7.concurrency.chapter1.recipe8.handler.ExceptionHandler;
4
5  public class Main {
6
7      public static void main(String[] args) {
8          // Creates the Task
9          Task task=new Task();
10         // Creates the Thread
11         Thread thread=new Thread(task);
12         // Sets de uncaugh exceptio handler
13         thread.setUncaughtExceptionHandler(new ExceptionHandler());
14         // Starts the Thread
15         thread.start();
16
17         try {
18             thread.join();
19         } catch (InterruptedException e) {
20             e.printStackTrace();
21         }
22
23         System.out.printf("Thread has finished\n");
24     }
25 }
26
```

Processing Uncontrolled Exceptions in a Thread

ExceptionHandler.java | Task.java | *Main.java

```
4
5  /**
6   * Class that process the uncaught exceptions throwed in a Thread
7   *
8   */
9  public class ExceptionHandler implements UncaughtExceptionHandler {
10
11
12     /**
13      * Main method of the class. It process the uncaught excpetions throwed
14      * in a Thread
15      * @param t The Thead than throws the Exception
16      * @param e The Exception throwed
17      */
18     @Override
19     public void uncaughtException(Thread t, Throwable e) {
20         System.out.printf("An exception has been captured\n");
21         System.out.printf("Thread: %s\n",t.getId());
22         System.out.printf("Exception: %s: %s\n",e.getClass().getName(),e.getMessage());
23         System.out.printf("Stack Trace: \n");
24         e.printStackTrace(System.out);
25         System.out.printf("Thread status: %s\n",t.getState());
26     }
27
28 }
```

Processing Uncontrolled Exceptions in a Thread

ExceptionHandler.java | Task.java | *Main.java

```
1  package com.packtpub.java7.concurrency.chapter1.recipe8.task;
2
3  /**
4   * Runnable class than throws and Exception
5   *
6   */
7  public class Task implements Runnable {
8
9
10     /**
11      * Main method of the class
12      */
13     @Override
14     public void run() {
15         // The next instruction always throws and exception
16         int numero=Integer.parseInt("TTT");
17     }
18
19 }
```

Processing Uncontrolled Exceptions in a Thread

La salida por consola:

Console

```
<terminated> Main (35) [Java Application] C:\Program Files\Java\jre7\bin\javaw.exe (10 de abr. de 2017 6:43:38 p. m.)
An exception has been captured
Thread: 9
Exception: java.lang.NumberFormatException: For input string: "TTT"
Stack Trace:
java.lang.NumberFormatException: For input string: "TTT"
        at java.lang.NumberFormatException.forInputString(Unknown Source)
        at java.lang.Integer.parseInt(Unknown Source)
        at java.lang.Integer.parseInt(Unknown Source)
        at com.packtpub.java7.concurrency.chapter1.recipe8.task.Task.run(Task.java:16)
        at java.lang.Thread.run(Unknown Source)
Thread status: RUNNABLE
Thread has finished
```

Creating Threads Trough a Factory

- El patrón FACTORY, es uno de los mas usados en la programación orientada a objetos.

- Es un patrón de creación, y su objetivo es desarrollar un objeto único cuya misión es la creación de otros objetos de una o varias clases.

- Luego cuando queremos un nuevo objeto de esa clase/s usamos el factory y no el método new().

Creating Threads Trough a Factory

- Con este FACTORY, centralizamos la creación de objetos con algunas ventajas:
  - Es fácil cambiar la clase de objetos creados y la manera en que se crean.
  - Es fácil limitar la creación de objetos para limitados recursos. Por ej. Podemos tener solo n objetos de una determinada clase.
  - Es fácil crear info estadística respecto a la creación de objetos.

Creating Threads Trough a Factory

- Java provee una interfaz, ThreadFactory, para implementar este patrón en Threads.
- En el siguiente ej. se aprenderá a crear hilos a traves de un FACTORY.

- Primero crear la clase MyThreadFactory, y especificar que implementa la interfaz ThreadFactory.
- Declarar 3 atributos:
  - 1 numero entero llamado NUMBER.

Creating Threads Trough a Factory

- Un string con el nombre base para todos los hilos creados.
- Una lista de strings llamada stats, para almacenar datos estadísticos.
- También debe implementarse el constructor que inicializa estos atributos.
- Implementar el método newThread(). Este método recibe un objeto runnable y devuelve un objeto Thread para este runnable. En nuestro ej. Generamos el nombre del hilo, creamos un nuevo hilo y guardamos estadística.

Creating Threads Trough a Factory

  - Implementar el método getStatics(), que devuelve el string con los datos de todos los hilos creados.
- Crear una clase llamada Task y especificar que implementa la interfaz runnable.
  - Para este ej. Esta clase solo dormirá por un segundo.
- Crear la clase Main.
- Crear un objeto MyThreadFactory.
- Crear objetos Threads usando el FACTORY y lanzarlos.
- Imprimir estadísticas en la consola!...

Creating Threads Trough a Factory

- El código: Clase MyThreadFactory

*MyThreadFactory.java

```
1  package com.packtpub.java7.concurrency.chapter1.recipe12.factory;
2  import java.util.ArrayList;
7
8  public class MyThreadFactory implements ThreadFactory {
9
10     // Attributes to save the necessary data to the factory
11     private int counter;
12     private String name;
13     private List<String> stats;
14
16      * Constructor of the class
19     public MyThreadFactory(String name){
20         counter=0;
21         this.name=name;
22         stats=new ArrayList<String>();
23     }
24
26      * Method that creates a new Thread object using a Runnable object
```

Creating Threads Trough a Factory

- El código: Clase MyThreadFactory cont.

```
@Override
public Thread newThread(Runnable r) {
    // Create the new Thread object
    Thread t=new Thread(r,name+"-Thread_"+counter);
    counter++;
    // Actualize the statistics of the factory
    stats.add(String.format("Created thread %d with name %s on %s\n",
            t.getId(),t.getName(),new Date()));
    return t;
}

/**
 * Method that returns the statistics of the ThreadFactory
 * @return The statistics of the ThreadFactory
 */
public String getStats(){
    StringBuffer buffer=new StringBuffer();
    Iterator<String> it=stats.iterator();

    while (it.hasNext()) {
        buffer.append(it.next());
    }

    return buffer.toString();
}
```

Creating Threads Trough a Factory

- El código: Clase Task

MyThreadFactory.java | Task.java

```
1  package com.packtpub.java7.concurrency.chapter1.recipe12.task;
2
3  import java.util.concurrent.TimeUnit;
4
5  public class Task implements Runnable {
6
7      @Override
8      public void run() {
9          try {
10             TimeUnit.SECONDS.sleep(1);
11         } catch (InterruptedException e) {
12             e.printStackTrace();
13         }
14     }
15
16 }
17
```

Creating Threads Trough a Factory

- El código: Clase Main

MyThreadFactory.java | Task.java | *Main.java

```
1  package com.packtpub.java7.concurrency.chapter1.recipe12.core;
2
3  import com.packtpub.java7.concurrency.chapter1.recipe12.factory.MyThreadFactory;
6   * Main class of the example. Creates a Thread factory and creates ten
10 public class Main {
12     * Main method of the example. Creates a Thread factory and creates
16     public static void main(String[] args) {
17         // Creates the factory
18         MyThreadFactory factory=new MyThreadFactory("MyThreadFactory");
19         // Creates a task
20         Task task=new Task();
21         Thread thread;
22
23         // Creates and starts ten Thread objects
24         System.out.printf("Starting the Threads\n");
25         for (int i=0; i<10; i++){
26             thread=factory.newThread(task);
27             thread.start();
28         }
29         // Prints the statistics of the ThreadFactory to the console
30         System.out.printf("Factory stats:\n");
31         System.out.printf("%s\n",factory.getStats());
32
33     }
```

Creating Threads Trough a Factory

- El código: Clase Main

MyThreadFactory.java | Task.java | *Main.java

```
1  package com.packtpub.java7.concurrency.chapter1.recipe12.core;
2
3  import com.packtpub.java7.concurrency.chapter1.recipe12.factory.MyThreadFactory;
6   * Main class of the example. Creates a Thread factory and creates ten
10 public class Main {
12     * Main method of the example. Creates a Thread factory and creates
16     public static void main(String[] args) {
17         // Creates the factory
18         MyThreadFactory factory=new MyThreadFactory("MyThreadFactory");
19         // Creates a task
20         Task task=new Task();
21         Thread thread;
22
23         // Creates and starts ten Thread objects
24         System.out.printf("Starting the Threads\n");
25         for (int i=0; i<10; i++){
26             thread=factory.newThread(task);
27             thread.start();
28         }
29         // Prints the statistics of the ThreadFactory to the console
30         System.out.printf("Factory stats:\n");
31         System.out.printf("%s\n",factory.getStats());
32
33     }
```

Creating Threads Trough a Factory

- Como Funciona?
- El FACTORY de Threads tiene solo un método llamado newThread.
  - Este recibe un objeto runnable como parámetro y devuelve un objeto Thread.
  - Muchos ThreadFactory tiene solo una línea return new Thread(r).
  - Puede mejorar esto:
    - Creando hilos personalizados.
    - Guardando estadísticas.
    - Validando la creación de hilos….etc

Creating Threads Trough a Factory

- Es importante, asegurarse en nuestro software, si se crean hilos a traves de un Factory, todos los hilos del programa se creen de la misma manera.
- La salida: