|  |
| --- |
| **TECNICATURA UNIVERSITARIA EN PROGRAMACION**  **Laboratorio de Computación II - Ciclo lectivo 2025** |

|  |  |
| --- | --- |
| **Características del Práctico : Excepciones en Java** | |
| **Unidad** | **3** |
| **Tema** | **Excepciones en Java** |
| **Resultados de Aprendizajes** | RA1: Domina el sistema de excepciones Java, lanzamiento captura,  RA2: Domina la creación de tipos propios de excepciones. |
| **Objetivo** | El objetivo de este trabajo práctico es que los estudiantes adquieran un entendimiento del sistema de excepciones provisto por Java y puedan aplicarlo a sus propios desarrollos. |
| **Requisitos técnicos** | Codificar una solución básica en Java para probar el sistema de logging. |
| **Fecha de inicio** | Marzo de 2025 - |
| **Fecha de entrega:** | Marzo de 2025 |
| **Modalidad de entrega :** | Adjuntar el trabajo en formato Word al link correspondiente en el aula virtual. |
| **Comisiones** | **Comisiones: Mañana y tarde** |
| **Modalidad del Trabajo Práctico** | Desarrollo Grupal – Entrega en aula virtual |
|  | |
| **Conclusiones:**  Los estudiantes deben resumir lo que aprendieron durante la realización del trabajo práctico y destacar cómo el manejo de excepciones en Java permite un mejor seguimiento de la ejecución de la aplicación, como así también el manejo de errores y situaciones anómalas. | |
| Forma de Presentación | El trabajo debe presentarse en el link correspondiente al trabajo práctico alojado en el aula virtual. |
| Tipo de Evaluación | Formativa - Grupal |
| **Modalidad de Evaluación** | |  |  |  |  | | --- | --- | --- | --- | |  | **100** | **>70%** | **<50 %** | | **Lanzamiento de Excepciones** |  |  |  | | **Manejo de Excepciones** |  |  |  | | **Creación de tipos propios de excepciones** |  |  |  | | **RESULTADOS** | **Excelente** | **Satisfactorio** | **No Satisfactorio** |   **Se establece una sumatoria de cada ítem y se determina el porcentaje individual de cada uno. Posteriormente se establece la sumatoria de todos los ítems y se lo divide por la cantidad de los mismos para determinar en cuál de los valores de la escala se encuadra.**   |  |  |  |  | | --- | --- | --- | --- | |  |  |  |  | |  |  |  |  | |  |  |  |  | |  |  |  |  | |  |  |  |  | |  |  |  |  | |
| Bibliografía | Effective Java 3rd Edición. [Joshua Bloch](https://www.amazon.com/-/es/Joshua-Bloch/e/B001CDCVUG/ref=dp_byline_cont_book_1). (2017)  Java: The Complete Reference, Twelfth Edition 12th Edición.  [Herbert Schildt](https://www.amazon.com/-/es/Herbert-Schildt/e/B001H6PSMG/ref=dp_byline_cont_book_1) (2021) |
| Profesores responsables del TP. | |  |  |  |  | | --- | --- | --- | --- | | Apellido y nombre | | Apellido y nombre | | | Julio Monetti | | Claudia Naveda | | |  |  | |  | |  |  | |

DESARROLLO

**PARTE A**

Ejercicio

Responda.

* 1. Cuál es el nombre en Java de la clase que define las excepciones ?

El nombre de la clase es Throwable

* 1. Cuál es el nombre en Java de la clase que representa las excepciones que se producen al invocar un método que pertenece a un objeto nulo

NullPointerExeption

* 1. Qué información aporta el método *printStackTrace* en el manejo de excepciones ?

Específicamente este método imprime por consola:

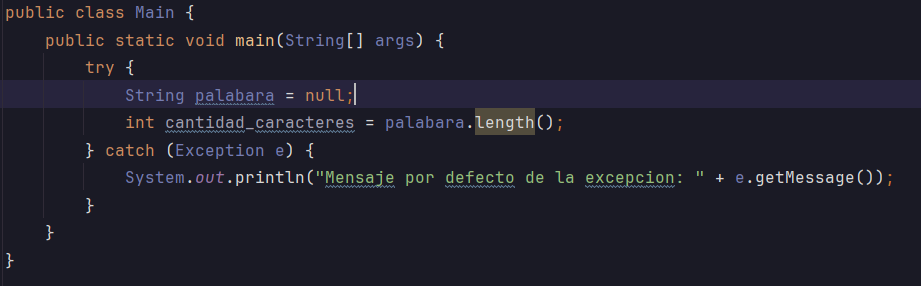
* el tipo de excepción (por ejemplo, NullPointerExeption)
* el mensaje asociado a la excepción, si lo hay
* la pila de llamadas (stack trace). Es la secuencia de métodos que se estaban ejecutando cuando ocirrio la excepcion

**Ejercicio**

Como obtiene el mensaje perteneciente a una excepción ?. De un ejemplo de código donde se utilice este dato.

Para obtener el mensaje perteneciente a una excepción se utilixa el método getMessage(). El cual devuelve el mendaje que la excepción genera por defecto

Por ejemplo:



**Ejercicio**

Observe el siguiente fragmento de código

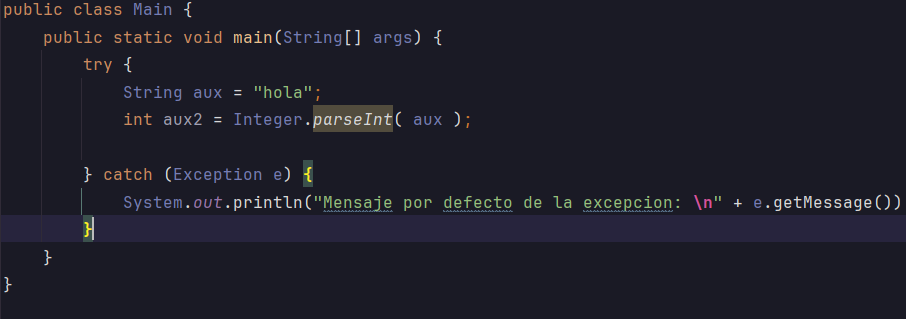
*String aux = “hola”;*

*int aux2 = Integer.parseInt( aux );*

Qué sucede al ejecutar el mismo ?

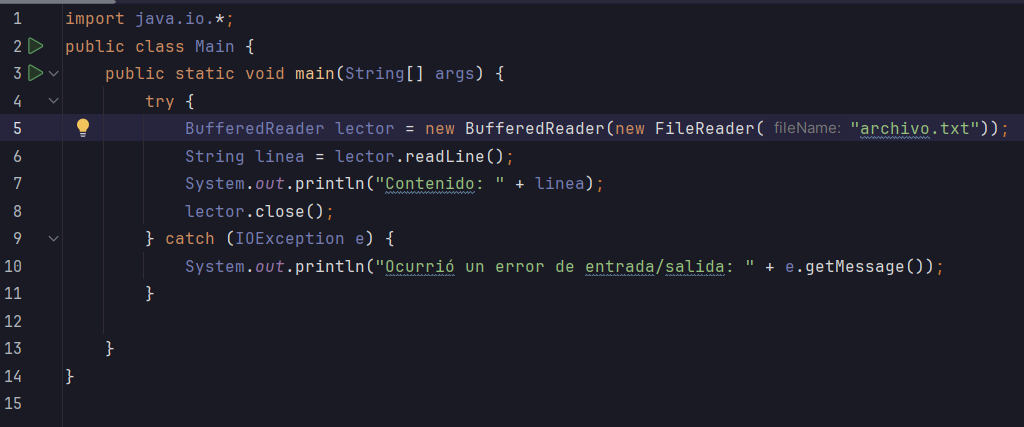
Lo que sucede es que larga una excepción NumberFormatExeption con un mensaje “For input String: ‘hola’ ” que nos indica que no podemos castear la variable aux de String a int

Modifíquelo para realizar manejo de excepciones.



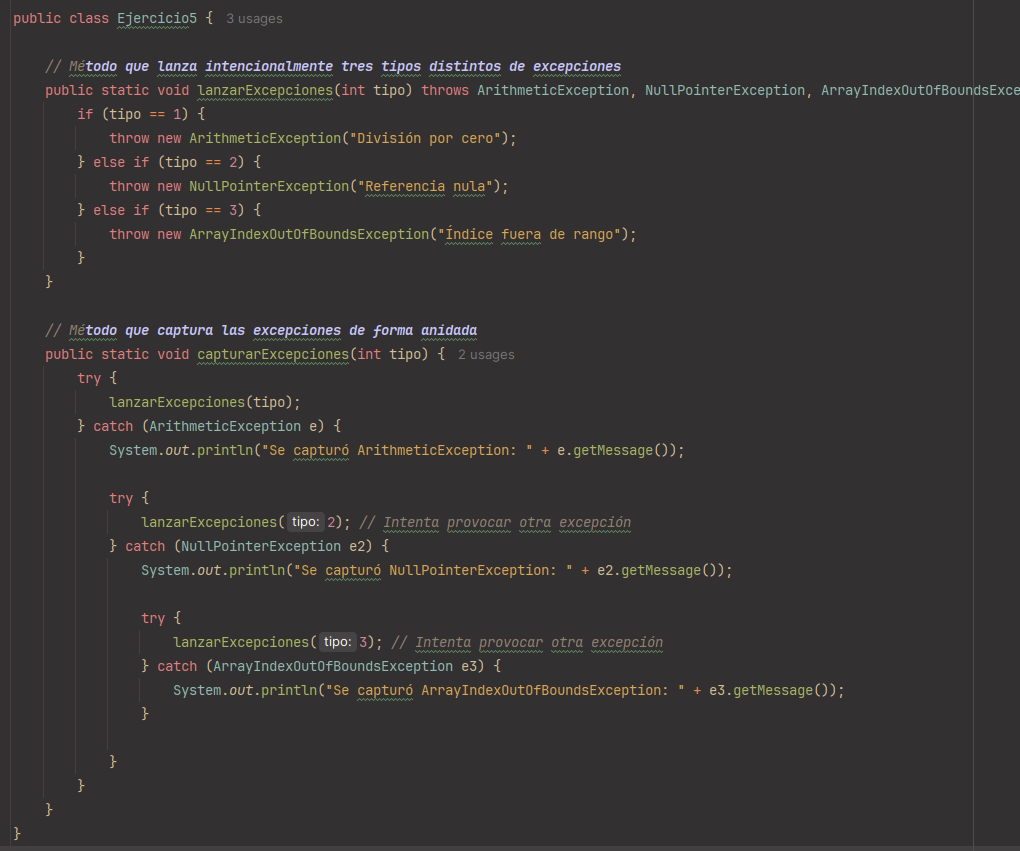
**Ejercicio**

Cree un programa en Java que produzca una IOException. Lance y maneje la excepción adecuadamente.



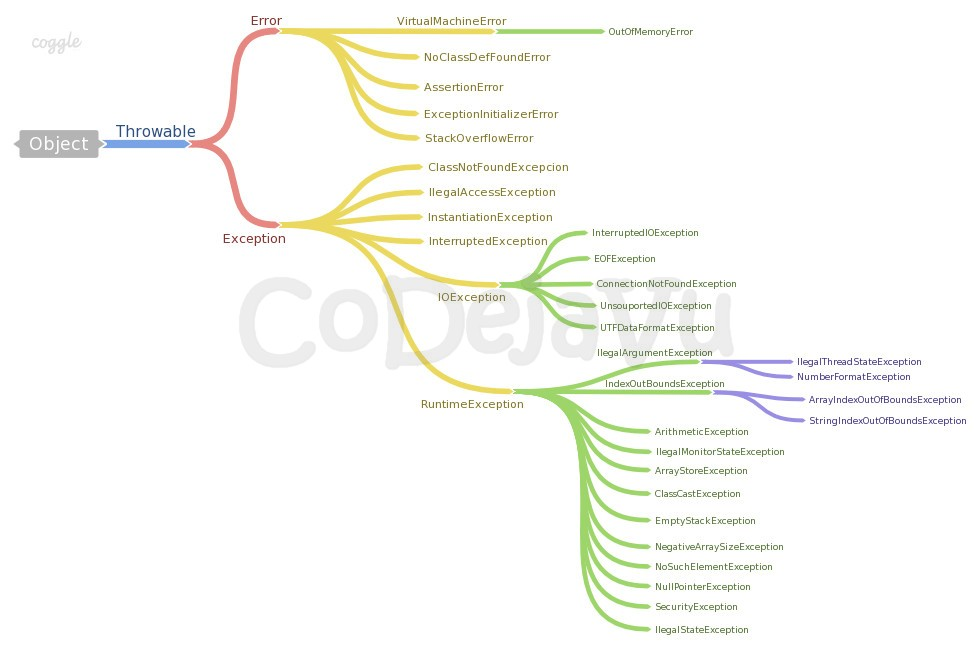
**Ejercicio**

Cree un método que capture al menos tres tipos de excepciones en forma anidada. (Cree también el método que las lanza).



**Ejercicio**

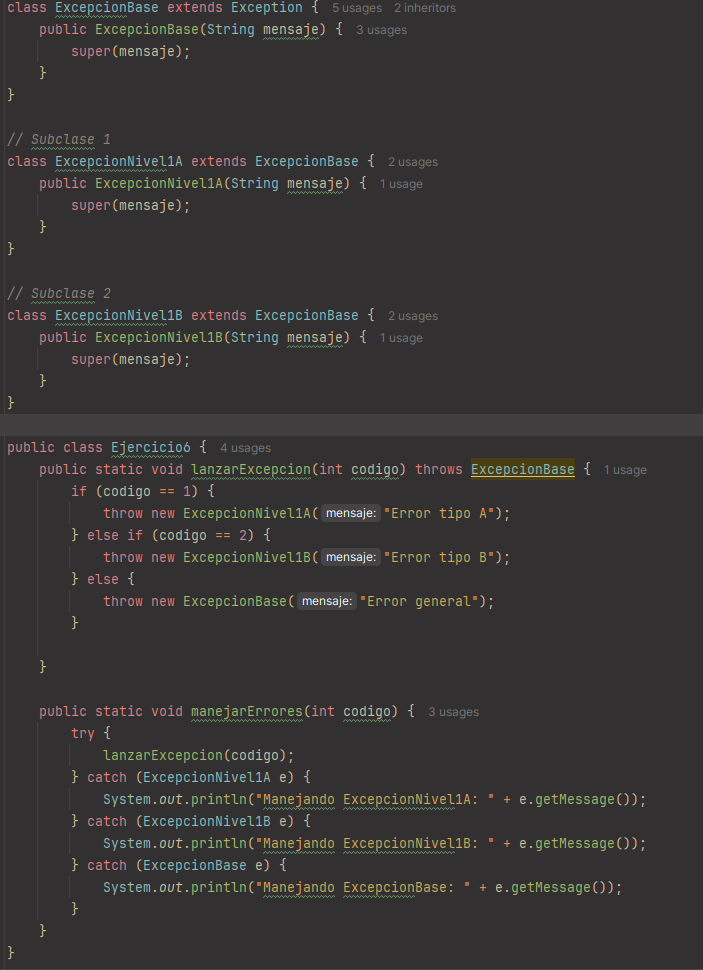
Investigue en Internet y copie un árbol de excepciones de Java. Documente al menos cuatro de ellas.



* ArithmeticException:  
   Se lanza cuando ocurre un error aritmético, como dividir un número entero por cero.
* EOFException (End of File Exception):  
   Se lanza al intentar leer más datos de un flujo de entrada (como un archivo) cuando ya se ha alcanzado el final del archivo.
* ClassNotFoundException:  
   Se lanza cuando el código intenta cargar una clase mediante su nombre (por ejemplo, con Class.forName("...")), pero no se encuentra en el classpath.
* OutOfMemoryError (no existe OutOfMemoryException como tal):  
   Se lanza cuando la máquina virtual de Java (JVM) no puede asignar más memoria, por ejemplo, al crear nuevos objetos.

**Ejercicio**

Cree una jerarquía de excepciones de dos niveles. De ejemplos de usos de la misma.



**Ejercicio**

Qué muestra el siguiente programa ? (identifique la salida sin ejecutarlo)

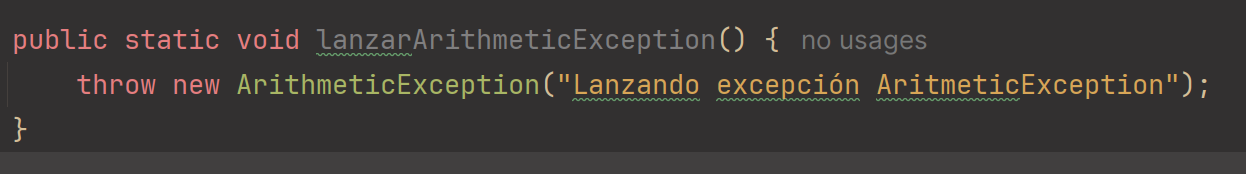
|  |
| --- |
| p*ublic class Main {*  *public static int devuelveNumero(int num) {*  *try {*  *if (num % 2 == 0) {*  *throw new Exception("Lanzando excepcion");*  *}*  *return 1;*  *} catch (Exception ex) {*  *return 2;*  *} finally {*  *return 3;*  *}*  *}*    *public static void main(String[] args) {*  *System.out.println(devuelveNumero(1));*  *}*    *}* |

El código siempre va a imprimir en la consola el número 3 debido a que el bloque de código contenido en *finally* siempre se ejecuta antes de que el método termine, inclusive si hay retornos antes en *try* o *catch*.

**PARTE B**

**Ejercicio**

Cree un programa que lance una ArithmeticException.



**Ejercicio**

Codifique la siguiente aplicación



El método *acelera* lanza una excepción de tipo **miException** si alguno de los vehículos intenta acelerar a más de 120km/h. Agregar a esta clase la funcionalidad necesaria para mostrar por pantalla el nombre del objeto/clase que la produce (Auto o Camion).

Al crear un chofer, se debe lanzar una excepción **matriculaVencidaException** si la matrícula está vencida.

*(Archivo .java adjunto)*

**Ejercicio**

Cree un programa que genere un número aleatorio e indique si el número generado es par o impar. El programa utilizará para ello el lanzamiento de una excepción.



**Ejercicio**

Cree un programa que lance una excepción a través de una pila de llamada de métodos de longitud cuatro. El último método debe manejar la excepción.

