**02230131025 – Paula Andrea Muñoz Preciado**

1. **Excepciones Controladas (Checked Exceptions):**

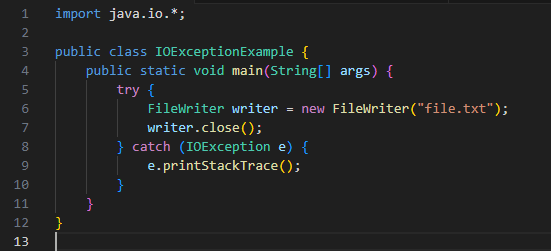
Las Excepciones Controladas, también conocidas como Checked Exceptions en Java, son aquellas que el compilador obliga a manejar explícitamente en el código. Cuando un método puede generar una excepción controlada, debe declararla en su firma utilizando la palabra clave `throws`, lo que indica que el método podría lanzar esa excepción y que el código que lo llama debe manejarla o propagarla.

Estas excepciones generalmente representan condiciones que están fuera del control del programador pero que aún pueden ocurrir durante la ejecución del programa y que deben ser manejadas adecuadamente para garantizar la robustez y la integridad del software.

El hecho de que se les llame "checked" (controladas) se debe a que el compilador verifica si el código que llama a un método que puede lanzar una excepción controlada está manejando esa excepción o propagándola a su vez. Esto ayuda a garantizar que se tomen medidas adecuadas para manejar posibles errores que puedan ocurrir durante la ejecución del programa.

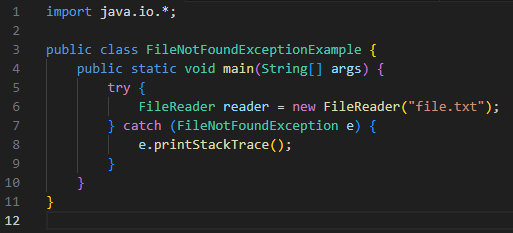
* **IOException:**

Esta excepción se lanza cuando ocurre un error relacionado con la entrada o salida de datos, como fallos al leer o escribir archivos.



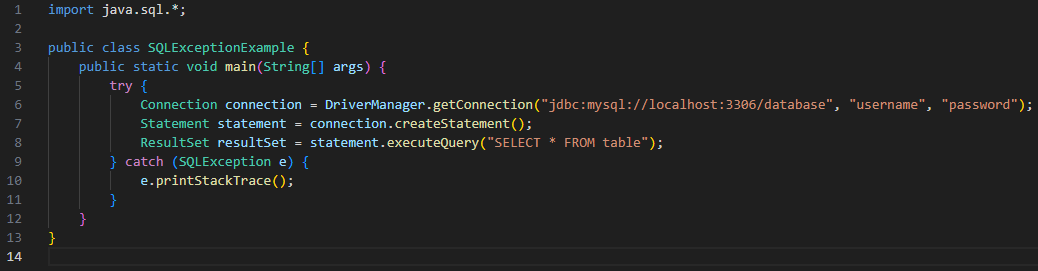
* **FileNotFoundException:**

Se produce cuando un intento de abrir un archivo para lectura no encuentra el archivo especificado en el sistema de archivos.



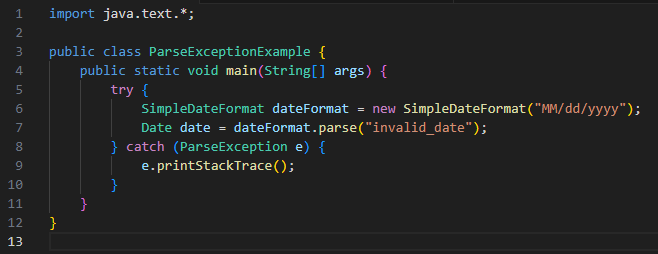
* **SQLException:**

Esta excepción se produce cuando ocurre un error al interactuar con una base de datos a través de JDBC (Java Database Connectivity).



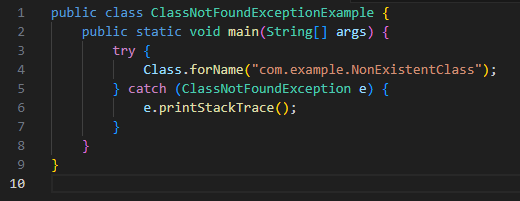
* **ParseException:**

Lanzada cuando ocurre un error al analizar (convertir) una cadena en un formato específico, como convertir una cadena en una fecha.



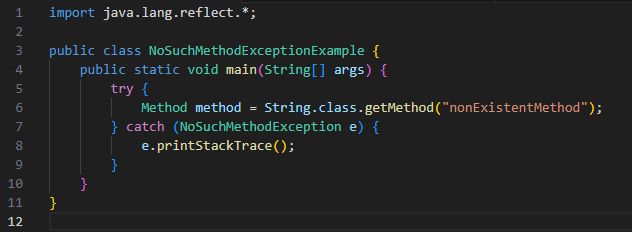
* **ClassNotFoundException:**

Se lanza cuando el código intenta cargar una clase en tiempo de ejecución utilizando Class.forName() pero la clase especificada no se encuentra en el classpath.



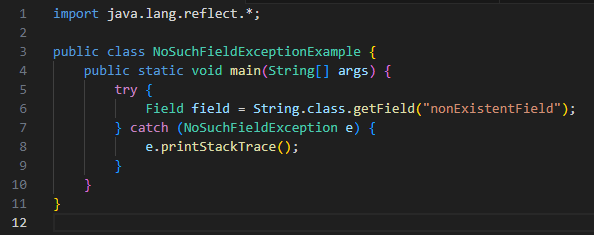
* **NoSuchMethodException:**

Se produce cuando un método especificado no se encuentra en una clase. Esto podría ocurrir al intentar acceder a un método que no existe en un objeto utilizando reflexión.



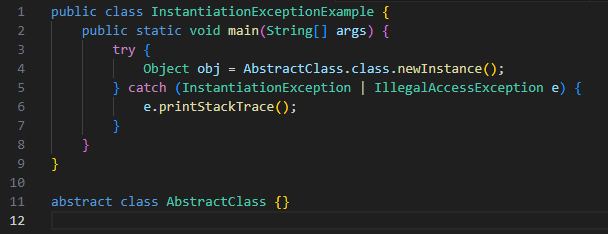
* **NoSuchFieldException:**

Lanzada cuando se intenta acceder a un campo (variable de instancia) en una clase usando reflexión y el campo especificado no existe.



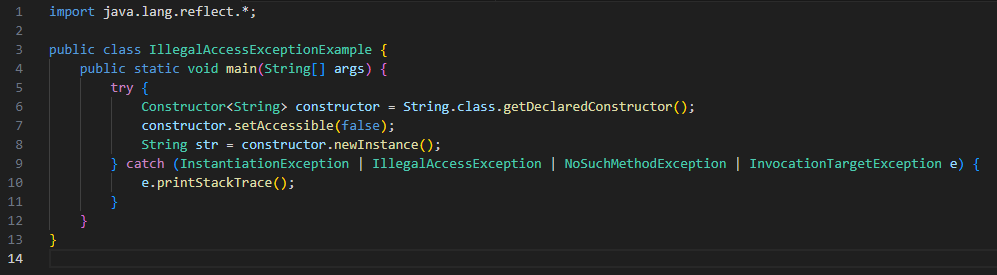
* **InstantiationException:**

Ocurre cuando se intenta instanciar una clase abstracta o una interfaz, o cuando se intenta instanciar una clase que no tiene un constructor accesible.



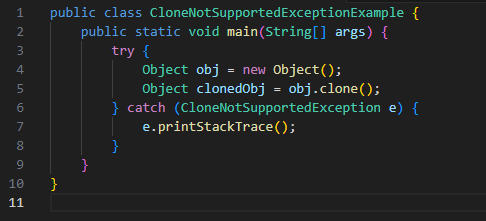
* **IllegalAccessException:**

Se lanza cuando se intenta acceder a un miembro (método o campo) de una clase que no se puede acceder debido a la configuración de acceso (por ejemplo, el modificador de acceso private).



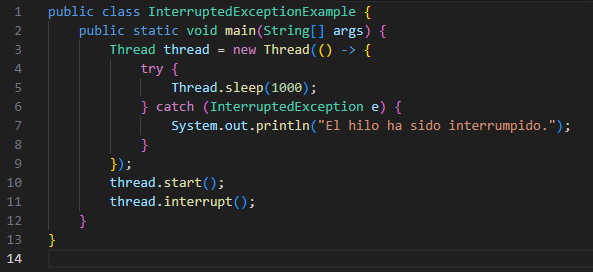
* **CloneNotSupportedException:**

Se produce cuando se intenta clonar un objeto utilizando el método clone(), pero la clase del objeto no implementa la interfaz Cloneable, o si el método clone() no está disponible (por ejemplo, si se ha anulado como protected o no se ha implementado).



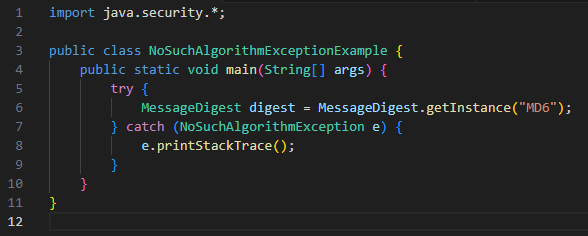
* **InterruptedException:**

Esta excepción se lanza cuando un hilo en ejecución es interrumpido, típicamente porque otro hilo llama al método interrupt() en el hilo que está siendo interrumpido.



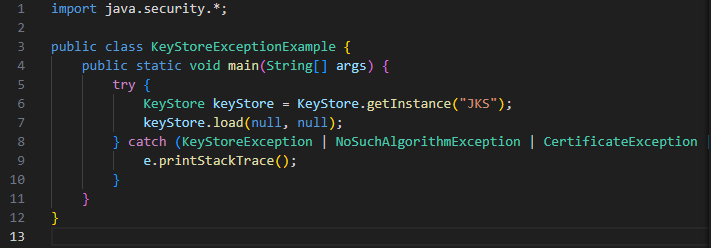
* **NoSuchAlgorithmException:**

Se produce cuando un algoritmo de cifrado solicitado no está disponible en el entorno de ejecución. Por ejemplo, al intentar utilizar un algoritmo de cifrado que no está implementado en el proveedor de seguridad de Java.



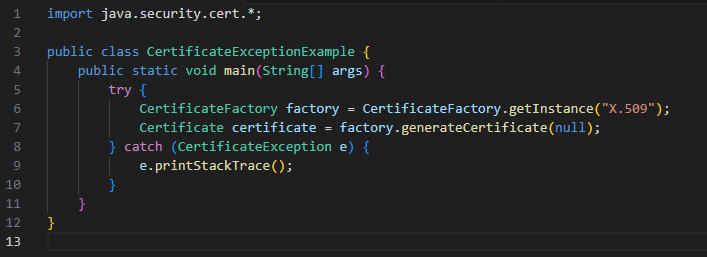
* **KeyStoreException:**

Lanzada cuando ocurre un error relacionado con el almacenamiento de claves, como problemas al acceder o manipular el keystore (almacén de claves) en Java.



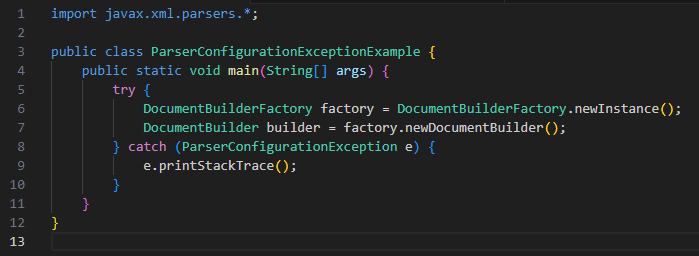
* **CertificateException:**

Se produce cuando se produce un error en la carga o validación de un certificado digital. Esto podría ser debido a un certificado inválido o corrupto.



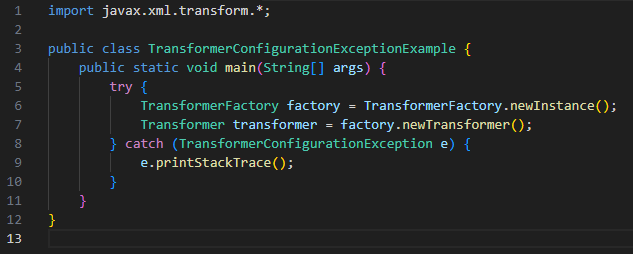
* **ParserConfigurationException:**

Lanzada durante la configuración de un parser XML si se produce un error en la configuración del parser, como configuraciones incorrectas o no válidas.



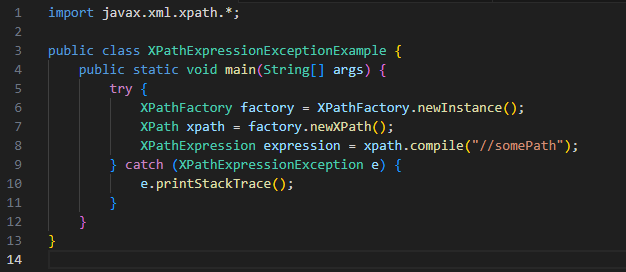
* **TransformerConfigurationException:**

Se produce cuando ocurre un error en la configuración de un transformador XML, por ejemplo, si la plantilla XSLT utilizada no es válida.



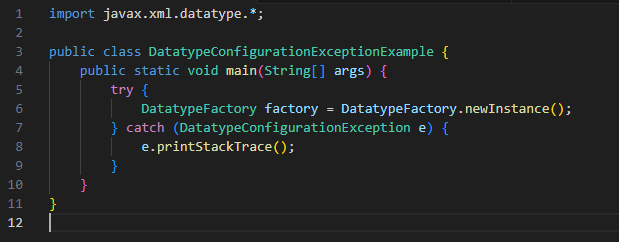
* **XPathExpressionException:**

Lanzada cuando ocurre un error al evaluar una expresión XPath durante el procesamiento XML, como una expresión sintácticamente incorrecta o que no puede evaluarse correctamente.



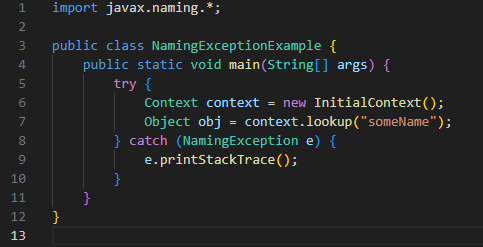
* **DatatypeConfigurationException:**

Se produce cuando se produce un error en la configuración de tipos de datos JAXB (Java Architecture for XML Binding), como cuando no se puede encontrar o configurar un factory para tipos de datos JAXB.



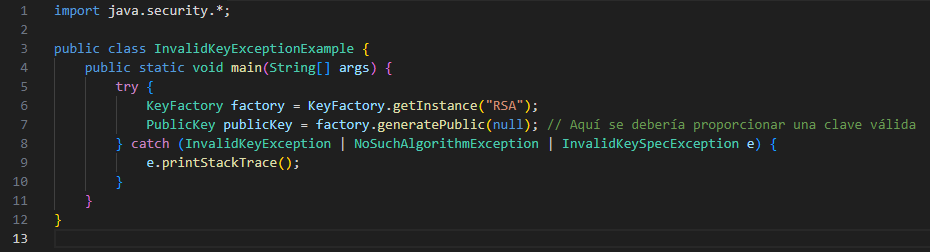
* **NamingException:**

Esta excepción es la superclase de todas las excepciones relacionadas con operaciones de naming y directory en JNDI (Java Naming and Directory Interface). Se produce cuando ocurre un error en operaciones como búsqueda, enlace o eliminación de objetos en un servicio de nombres.



* **InvalidKeyException:**

Se lanza cuando se encuentra una clave no válida durante operaciones criptográficas, como cifrado o firma digital. Por ejemplo, si se intenta utilizar una clave inválida para un algoritmo de cifrado específico.



1. **Excepciones No Controladas (Unchecked Exceptions o Runtime Exceptions):**

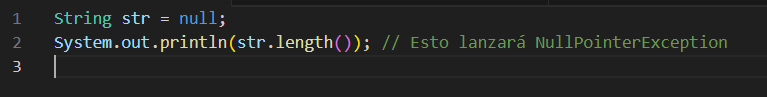
Las Excepciones No Controladas (Unchecked Exceptions), también conocidas como Runtime Exceptions, son un tipo de excepción en Java que el compilador no requiere que sean manejadas explícitamente mediante bloques try-catch o que sean declaradas en la firma del método o la cláusula throws.

Estas excepciones generalmente ocurren debido a errores en el código del programador o condiciones imprevistas en tiempo de ejecución que no pueden ser fácilmente anticipadas o recuperadas. Algunos ejemplos comunes de excepciones no controladas incluyen `NullPointerException`, `ArrayIndexOutOfBoundsException`, `ArithmeticException`, entre otras.

Aunque el manejo de excepciones no controladas no es obligatorio, es una buena práctica anticipar y evitar estas excepciones siempre que sea posible mediante la escritura de código robusto y defensivo. En casos donde no se pueda evitar la posibilidad de excepciones no controladas, se pueden utilizar bloques try-catch para manejarlas apropiadamente o dejarlas propagar hacia arriba en la pila de llamadas para ser manejadas por componentes superiores.

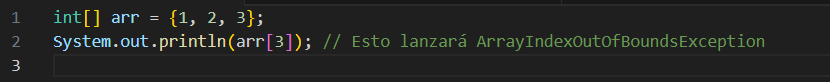
* **NullPointerException:**

Esta excepción se produce cuando intentas acceder a un objeto que es null. Por ejemplo, si intentas llamar a un método o acceder a un campo de un objeto que no ha sido inicializado.



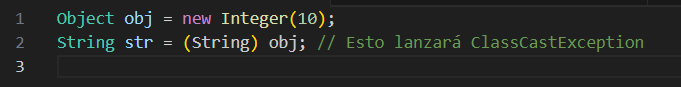
* **ArrayIndexOutOfBoundsException:**

Se lanza cuando intentas acceder a un índice fuera del rango válido de un array. Por ejemplo, si intentas acceder al índice -1 o a un índice mayor que el tamaño del array.



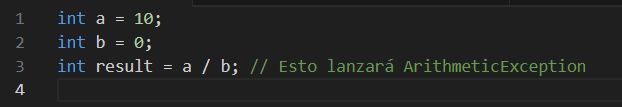
* **ClassCastException:**

Esta excepción ocurre cuando intentas realizar un casting de un objeto a un tipo que no es compatible. Por ejemplo, si intentas convertir un objeto de una clase a otra clase que no está relacionada por herencia o implementación de interfaces.



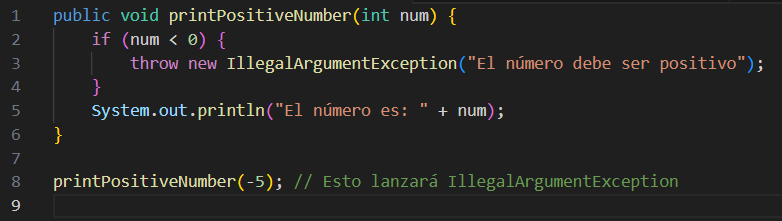
* **ArithmeticException:**

Se produce cuando se produce un error aritmético durante una operación matemática. Por ejemplo, si intentas dividir un número por cero.



* **IllegalArgumentException:**

Esta excepción indica que un método ha sido pasado un argumento ilegal o inapropiado. Por ejemplo, si pasas un valor negativo a un método que espera un número positivo.



* **IllegalStateException:**

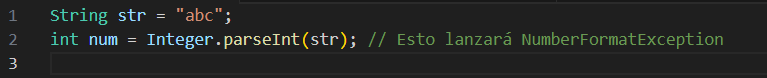
Se lanza cuando un objeto no está en el estado adecuado para realizar una operación. Por ejemplo, intentar acceder a un recurso que ya ha sido cerrado.

Texto

Descripción generada automáticamente

* **NumberFormatException:**

Esta excepción se lanza cuando se intenta convertir una cadena a un número, pero la cadena no tiene el formato correcto. Por ejemplo, si intentas convertir la cadena "abc" a un número.



* **UnsupportedOperationException:**

Se produce cuando se intenta realizar una operación que no está soportada por una clase. Por ejemplo, si intentas modificar una colección que no permite modificaciones.

Interfaz de usuario gráfica, Texto

Descripción generada automáticamente

* **ConcurrentModificationException:**

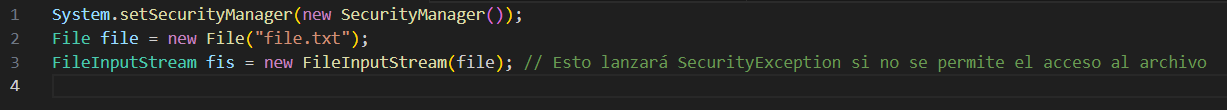
Esta excepción se produce cuando se intenta modificar una colección mientras se está iterando sobre ella de forma concurrente (por ejemplo, en un bucle for-each). Esto puede ocurrir, por ejemplo, si se añaden o eliminan elementos de una lista mientras se está iterando sobre ella.

Texto

Descripción generada automáticamente

* **SecurityException:**

Se lanza cuando se produce un error de seguridad. Por ejemplo, si intentas acceder a un recurso que no tienes permiso para utilizar.



* **TypeNotPresentException:**

Esta excepción se lanza cuando una clase no se puede encontrar en tiempo de ejecución. Esto puede ocurrir, por ejemplo, cuando intentas cargar una clase que no está presente en el classpath.

Interfaz de usuario gráfica, Texto

Descripción generada automáticamente

* **DateTimeException:**

Esta excepción es la clase base para todas las excepciones relacionadas con operaciones de fecha y hora en la API de fecha y hora de Java. Puede ser lanzada, por ejemplo, cuando ocurre un error al manipular una fecha o una hora.

Texto

Descripción generada automáticamente

* **DateTimeParseException:**

Esta excepción se lanza cuando ocurre un error al analizar una cadena en un objeto de fecha y hora utilizando el método parse() de la clase DateTimeFormatter.

Texto

Descripción generada automáticamente

* **EnumConstantNotPresentException:**

Se lanza cuando intentas acceder a un valor de un enumerado que no está definido en tiempo de ejecución. Esto puede ocurrir si el enumerado ha sido modificado desde que se compiló el código.

Texto

Descripción generada automáticamente

* **IllegalMonitorStateException:**

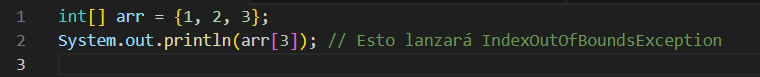
Esta excepción se lanza cuando ocurre un error al utilizar los métodos wait(), notify(), o notifyAll() en un objeto sin tener el bloqueo correspondiente adquirido.

Texto

Descripción generada automáticamente

* **IndexOutOfBoundsException:**

Esta excepción se lanza cuando intentas acceder a un índice fuera del rango válido de una estructura de datos, como un array, una lista o una cadena.



* **NegativeArraySizeException:**

Se lanza cuando intentas crear un array con un tamaño negativo. Esto es claramente un error en la lógica del programa.



* **NoSuchElementException:**

Esta excepción se lanza por varias clases de la API de colecciones, como por ejemplo Iterator y Scanner, cuando intentas acceder a un elemento que no existe.

Texto

Descripción generada automáticamente

* **PatternSyntaxException:**

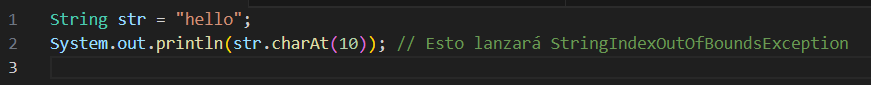
Esta excepción se lanza cuando ocurre un error al analizar una expresión regular. Por ejemplo, si hay un error de sintaxis en la expresión regular proporcionada.

Texto

Descripción generada automáticamente

* **StringIndexOutOfBoundsException:**

Se lanza cuando intentas acceder a un índice fuera del rango válido de una cadena.



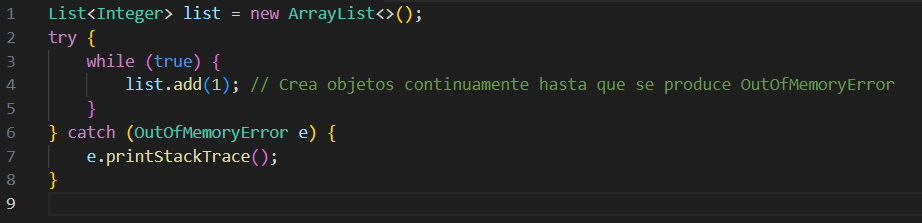
1. **Errores (Errors):**

En Java, los Errores (Errors) son problemas que normalmente surgen debido a circunstancias que escapan al control del programador y que, por lo general, no deberían ser capturados o manejados en tiempo de ejecución. A menudo, los Errores indican condiciones graves que pueden hacer que la aplicación deje de funcionar correctamente o que se vuelva inestable.

Los Errores no deben confundirse con las Excepciones. Mientras que las Excepciones se utilizan para manejar condiciones excepcionales o no deseadas que pueden ocurrir durante la ejecución del programa y que podrían ser recuperadas o controladas, los Errores generalmente indican problemas más graves que pueden requerir intervención a nivel del sistema o de la infraestructura en lugar de soluciones dentro del código de la aplicación.

* **OutOfMemoryError:**

Este error ocurre cuando la aplicación no tiene suficiente memoria para alojar más objetos. Puede ocurrir debido a una fuga de memoria (cuando los objetos no son liberados adecuadamente), a la creación de un número excesivo de objetos grandes o a la falta de memoria asignada a la JVM.



* **StackOverflowError:**

Se produce cuando la pila de llamadas de un programa se desborda. Esto generalmente ocurre debido a una recursión infinita o a un número excesivo de llamadas a métodos que se colocan en la pila de llamadas.

Texto

Descripción generada automáticamente

* **NoClassDefFoundError:**

Este error ocurre cuando la JVM intenta cargar una clase pero no puede encontrar su definición en tiempo de ejecución. Puede ocurrir si la clase estaba disponible durante la compilación pero no está disponible durante la ejecución, debido a problemas de configuración del classpath o a la ausencia de dependencias en tiempo de ejecución.

Pantalla negra con letras blancas

Descripción generada automáticamente

* **NoSuchFieldError:**

Se produce cuando se intenta acceder a un campo (variable de instancia) en una clase que no existe en tiempo de ejecución. Esto puede ocurrir, por ejemplo, si la clase fue compilada con una versión de una biblioteca que contiene un campo, pero se ejecuta con una versión diferente de esa biblioteca que no contiene ese campo.

* **NoSuchMethodError:**

Este error ocurre cuando se intenta invocar un método que no existe en una clase en tiempo de ejecución. Puede ocurrir si la clase fue compilada con una versión de una biblioteca que contiene un método, pero se ejecuta con una versión diferente de esa biblioteca que no contiene ese método.

* **AssertionError:**

Se produce cuando una declaración de aserción falla. Las aserciones son declaraciones que verifican suposiciones en el código y suelen utilizarse para detectar errores durante el desarrollo y pruebas.

Texto

Descripción generada automáticamente

* **LinkageError:**

Este error se produce cuando hay un error en el proceso de enlace de una clase o una biblioteca. Puede ocurrir, por ejemplo, si hay conflictos de versión entre clases o bibliotecas.

* **InternalError:**

Este error indica un problema interno en la JVM o en sus componentes subyacentes. Puede ocurrir debido a un error en la implementación de la JVM o a problemas graves de configuración.

* **UnknownError:**

Se produce cuando ocurre un error que no se puede clasificar en ninguna otra categoría de error.

* **UnsatisfiedLinkError:**

Este error se produce cuando la JVM no puede encontrar una biblioteca nativa (generalmente escrita en C o C++) que es requerida por la aplicación. Puede ocurrir si la biblioteca nativa no está disponible en el sistema o si hay problemas de configuración en la carga de la biblioteca.

Texto

Descripción generada automáticamente

* **VirtualMachineError:**

Esta clase es la superclase de todos los errores lanzados por la JVM para indicar que algo va mal a nivel de la máquina virtual misma.