APRENDERAPROGRAMAR.COM

GESTIÓN DE EXCEPCIONES EN JAVA. CAPTURA CON BLOQUES TRY CATCH Y FINALLY. EJEMPLOS RESUELTOS. (CU00927C)

Sección: Cursos
Categoría: Lenguaje de programación Java nivel avanzado I

Fecha revisión: 2029

Resumen: Entrega nº27 curso "Lenguaje de programación Java Nivel Avanzado I".

Autor: Manuel Sierra

INTRODUCCIÓN: GESTIÓN EXCEPCIONES EN JAVA

A continuación vamos a ver como el lenguaje Java implementa su propio sistema de gestión de excepciones, o como hemos mencionado anteriormente, también llamado sistema de tratamiento de errores. También veremos los primeros ejemplos sencillos sobre la gestión de excepciones.

EL SISTEMA DE GESTIÓN DE EXCEPCIONES

El control de flujo de un programa Java sabemos hasta ahora que se lleva a cabo con sentencias del tipo if, while, for, return, break, etc… Estas sentencias forman un conjunto de palabras reservadas que determinan cierta funcionalidad. Pues bien, ninguna de ellas tiene en cuenta que se puedan producir errores en tiempo de ejecución de un programa y por tanto Java necesita de un conjunto de palabras nuevas para tener en cuenta que cualquier código puede fallar o ser mal interpretado en tiempo de ejecución.

Vamos a ver tres de las palabras reservadas para tratamiento de excepciones:

- Try.
- Catch.
- Finally.

Aunque posteriormente veremos otras palabras más avanzadas y otras formas de tratamiento de errores, éstas son las primeras y más básicas con las que vamos a trabajar.

De forma introductoria diremos que hay dos formas de tratar errores en Java: capturarlos o lanzarlos. El uso de try – catch – finally corresponde a la captura de errores. Vamos a poner un símil sencillo: un error es algo inesperado, como encontrarte un ladrón dentro de tu casa. Cuando nos encontramos con un error podemos capturarlo (equivaldría a capturar el ladrón) o lanzarlo (equivaldría a tratar de hacer huir al ladrón, de hacer que salga fuera del lugar donde se encuentra).

BLOQUE TRY

Try en inglés es el verbo intentar, así que todo el código que vaya dentro de esta sentencia será el código sobre el que se intentará capturar el error si se produce y una vez capturado hacer algo con él. Lo ideal es que no ocurra un error, pero en caso de que ocurra un bloque try nos permite estar preparados para capturarlo y tratarlo. Así un ejemplo sería:

```
try {
    System.out.println("bloque de código donde pudiera saltar un error es este");
    }
```

© aprenderaprogramar.com, 2006-2029

BLOQUE CATCH

En este bloque definimos el conjunto de instrucciones necesarias o de tratamiento del problema capturado con el bloque try anterior. Es decir, cuando se produce un error o excepción en el código que se encuentra dentro de un bloque try, pasamos directamente a ejecutar el conjunto de sentencias que tengamos en el bloque catch. Esto no es exactamente así pero ya explicaremos más adelante todo el funcionamiento. De momento para una mejor comprensión vamos a considerar que esto es así.

```
catch (Exception e) {
    System.out.println("bloque de código donde se trata el problema");
}
```

Fíjate que después de catch hemos puesto unos paréntesis donde pone "Exception e". Esto significa que cuando se produce un error Java genera un objeto de tipo Exception con la información sobre el error y este objeto se envía al bloque catch.

BLOQUE FINALLY

Y para finalizar tenemos el bloque finally que es un bloque donde podremos definir un conjunto de instrucciones necesarias tanto si se produce error o excepción como si no y que por tanto se ejecuta siempre.

```
finally {
    System.out.println("bloque de código ejecutado siempre");
}
```

EJEMPLO SIN ERROR

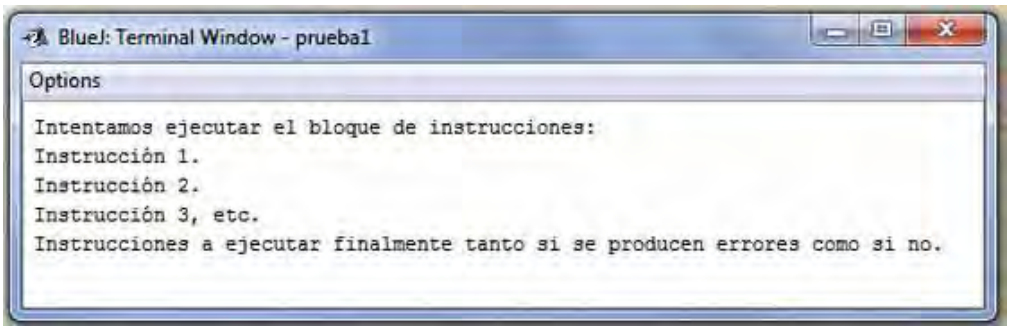
A continuación vamos a ver cómo se comporta un programa con tratamiento de errores pero donde no se produce ningún error. Escribe este código en tu editor.

```
/* Ejemplo Gestión de Excepciones Java aprenderaprogramar.com */
public class Programa {
   public static void main (String [] args)    {
      try{
         System.out.println("Intentamos ejecutar el bloque de instrucciones:");
         System.out.println("Instrucción 1.");          System.out.println("Instrucción 2.");
         System.out.println("Instrucción 3, etc.");
      }
      catch (Exception e) { System.out.println("Instrucciones a ejecutar cuando se produce un error");  }

     finally{ System.out.println("Instrucciones a ejecutar finalmente tanto si se producen errores como si no."); }
   }
}
```

© aprenderaprogramar.com, 2006-2029

La salida obtenida tras ejecutar el programa anterior es:



Como podemos observar, se han ejecutado todas las instrucciones del bloque try y finalmente se ejecutó la instrucción del bloque finally. No se ejecuta el bloque catch porque no hubo error.

EJEMPLO CON ERROR

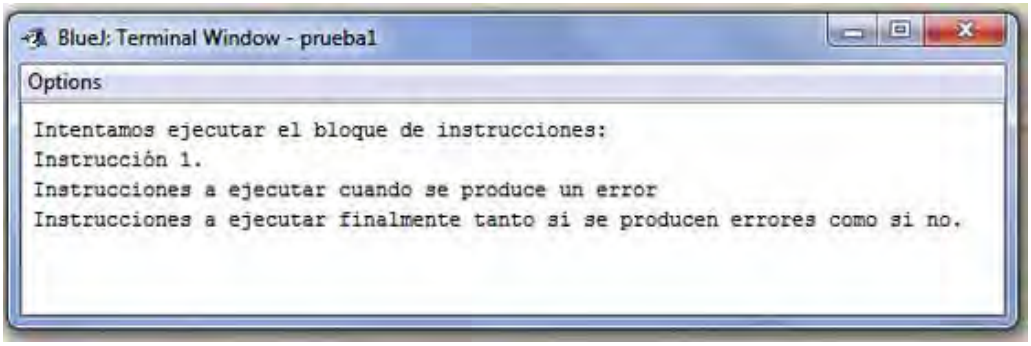
A continuación vamos a ver cómo se comporta un programa con tratamiento de errores cuando se produce un error y cómo afecta al control de flujo del programa. Escribe este código en tu editor.

```
/* Ejemplo Gestión de Excepciones Java aprenderaprogramar.com */
public class Programa {
   public static void main (String [] args)    {
      try{
         System.out.println("Intentamos ejecutar el bloque de instrucciones:");
         System.out.println("Instrucción 1.");
         int n = Integer.parseInt("M");//error forzado en tiempo de ejecución.
         System.out.println("Instrucción 2.");
         System.out.println("Instrucción 3, etc.");
      }
      catch(Exception e){
         System.out.println("Instrucciones a ejecutar cuando se produce un error");
      }
      finally{
         System.out.println("Instrucciones a ejecutar finalmente tanto si se producen errores como si no.");
      }
   }
}
```

Se produce un error porque el método parseInt de la clase Integer espera que dentro de las comillas llegue un número y no una letra. Por ejemplo int n = Integer.parseInt("65"); sirve para transformar el String 65 en un int de valor 65. Al no encontrar un valor válido se produce un error de tipo java.lang.NumberFormatException.

© aprenderaprogramar.com, 2006-2029

La salida obtenida en este caso donde se produce error es:



Prueba a escribir dentro del bloque catch lo siguiente:

System.out.println("Se ha producido un error " +e);

Trata de interpretar lo que se visualiza en pantalla. Si tienes dudas consulta en los foros de aprenderaprogramar.com.

Como podemos observar, ejecutamos las instrucciones del bloque try que no dan errores, pero cuando en una instrucción se produce un error o excepción inesperada se deja de ejecutar el código del bloque try, y pasamos a ejecutar el código del bloque catch. Hay un salto o cambio en el flujo del programa.

Finalmente se ejecutan, en todo caso, las instrucciones del bloque finally como hemos comentado anteriormente. El bloque finally no es obligatorio, es decir, puede existir un bloque try catch y no existir bloque finally.

CONCLUSIONES

Los errores en Java se pueden capturar o lanzar. La captura se realiza con bloques try catch, donde por un lado tenemos el código para el caso de flujo sin problemas del programa y por otro el código con instrucciones para el tratamiento de errores.

Próxima entrega: CU00928C

Acceso al curso completo en aprenderaprogramar.com -- > Cursos, o en la dirección siguiente:
http://aprenderaprogramar.com/index.php?option=com_content&view=category&id=58&Itemid=180

© aprenderaprogramar.com, 2006-2029