Universidad Autónoma del Cauca

Taller entregable

DOCENTE : Cristhian Cañar

ASIGNATURA : Programación 2

ALUMNO : David López

GRADO : 5to semestre, ING Software

2025

Código:

```java
public class Main {
    // Método para generar StackOverflowError (recursión infinita)
    static void metodoRecursivo() {  2 usages
        metodoRecursivo();
    }
    public static void main(String[] args) {
        // Manejo de una Excepción (Checked Exception)
        try {
            FileReader file = new FileReader( fileName: "archivo_no_existente.txt"); // Generará IOException
        } catch (IOException e) {
            System.out.println("¡Excepción capturada! Archivo no encontrado: " + e);
        }
        // Manejo de una Excepción (Unchecked Exception)
        try {
            int resultado = 10 / 0; // División por cero genera ArithmeticException
        } catch (ArithmeticException e) {
            System.out.println("¡Excepción capturada! División por cero: " + e);
        }
        // Generar un Error (OutOfMemoryError)
        try {
            List<int[]> lista = new ArrayList<>();
            while (true) {
                lista.add(new int[1000000]); // Consumirá toda la memoria heap
            }
        } catch (OutOfMemoryError e) {
            System.out.println("¡Error crítico! Memoria insuficiente: " + e);
        }
        // Generar un Error (StackOverflowError)
        try {
            metodoRecursivo();
        } catch (StackOverflowError e) {
            System.out.println("¡Error crítico! Desbordamiento de pila: " + e);
        }
    }
```

```
¡Excepción capturada! Archivo no encontrado: java.io.FileNotFoundException: archivo_no_existente.txt (El sistema no puede encontrar el archivo especificado)
¡Excepción capturada! División por cero: java.lang.ArithmeticException: / by zero
¡Error crítico! Memoria insuficiente: java.lang.OutOfMemoryError: Java heap space
¡Error crítico! Desbordamiento de pila: java.lang.StackOverflowError
```

Análisis y respuestas a las preguntas planteadas:

1. ¿Cuál es la diferencia entre Exception y Error en Java?

   En java, un error es un problema en un programa que impide que éste complete su tarea. En comparación, una excepción es una condición que interrumpe el flujo normal del programa. En palabras sencillas, el error es un problema crítico que una

aplicación normal no debe captar, mientras que una excepción es una condición que un programa debe captar (datacamp.com).

2. ¿Por qué IOException es una Checked Exception y ArithmeticException una Unchecked Exception?

   IOException es una Checked Exception ya que trata sobre problemas que se pueden crear a través del código por problemas externos a él, ya sea de entrada o salida; Mientras que, AritmethicException es una Unchecked Exception ya que muestra los errores internos al código, como es nuestro caso al intentar dividir entre 0.

3. ¿Es recomendable manejar errores (Error) con try-catch? Justifica tu respuesta.

   En realidad, no, ya que, como su nombre lo indica, son errores, errores que en un desarrollo de software no puede haber, ya que se busca la optimización de código y el funcionamiento claro y efectivo del mismo.

4. ¿En qué orden se generaron los errores y excepciones?

   En el código primero se ejecuta la excepción: IOException, con el archivo .txt que no existe, luego la excepción: ArithmeticException, con la división entre 0, enseguida genera un error de tipo: OutOfMemoryError, tratando de llenar un arreglo de 1000000 posiciones, y por último genera el error: StackOverflowError, llamando a una función recursiva sin condición de salida.

5. ¿Cómo podrías evitar el OutOfMemoryError en este código?

Colocando un límite al número de posciones que se requieren de arreglos creados, otra forma podría ser evitando el ciclo infinito ya que esto evitaría el problema, conesto podría solucionarse el **Error**.

6. ¿Cómo evitarías el StackOverflowError sin modificar el try-catch?

   Como es una llamada recursiva (que se llama a si misma las veces que sea requerida) una de las soluciones puede ser colocar un caso base (tope de iteraciones de la recursividad) para que así se detenga al llegar a cierto número solicitado.

Código solucionado:

```java
public class Solved {
    public static void main(String[] args) {

        // Intentamos trabajar con un archivo que puede no existir
        try {
            File archivo = new File( pathname: "archivo_no_existente.txt");

            // Revisamos si el archivo realmente está ahí
            if (archivo.exists()) {
                FileReader lector = new FileReader(archivo);

            } else {
                System.out.println("Ups, the file you're trying to in isn't available.");
            }

        } catch (IOException e) {
            // Capturamos cualquier problema al manejar el archivo (por ejemplo, si no se puede abrir o que simplemente no exista)
            System.out.println("File error occurred: " + e);
        }

        // Ahora vamos a manejar una posible división por cero de forma segura
        try {
            int divisor = 2; // Cambiamos el cero por otro número para evitar el caos

            int resultado = 10 / divisor;
            System.out.println("All good, Result: " + resultado);

        } catch (ArithmeticException e) {
            // Esto atraparía un error si intentáramos dividir por cero (aunque aquí no pasa)
            System.out.println("Math error: " + e);

        }
```

```java
        }

        // Simulamos una carga de memoria pero controlada para evitar un OutOfMemoryError

        List<int[]> lista = new ArrayList<>();

        for (int i = 0; i < 5; i++) { // Solo creamos 5 arrays grandes
            lista.add(new int[1000000]); // Cada array tiene 1000000 de posiciones

        }

        // Llamamos a un método recursivo adaptado
        stopResource( contador: 0);

    }

    // Método recursivo con tope para evitar StackOverflowError
    public static void stopResource(int contador) {  2 usages

        if (contador >= 1000) return; // Condición para detener la recursión antes del desastre
        stopResource( contador: contador + 1);

    }
}
```

```
Ups, the file you're trying to in isn't available.
All good, Result: 5
```