Excepciones

Emmanuel Cruz Hernández
emmanuel_cruzh@ciencias.unam.mx

19 de enero de 2021

Contenido

1. ¿Qué es una excepción?
2. Clase Exception
3. Bloque try... catch
4. Creación de una propia excepción
5. Declaración throws
6. Tipos de excepciones
7. Bibliografía

¿Qué es una excepción?

Una excepción es un mecanismo muy poderoso que tiene Java para detectar errores de ejecución de todo tipo. [1]
También se puede ver como un evento *excepcional* que ocurre cuando hay una falla en la ejecución de un programa o hay un error.

¿Qué pasa cuando ocurre una excepción?

Cuando un programa se encuentra con un error, este termina abruptamente, se pierde el estado de las variables, incluyendo todos los datos que no han sido guardados en un archivo.

Clase Exception

Las clases de excepción permiten que las excepciones sean representadas como objetos.
Un objeto de tipo *Exception* almacena la información acerca de la causa de la excepción y puede pasarse como un valor a un manejador de excepciones que se implementa en un bloque que permite **cachar** el error. [2]

Excepciones de Java

- **IllegalArgumentException**: Esta excepción es lanzada cuando métodos de Java o definidos por el usuario detecta que un argumento no es como se esperaba. Por ejemplo, si se desea sacar la raíz cuadrada de un número negativo.
- **IndexOutOfBoundsException**: Es lanzado cuando se intenta usar un elemento de un arreglo que no existe, porque el índice dado no está en el rango dado para los índices del arreglo.

Excepciones de Java

- **NegativeArraySizeException**: Se lanza si es que se intenta crear un arreglo con un número negativo de elementos.
- **NullPointerException**: Se lanza si se trata de usar al objeto referido por una referencia nula.

Bloque try... catch

Las palabras reservadas **try** y **catch** proveen de un núcleo sintáctico para manejar excepciones.
La palabra reservada **finally** provee de un mecanismo para garantizar que un bloque de código se ejecuta sin importar que otra cosa suceda.

Sintaxis del bloque try... catch

try{
    Secuencia de enunciados
}
catch(*Exception1* identificador1){
    Código a ejecutar en caso de ocurrir Exception1
}
catch(*Exception2* identificador2){
    Código a ejecutar en caso de ocurrir Exception2
}
...

Sintaxis del bloque try... catch... finally

try{
    Secuencia de enunciados
}
catch(*Exception1* identificador1){
    Código a ejecutar en caso de ocurrir Exception1
}
catch(*Exception2* identificador2){
    Código a ejecutar en caso de ocurrir Exception2
}
...
finally{
    Código que se ejecuta sin importar qué suceda
}

Bloque catch

Normalmente, el programador desea utilizar un bloque **catch** para recobrar el estado del programa, y dejarlo proseguir sin terminar.
El bloque **catch** puede usar cualquier variable y llamar a cualquier método en el entorno, y puede desplegar un mensaje informando al usuario que ha habido un problema.

Bloque finally

Un bloque **finally** siempre se ejecuta, sin importar qué pase con el bloque **try**, aún si se arroja otra excepción o se ejecuta un enunciado **return**. Esto provee de un lugar para poner enunciados cuya ejecución está garantizada.

Creación de una Excepción

Una excepción nueva puede ser definida por el programador implementando una clase que herede de la superclase **Exception**.

Sintaxis de definición de excepción

Dado que se debe heredar de **Exception**, hacemos uso de la palabra reservada **extends**.

public class *NombreExcepcion* **extends Exception**{

    ...

}

Método constructor

El método constructor de la clase **Exception** recibe un String, que representa el mensaje de la causa del error.

```
public NombreExcepcion(String mensaje) {
    super(mensaje);
}
```

Otra forma de declarar el método constructor

Sin embargo se le puede pasar por default un mensaje. De esta forma no sería necesario pasarlo como parámetro.

```
public NombreExcepcion() {
    super("Mensaje de exepcion");
}
```

Declaración throws

Se requiere hacer una declaración **throws** para cualquier método que arroje o propague una excepción que deba ser tratada explícitamente.

Cualquier método estático, método de instancia, método abstracto o declaración de constructor puede incluir una declaración **throws** en seguida de la lista de parámetros y antes del cuerpo del método

Sintaxis de la declaración throws

public *tipoDeRegreso* metodo(*parámetros*) **throws** Exception1,
Exception2, Exception3, ... {
    ...
}

Lanzar una excepción

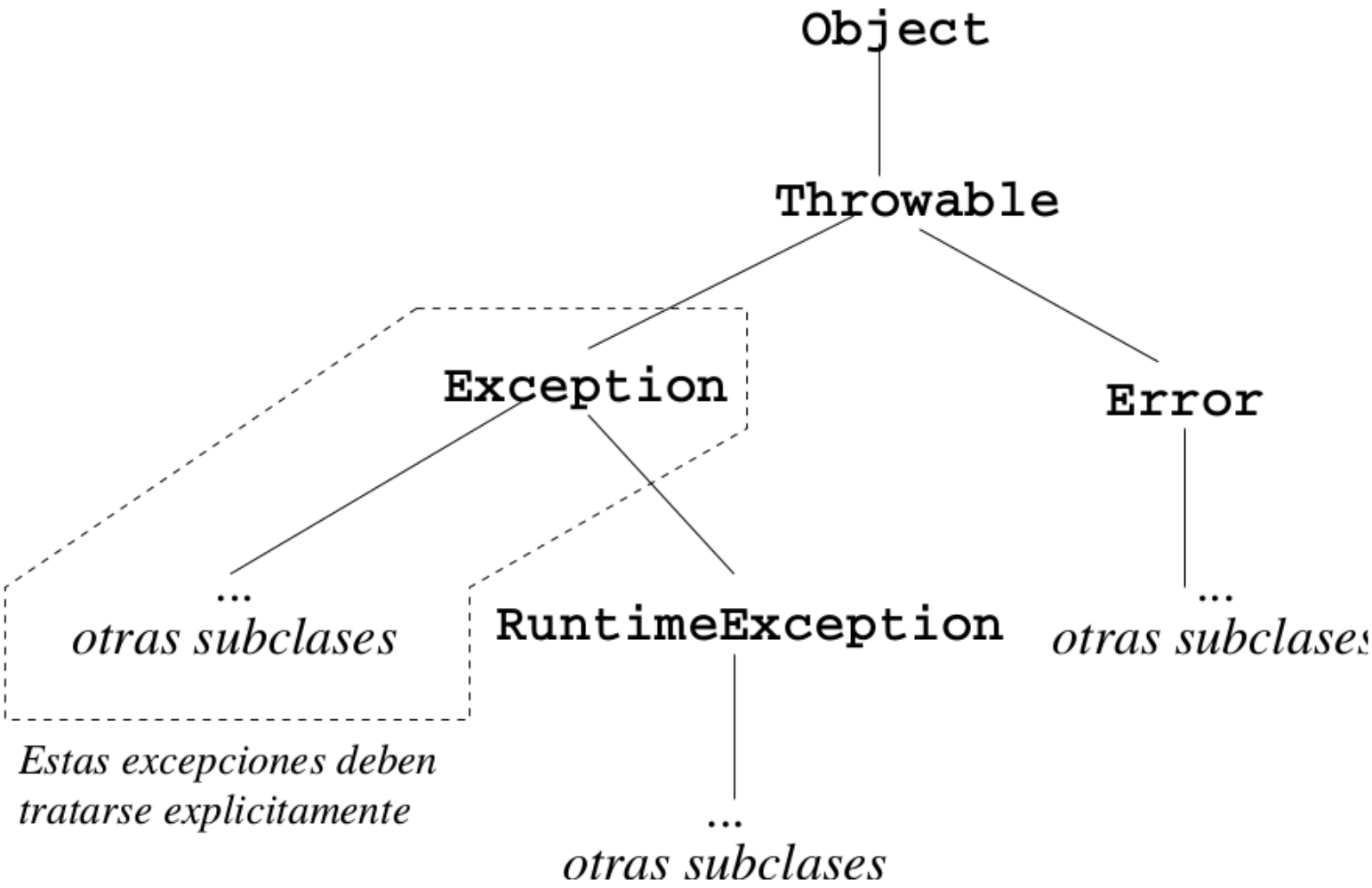
Un enunciado **throw** permite que una excepción sea arrojada. Una instancia de una excepción se crea usando una expresión new como parte del enunciado **throw**, como por ejemplo:

```
throw new Exception("Algo ha fallado");
```

Tipos de excepciones

- **RuntimeException**: representan serios errores causa- dos en general por la Máquina Virtual de Java.
  Las excepciones de error indican una falla en el sistema fuera del control del programa. Un programa en Java no necesita explícitamente tratar con estas excepciones.
- **Exception y sus subclases**: representan errores causados por omisiones y fallas lógicas iniciadas por el programador. Una excepción representada por una de estas clases debe ser explícitamente tratada por el programa en Java.

Tipos de excepciones



Bibliografía

Elisa Viso G. Canek Peláez V.
Introducción a las Ciencias de la Computación con Java.
Segunda edition, 2012.

Jorge L. Ortega Arjona.
Notas de Introducción al Lenguaje de Programación Java.
2004.