Excepciones

Algoritmos y Programación 2
Angela Villota Gómez 2023-2

Agenda

01

Introducción

Excepciones

02

Conceptos

Tipos y el requerimiento de captura o especificación

03

Try-Catch, Throw

Sintaxis - Ejemplos

04

Ejercicios

Un sistema de matrículas

01

Introducción

Tipos de errores en programación

Errores de sintaxis: Estos errores ocurren cuando el código no cumple con las reglas gramaticales del lenguaje de programación. El compilador los detecta y muestra mensajes de error.

Errores de tiempo de ejecución: Estos son errores que ocurren durante la ejecución del programa y pueden ser causados por problemas como la división por cero, el acceso a un índice no válido en un arreglo, etc.

Errores lógicos: Estos son errores en la lógica del programa que no generan errores en tiempo de compilación ni en tiempo de ejecución, pero hacen que el programa no funcione como se espera.

¿Qué es una excepción?

Las excepciones son eventos o condiciones anormales que ocurren durante la ejecución de un programa y que interrumpen el flujo normal de ejecución.

Definir un conjunto de excepciones en el momento del diseño permite manejar situaciones inesperadas o errores que pueden surgir durante la ejecución de un programa.

¿Qué es una excepción?

¿Para qué?

- Evitar que los errores inesperados causen la terminación abrupta de un programa.
- Proporcionar una forma elegante de lidiar con problemas y comunicarlos al usuario o al desarrollador.
- Mejorar la robustez y la calidad de un programa al anticipar y gestionar posibles problemas.

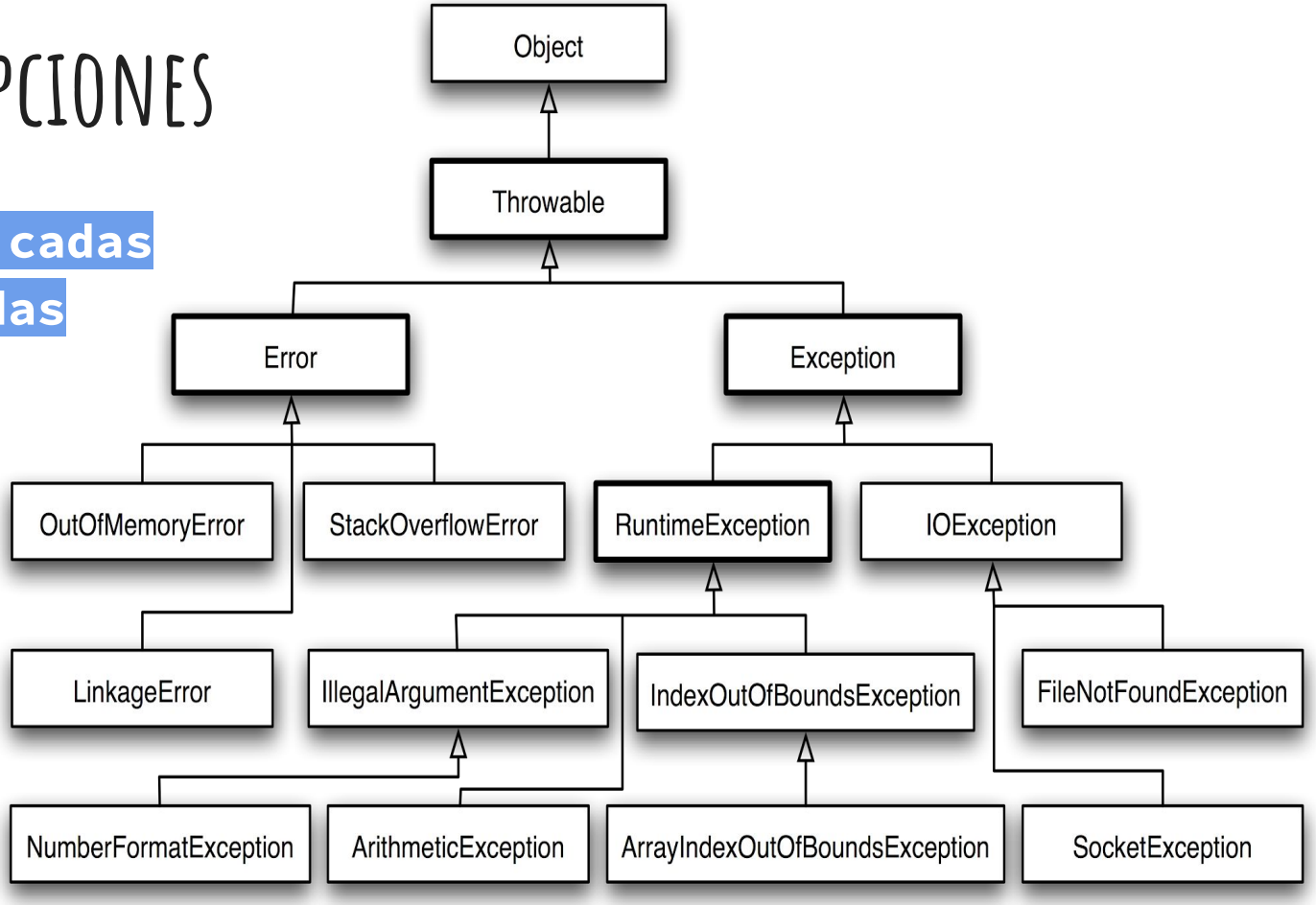
Pregunta

02

CONCEPTOS

Tipos de excepciones

1. **No Verificadas**
2. **Verificadas**
3. **Errores**



Tipos de excepciones

No Verificadas

- son excepciones que el compilador no requieren ser manejadas de manera explícita.
- Estas son condiciones excepcionales que son internas a la aplicación y que la aplicación generalmente no puede anticipar ni recuperar. Por lo general, indican errores de programación, como errores lógicos o uso incorrecto de una API.
- **No se manejan porque tiene más sentido eliminar el error que causó que ocurriera la excepción.**

  Ejemplos de excepciones no verificadas incluyen NullPointerException y ArrayIndexOutOfBoundsException

Tipos de excepciones

Verificadas

- Son excepciones que el compilador requiere que sean manejadas explícitamente en el código utilizando bloques try-catch o declarándolas en la firma del método.
- **¿Y por qué?** Estas son condiciones excepcionales que una aplicación bien escrita debería anticipar y recuperarse.

  Ejemplos de excepciones verificadas incluyen IOException y SQLException.

Tipos de excepciones

Errores

- Son condiciones excepcionales que son externas a la aplicación y que la aplicación generalmente no puede anticipar ni recuperar.

  Por ejemplo, supongamos que una aplicación abre con éxito un archivo de entrada, pero no puede leer el archivo debido a un mal funcionamiento del hardware o del sistema. La lectura fallida generará java.io.IOError.

Ejercicio

Descargue el archivo [Link]

Responda a las preguntas

1. Qué hace el programa?
2. Qué valores de entrada causan los errores?
3. Para cada una de las apariciones de la instrucción try-catch, ¿para qué sirven?

```java
package ui;

import java.util.InputMismatchException;

public class Main {

    private Scanner reader;
    public Main() {
        reader = new Scanner(System.in);

    }

    public static void main(String[] args) {
        Main m = new Main();

        // la excepción se lanza al leer los valores del reader
        int a = m.reader.nextInt();
        int b = m.reader.nextInt();
        try{
            int c = m.sum(a, b);
            System.out.print(c);
            m.readInfo();
        }
        catch(InputMismatchException ie){
            System.out.print("el valor debe ser un entero!");
        }
        //
        // la excepción se lanza al hacer la operación
        int d = m.div(a, b);
        System.out.print(d);

    }

    public int sum(int a, int b){
        return a + b;
    }

```

El requerimiento de captura o especificación

¿Qué es?

Un requerimiento para que el programa sea válido y se pueda compilar.

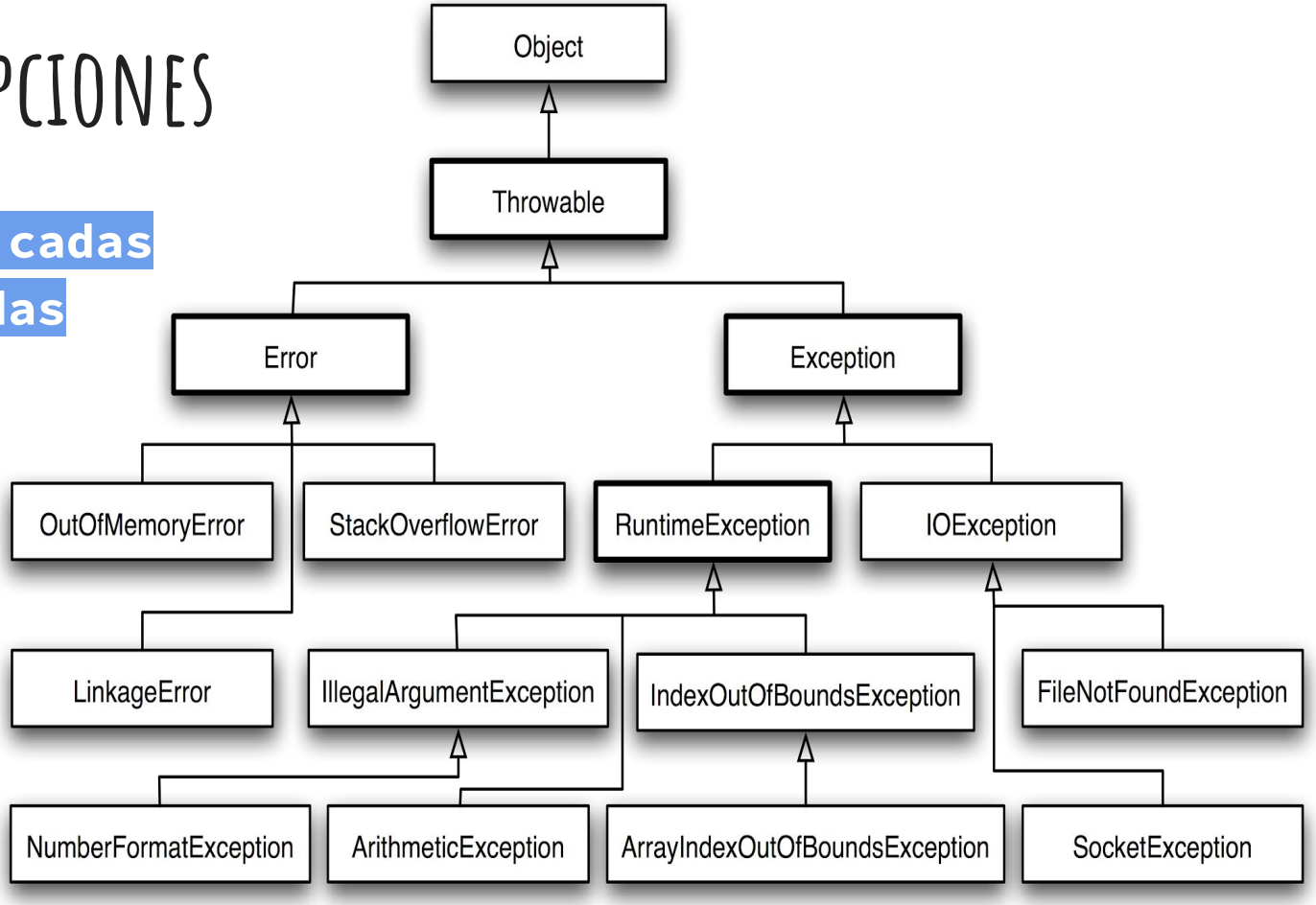
¿En qué consiste?

el código que podría generar ciertas excepciones debe estar contenido en cualquiera de los siguientes:

1. Una declaración try que capture la excepción. El bloque try debe proporcionar un manejador para la excepción
2. Un método que especifique que puede generar la excepción. El método debe proporcionar una cláusula throws que enumere la excepción.

Tipos de excepciones

1. **No Verificadas**
2. **Verificadas**
3. **Errores**



La clase Throwable

- Es la clase base para todas las excepciones en Java. Todas las excepciones en Java heredan de Throwable, ya sea directa o indirectamente.
- Throwable proporciona métodos esenciales para el manejo de excepciones (como getMessage() y printStackTrace()) que se pueden utilizar para obtener información sobre la excepción y su seguimiento de pila.
- Es posible crear excepciones propias para un problema creando clases que hereden de Exception o RuntimeException, dependiendo de si se desea que sean excepciones verificadas o no verificadas.

03

Try-catch Throw

El bloque try-catch

¿Qué es?

los bloques try-catch son una estructura fundamental en Java para manejar excepciones. Estos bloques permiten al programador capturar y manejar excepciones de manera controlada.

Opciones

un bloque try-catch, puede tener varios bloques catch para manejar diferentes tipos de excepciones.

- Los bloques catch se evalúan secuencialmente, y solo se ejecutará el primer bloque cuyo tipo de excepción coincida con la excepción generada.
- Es posible usar un bloque finally opcional que se ejecuta siempre, ya sea que se produzca una excepción o no.

Try-catch Sintaxis (1)

```
try {
        // Código que puede generar
           una excepción
} catch (TipoDeExcepcion e) {
        // Código para manejar la excepción
}
```

Try-catch Sintaxis (2)

```
try {
    // Código que puede generar una excepción
} catch (TipoDeExcepcion1 e) {
    // Manejar TipoDeExcepcion1
} catch (TipoDeExcepcion2 e) {
    // Manejar TipoDeExcepcion2
} catch (TipoDeExcepcion3 e) {
    // Manejar TipoDeExcepcion3
} finally {
    // Código que se ejecuta siempre
}
```

es posible usar un bloque finally opcional que se ejecuta siempre, ya sea que se produzca una excepción o no.

EJEMPLO

```java
public class Calculator {

    public int dividir(int numerador, int denominador) {
        return numerador / denominador;
    }
```

```java
public void dividirMultiplesEx() {
    Calculator ca=null;
    try {
        // Intentamos dividir por cero
        int resultado = ca.dividir(10, 0);
        System.out.println("Resultado: " + resultado);
    } catch (ArithmeticException e) {
        // Manejo de la excepción de división por cero
        System.err.println("Error: No se puede dividir por cero.");
    } catch (NullPointerException e) {
        // Manejo de la excepción de objeto no inicializado
        System.err.println("Error: el objeto no ha sido inicializado.");
    } catch (Exception e) {
        // Manejo de cualquier otra excepción
        System.err.println("Error desconocido: " + e.getMessage());
    }
}
```

Lanzando excepciones (Throw)

throw

- La instrucción throw se utiliza para lanzar excepciones manualmente.
- Es útil cuando se desea señalar una condición excepcional en el código

throw vs throws

La instrucción throws, que se utiliza en la firma del método para indicar que el método puede lanzar una excepción, **solo cuando se lanza una excepción verificada.**

Throw -sintaxis - ejemplo

```java
//Sin declaración de excepción
public  double calcularRaizCuadradaV1(double numero) {
    if (numero < 0) {
        throw new IllegalArgumentException(
                "No se puede calcular la raíz cuadrada de un número negativo.");
    }
    return Math.sqrt(numero);
}

//Con declaración de excepción
public  double calcularRaizCuadradaV2(double numero) throws Exception {
    if (numero < 0) {
        throw new Exception(""
                + "No se puede calcular la raíz cuadrada de un número negativo.");
    }
    return Math.sqrt(numero);
}
```

Throw -sintaxis - ejemplo

```
public void raiz() {
    Calculator ca=new Calculator();
    System.out.println(ca.calcularRaizCuadradaV1(-2));

    try {
        System.out.println(ca.calcularRaizCuadradaV2(-2));
    } catch (Exception e) {
            e.printStackTrace();
    }
}
```

04

Ejercicios

Trabajar en la hoja de trabajo[link]

Preguntas?