

Programa Educativo:

Tecnologías de la Información y Comunicación

**Investigación sobre el Manejo de Excepciones**

Área:

Sistemas Informáticos

Materia:

Programación

Profesor:

Apolinar Trejo Cuevas

Autor:

Miguel Ángel Camacho Campos

Grupo: GSI1321 No. Control: 1216100766

Introducción

Las excepciones son muy importantes incluirlas siempre en cualquier programa que se realice ya que si no se incluyen el programa desarrollado no sería eficiente y tendría errores, por lo que se puede definir una excepción como un medio para evitar errores y que el programa de alguna forma no arroje los resultados esperados.

En esta investigación de abordaran varios puntos muy importantes sobre el manejo de las excepciones tales como que es una excepción, como se controla cuáles son las más comunes y porque es muy importante incluirlas.

En la programación siempre se producen errores, más o menos graves, pero que hay que gestionar y tratar correctamente. Por ello en java disponemos de un mecanismo consistente en el uso de bloques try/catch/finally. La técnica básica consiste en colocar las instrucciones que podrían provocar problemas dentro de un bloque try, y colocar a continuación uno o más bloques catch, de tal forma que si se provoca un error de un determinado tipo, lo que haremos será saltar al bloque catch capaz de gestionar ese tipo de error específico. El bloque catch contendrá el código necesario para gestionar ese tipo específico de error. Suponiendo que no se hubiesen provocado errores en el bloque try, nunca se ejecutarían los bloques catch.

Contenido

**¿Qué son las excepciones?**

En Java, como en cualquier otro lenguaje de programación los errores en tiempo de ejecución (cuando se está ejecutando el programa) se denominan **excepciones**, y esto ocurre cuando se produce un error en alguna de las instrucciones de nuestro programa, como por ejemplo cuando se hace una división entre cero, cuando un objeto es 'null' y no puede serlo, cuando no se abre correctamente un fichero, etcétera. Cuando se produce una excepción se muestra en la pantalla un mensaje de error y finaliza la ejecución del programa.

**¿Cómo se controlan éstas?**

Por suerte Java nos permite hacer un control de las excepciones para que nuestro programa no se pare inesperadamente y aunque se produzca una excepción, nuestro programa siga su ejecución. Para ello tenemos la estructura "try – catch – finally" que la mostramos a continuación:



NumberFormatEception

**Fuente:** Propia

Cuando se necesita poner 2 o más excepciones en un bloque se hacer un try catch con un finallty.

**¿Cuáles son las excepciones más comunes?**

Recordemos antes que tenemos dos tipos de excepciones las cuales son las siguientes:

* **Por la JVM:**

Estas excepciones o errores son exclusivamente o más lógicamente lanzadas por la JVM.

* **Programáticas:**

 Estas excepciones son lanzadas explícitamente por la aplicación y/o programadores de una API. Que son las creadas por el desarrollador de una API o una aplicación.

En el siguiente cuadro se presentan las excepciones más comunes, así como quien las lanza.

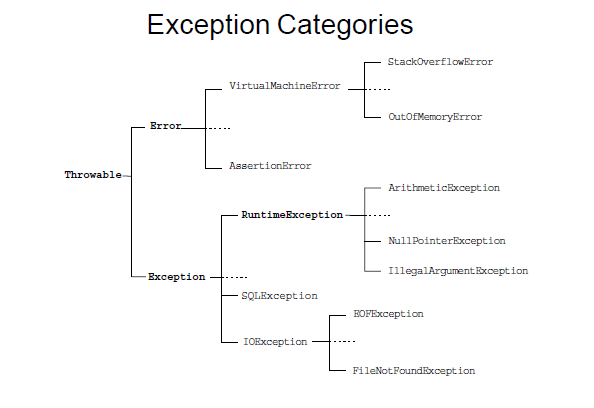
|  |  |  |
| --- | --- | --- |
| Excepción | Descripción | Lanzado por |
| ArrayIndexOutOfBoundsException | Es lanzada cuando intentamos acceder a un array con un index inválido (menor que 0 o mayor que la longitud del array). | Por la JVM |
| ClassCastException | Lanzado cuando intentamos hacer cast a una variable de referencia que no pasa el test IS-A | Por la JVM |
| IllegalArgumentException | Lanzado cuando un método recibe un argumento formateado de diferente manera del que el método espera. | Programáticamente |
| IllegalStateException | Lanzado cuando el estado del entorno no coincide con la operación que se intenta llevar a cabo. (Usar un Scaner que no ha sido cerrado) | Programáticamente |
| NullPointerException | Lanzado cuando intentamos acceder a un objeto con una variable de referencia cuyo valor actual es null. | Por la JVM |
| NumberFormatException | Lanzado cuando un método convierte un String a un número y ese String no puede ser convertido. | Programáticamente |
| AssertionError | Lanzado cuando una sentencia que comprueba un booleano devuelve false. | Programáticamente |
| ExceptionInInitializerError | Lanzado cuando intentamos inicializar una variable static en un bloque de inicialización. | Por la JVM |
| StackOverflowError | Típicamente lanzado cuando un método es muy recursivo y cada llamada se va añadiendo al stack. | Por la JVM |
| NoClassDefFoundError | Lanzado cuandl la JVM no puede encontrar la clase que se necesita, porque hay un error en la linea de comandos, o un problema con el classpath, ochivo .class que no se encuentra. | Por la JVM |

**Las excepciones se generan por las siguientes situaciones:**

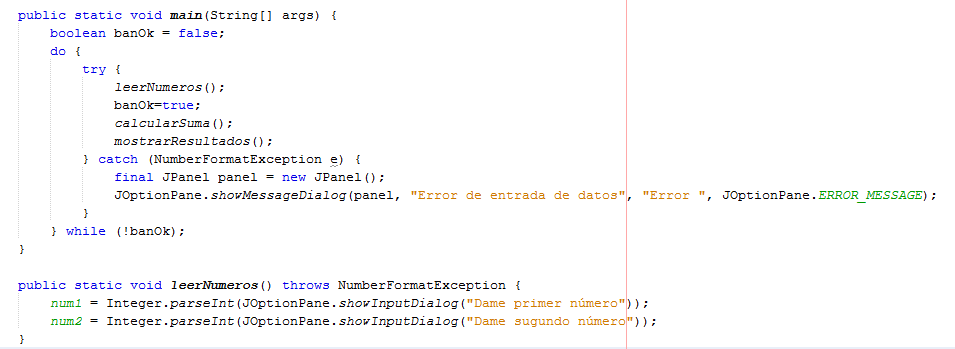
* No hay memoria disponible para asignar
* Acceso a un elemento de un array fuera de rango
* Leer por teclado un dato de un tipo distinto al esperado
* Error al abrir un fichero
* División por cero
* Problemas de Hardware

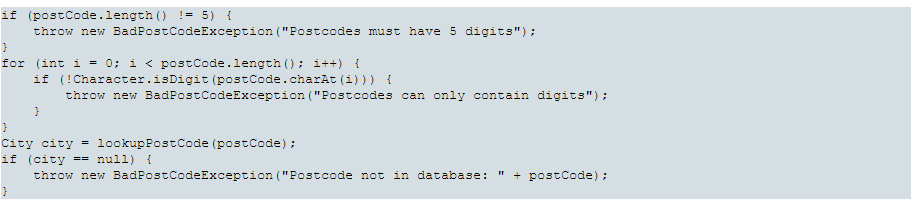
**Cuando ocurre una excepción:**

* La Máquina Virtual Java crea un objeto excepción y lo lanza. El objeto excepción creado contiene información sobre el error. La ejecución normal del programa se detiene.
* El sistema busca en el método donde se ha producido la excepción un manejador de excepciones que capture ese objeto y trate la excepción.
* Si el método no contiene un manejador para la excepción se busca en el método que llamó a este y así sucesivamente en toda la pila de llamadas.
* Cuando se encuentra un manejador apropiado se le pasa la excepción. Un manejador de excepciones es considerado apropiado si el tipo de objeto excepción lanzado es compatible al tipo que puede manejar.
* Si no se encuentra un manejador adecuado la Máquina Virtual Java muestra el error y acaba el programa.



**Ejemplos:**





Conclusiones

Ahora conocemos los esenciales de excepciones en java. El control o manejo de excepciones permite crear programas más tolerantes a fallas. Conocimos cómo declarar una excepción y utilizar los bloques o sentencias try, catch, finally, throw. Además, para el uso de la sentencia using simplifica la liberación de recursos en contraste con try-catch-finally. Al final exploramos cómo se compartan las excepciones en métodos asincrónicos, y varias propiedades de las excepciones.

El manejo de excepciones ayuda a lidiar con los errores de una aplicación por medio de la manipulación del código para hacer programas más robustos. Además existen herramientas que ayudan a manejarlas tal es el caso de los bloques try (intentar) que encierran el código que puede lanzar una excepción y los bloques el catch (atrapar) que lidian con las excepciones que surjan. También existen técnicas que el programador utiliza para conocer el posible funcionamiento del programa y detectar los errores que pueda contener.

Bibliografías

**JAVA ILLUMINATED**

3er edición

Autores: [Hervé Franceschi](https://www.google.com.mx/search?q=herve+franceschi&stick=H4sIAAAAAAAAAOPgE-LRT9c3NCqLzygyKDZVgvAKLDNMs4zjtWSyk630k_Lzs_XLizJLSlLz4svzi7KtEktLMvKLACvc-oc8AAAA&sa=X&ved=0ahUKEwi63t6o_cDTAhWBx4MKHQGjBjEQmxMIlQEoATAO),

[Julie Anderson](https://www.google.com.mx/search?q=java+illuminated+julie+anderson&stick=H4sIAAAAAAAAAOPgE-LRT9c3NCqLzygyKDZV4gXxDJPNyg1yKyyqtGSyk630k_Lzs_XLizJLSlLz4svzi7KtEktLMvKLAK77Ov49AAAA&sa=X&ved=0ahUKEwi63t6o_cDTAhWBx4MKHQGjBjEQmxMIlgEoAjAO)

Editorial: Mc Graw Hill

**ENCICLOPEDIA DEL LENJUAGE JAVA**

1er edición (2010)

Autores: Francisco Javier Ceballos Sierra

Editorial: Alfaomega