ULP Virtual

Tecnicatura Universitaria en Desarrollo de Software

Programación II

Guía 6

Manejo de excepciones

Universidad de LA PUNTA

GOBIERNO DE SAN LUIS

GUÍA DE MANEJO DE EXCEPCIONES

EXCEPCIONES

El término excepción es una abreviación de la frase “Evento Excepcional”. Una excepción es un evento que ocurre durante la ejecución de un programa que interrumpe el flujo normal de las instrucciones del programa.

Existen muchas clases de errores que pueden provocar una excepción, desde un desbordamiento de memoria o un disco duro estropeado hasta un intento de dividir por cero o intentar acceder a un vector fuera de sus límites. Cuando esto ocurre, la máquina virtual Java crea un objeto de la clase Exception o Error y se notifica el hecho al sistema de ejecución. Se dice, que se ha lanzado una excepción (“Throwing Exception”). Luego, el objeto, llamado excepción, contiene información sobre el error, incluyendo su tipo y el estado del programa cuando el error ocurrió.

Después de que un método lanza una excepción, el sistema, en tiempo de ejecución, intenta encontrar algo que maneje esa excepción. El conjunto de posibles “algo” para manejar la excepción es la lista ordenada de los métodos que habían sido llamados hasta llegar al método que produjo el error. Esta lista de métodos se conoce como pila de llamadas. Luego, el sistema en tiempo de ejecución busca en la pila de llamadas el método que contenga un bloque de código que pueda manejar la excepción. Este bloque de código es llamado manejador de excepciones.

Concretamente, una excepción en java es un objeto que modela un evento excepcional, el cual no debería haber ocurrido. Como observamos anteriormente, al ocurrir estos tipos de eventos la máquina virtual no debe continuar con la ejecución normal del programa. Es evidente que las excepciones son objetos especiales, son objetos con la capacidad de cambiar el flujo normal de ejecución. Cuando se detecta un error, una excepción debe ser lanzada.

Ejemplos de situaciones que provocan una excepción:

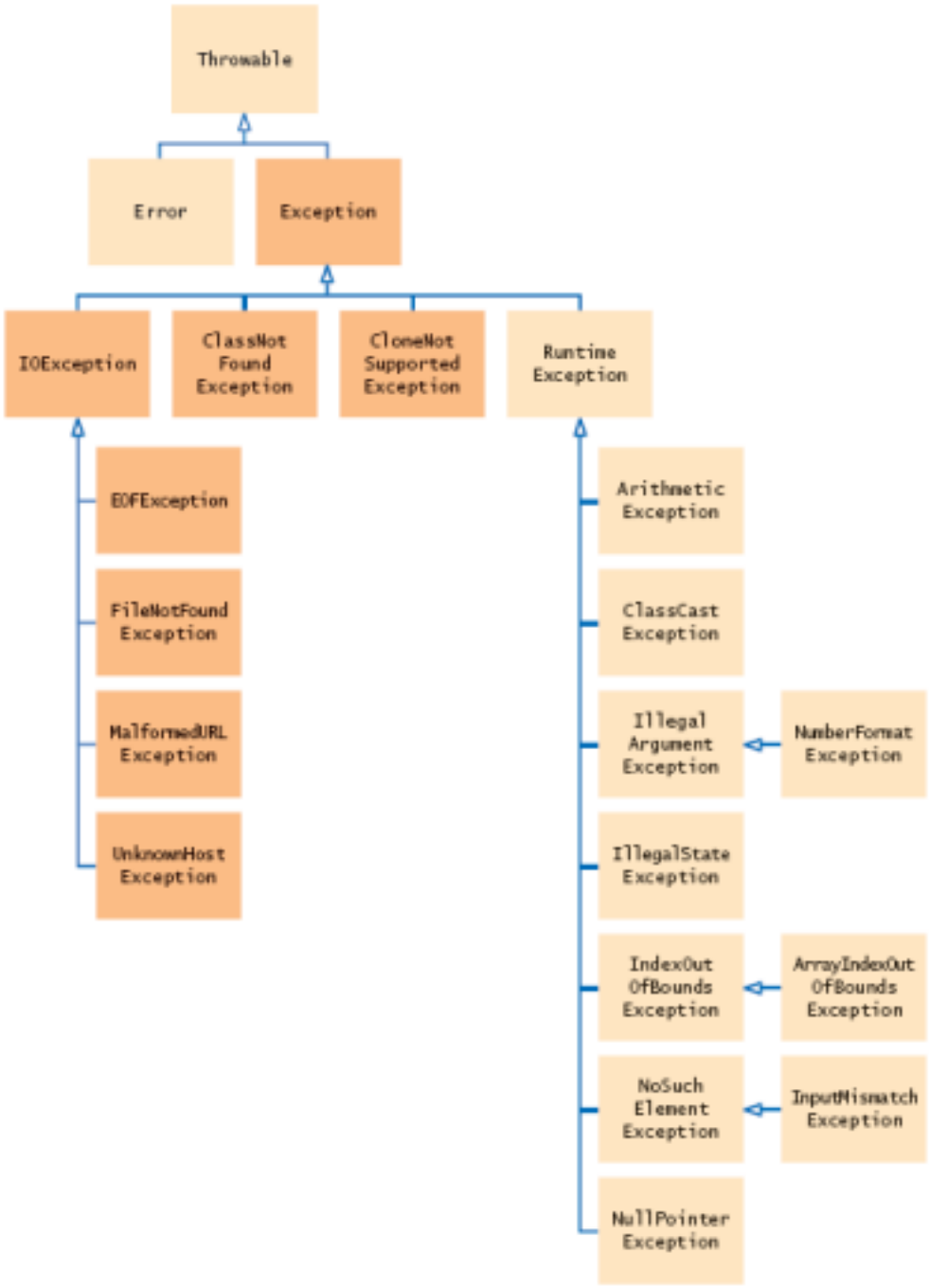
- No hay memoria disponible para asignar
- Acceso a un elemento de un array fuera de rango
- Leer por teclado un dato de un tipo distinto al esperado
- Error al abrir un fichero
- División por cero

JERARQUIA DE EXCEPCIONES

En Java, todas las excepciones están representadas por clases. Todas las clases de excepción se derivan de una clase llamada Throwable. Por lo tanto, cuando se produce una excepción en un programa, se genera un objeto de algún tipo de clase de excepción.

Hay dos subclases directas de Throwable: Exception y Error:

1. Las excepciones de tipo **Error** están relacionadas con errores que ocurren en la Máquina Virtual de Java y no en tu programa. Este tipo de excepciones escapan a su control y, por lo general, tu programa no se ocupará de ellas. Por lo tanto, este tipo de excepciones no se describen aquí.
2. Los errores que resultan de la actividad del programa están representados por subclases de Exception. Por ejemplo, dividir por cero, límite de matriz y errores de archivo caen en esta categoría. En general, tu programa debe manejar excepciones de estos tipos. Una subclase importante de Exception es RuntimeException, que se usa para representar varios tipos comunes de errores en tiempo de ejecución.



MANEJADOR DE EXCEPCIONES

El manejo de excepciones Java se gestiona a través de cinco palabras clave: try, catch, throw/throws y finally. Forman un subsistema interrelacionado en el que el uso de uno, implica el uso del otro.

Las declaraciones del programa que desea supervisar para excepciones están contenidas dentro de un bloque *try*. Si se produce una excepción dentro del bloque try, se lanza. Tu código puede atrapar esta excepción usando *catch* y manejarlo de una manera racional. Las excepciones generadas por el sistema son lanzadas automáticamente por el sistema de tiempo de ejecución de Java. Para lanzar manualmente una excepción, use la palabra clave *throw*. En algunos casos, una excepción arrojada por un método debe ser especificada como tal por una cláusula *throws*. Cualquier código que debe ejecutarse al salir de un bloque try se coloca en un bloque *finally*.

Ahora vamos a ver en detalle cada palabra clave dentro del manejo de excepciones.

EL BLOQUE TRY

Lo primero que hay que hacer para que un método sea capaz de tratar una excepción generada por la máquina virtual Java o por el propio programa mediante una instrucción throw es encerrar las instrucciones susceptibles de generarla en un bloque try. En el bloque try vamos a poner una serie de instrucciones que creemos que puede llegar a tirar una excepción durante su ejecución y queremos manejarla para evitar la finalización del programa.

```
try {
        Instrucción1;
        Intruscción2;
        Instrucción3;
        Instrucción4
        ...
}
```

Cualquier excepción que se produzca por alguna instrucción, dentro del bloque try será analizada por el bloque o bloques catch. En el momento en que se produzca la excepción, se abandona el bloque try, y las instrucciones que sigan al punto donde se produjo la excepción no son ejecutadas. Cada bloque try debe tener asociado por lo menos un bloque catch.

EL BLOQUE CATCH

Por cada bloque try pueden declararse uno o varios bloques catch, cada uno de ellos capaz de tratar un tipo u otro de excepción. Para declarar el tipo de excepción que es capaz de tratar un bloque catch, se declara un objeto cuya clase es la clase de la excepción que se desea tratar o una de sus superclases.

```
try {
 BloqueDeIntrucciones
} catch (TipoExcepción nombreVariable) {
 BloqueCatch
} catch (TipoExcepción nombreVariable) {
 BloqueCatch
}
```

Al producirse la excepción dentro de un bloque try, la ejecución del programa se pasa al primer bloque catch. Si la clase de la excepción se corresponde con la clase o alguna subclase de la clase declarada en el bloque catch, se ejecuta el bloque de instrucciones catch y a continuación se pasa el control del programa a la primera instrucción a partir de los bloques try-catch. Lo más adecuado es utilizar excepciones lo más cercanas al tipo de error previsto, ya que lo que se pretende es recuperar al programa de alguna condición de error y si "se meten todas las condiciones en la misma bolsa", seguramente habrá que averiguar después qué condición de error se produjo para poder dar una respuesta adecuada; en otras palabras, en la sentencia catch, en lugar de poner un supertipo para capturar cualquier excepción, lo más adecuado es colocar el tipo específico, por ejemplo si dentro del bloque try sabemos que se podría llegar a producir una NumberException, no capturarla con el tipo Excption que es el más genérico.

```
try {
        // Se intenta hacer la división
        int division = 10 / 0;
} catch (ArithmeticException a) {
        // Si la división falla el programa va al bloque catch y se ejecuta el  System.out.println
        System.out.println("Error: división por cero");
}
```

En este ejemplo en el bloque try hacemos una división por cero, las divisiones por cero generan un tipo de excepción llamado, ArithmeticExcepction. En el bloque catch ponemos como tipo de excepción la ArithmeticException y dentro del bloque, ponemos un mensaje que explique cual ha sido el error.

Aunque lo mismo funcionaría si en lugar de capturar la ArithmeticException capturamos una Excpetion no sería una buena práctica, ya que dijimos anteriormente que en lo posible se capture la excepción específica.

```
try {
        // Se intenta hacer la división
        int division = 10 / 0;
} catch (Exception a) {
        // Si la división falla el programa va al bloque catch y se ejecuta el  System.out.println
        System.out.println("Error: división por cero");
}
```

MÉTODOS THROWABLE

Dentro del bloque catch, utilizamos un System.out.print para mostrar el error, pero no hemos estado haciendo nada con el objeto de excepción en sí mismo. Como muestran todos los ejemplos anteriores, una cláusula catch especifica un tipo de excepción y un parámetro.

El parámetro recibe el objeto de excepción. Como todas las excepciones son subclases de Throwable, todas las excepciones admiten los métodos definidos por Throwable.

Estos métodos son:

Método	Sintaxis	Descripción
getMessage	String getMessage()	Devuelve una descripción de la excepción
fillInStackTrace	Throwable fillInStackTrace()	Devuelve un objeto Throwable que contiene un seguimiento de pila completo. Este objeto se puede volver a lanzar.
toString	String toString()	Devuelve un objeto String que contiene una descripción completa de la excepción. Este método lo llama println() cuando se imprime un objeto Throwable.

```
try {
      int division = 10 / 0;
} catch (ArithmeticException a) {
            System.out.println("Error:" + a.getMessage());
            System.out.println("Error:" + a)
            System.out.println(a.fillStrakTrace());
}
```

Resultado:

Error: / by zero

Error: / by zero

Error: java.lang.ArithmeticException: / by zero

EL BLOQUE FINALLY

El bloque finally se utiliza para ejecutar un bloque de instrucciones sea cual sea la excepción que se produzca. Este bloque se ejecutará siempre, incluso si no se produce ninguna excepción. Sirve para no tener que repetir código en el bloque try y en los bloques catch. El bloque finally es un buen lugar en donde liberar los recursos tomados dentro del bloque de intento.

```
try {
            BloqueDeIntrucciones
} catch (TipoExcepción nombreVariable) {
            MensajeDeError
} catch (TipoExcepción nombreVariable) {
            MensajeDeError
} finally {
            CodigoFinal
            Siempre se ejecuta.
}
```

Ejemplo:

```
try {
// Se intenta hacer la división
int division = 10 / 0;
} catch (ArithmeticException a) {
// Si la división falla el programa va al bloque catch y se ejecuta el  System.out.println
System.out.println("Error: división por cero");
```

```
} finally {
// Si el programa hizo la división o no, este System.out.println se va a  ejecutar igual
System.out.println("Saliendo del try");
}
```

LA CLAÚSULA THROWS

La cláusula *throws* lista las excepciones que un método puede lanzar. Los tipos de excepciones que lanza el método se especifica después de los paréntesis de un método, con una cláusula throws. Un método puede lanzar objetos de la clase indicada o de subclases de la clase indicada.

Java distingue entre las excepciones verificadas y errores. Las excepciones verificadas deben aparecer en la cláusula throws de ese método. Como las RuntimeExceptions y los Errores pueden aparecer en cualquier método, no tienen que listarse en la cláusula throws y es por esto que decimos que no están verificadas. Todas las excepciones que no son RuntimeException y que un método puede lanzar deben listarse en la cláusula throws del método y es por eso que decimos que están verificadas. El requisito de atrapar excepciones en Java exige que el programador atrape todas las excepciones verificadas o bien las coloque en la cláusula throws de un método.

Si la excepción no se trata, el manejador de excepciones realiza lo siguiente:

• Muestra la descripción de la excepción.

• Muestra la traza de la pila de llamadas.

• Provoca el final del programa.

Colocar una excepción en la cláusula throws obliga a otros métodos a ocuparse de la excepción. Esto se puede hacer colocando otro throws al método que llama al método, con el tipo de excepción que podría tirar o rodear el llamado del método con un try-catch, y de esa manera que el try-catch se encargue de manejar la excepción que podría tirar el método.

```
[acceso][modificador][tipo] nombreFuncion() throws TipoDeExcepcion { Bloque de
instrucciones
}
```

Ejemplo:

```
// Tenemos un método que devuelve un resultado y que tira una  //
ArithmeticException
public int division() throws ArithmeticException {
        int division;
division = 20 / 0; // Si la división tira una excepción la manejará el  // try/catch del llamado a la
                                                    función.
        return division; // Si tira una excepción el método no va a devolver ningún  // resultado.
}
```

Main

```
// Llamamos a la función, si tira una excepción va al bloque catch y ejecuta el mensaje de
error, sino imprime el resultado de la división.

try {

        System.out.println(division());

} catch (ArithmeticException a) {

        System.out.println("Error: división por cero");

}
```

LA PALABRA THROW

Los programas escritos en Java pueden lanzar excepciones explícitamente mediante la instrucción throw, lo que facilita la devolución de un "código de error" al método que invocó el método que causó el error. La cláusula throw debe ir seguida del tipo de excepción que queremos que lance el método. Puede lanzarse cualquier tipo de excepción que implemente la interfaz Throwable.

Cuando se lanza una excepción usando la palabra throw, el flujo de ejecución del programa se detiene y el control se transfiere al bloque try-catch más cercano que coincida con el tipo de excepción lanzada. Si no se encuentra tal coincidencia, el controlador de excepciones predeterminado finaliza el programa. La palabra clave throw es útil para lanzar excepciones basadas en ciertas condicione, por ejemplo, si un usuario ingresa datos incorrectos. También es útil para lanzar excepciones personalizadas específicas para un programa o aplicación.

Cuando utilicemos la palabra throw en un método, vamos a tener que agregarle la palabra throws al método con la excepción que va a tirar nuestro throw. De esa manera avisamos que cuando se llame al método hay que manejar una posible excepción.

```
throw new TipoExcepcion("Mensaje de error");
```

Ejemplo:

En este método recibimos una lista y un número para agregar a dicha lista. El método contiene la palabra throws para avisar que este método puede tirar una excepción

```
public void agregarNumeroLista(List<Integer> lista, int numero) throws  Exception {

        // Validamos si la lista ya tiene el numero a agregar

        if(lista.contains(numero){

                // Si lo tiene tiramos un excepción de tipo Exception, poniéndole un  mensaje
                entre los paréntesis

                throw new Exception("El numero ya está en la lista);

        }

        lista.add(numero); // Si no contiene el numero, lo agregamos a la lista Main:
```

```
}



//Luego, cuando llamamos al método, lo hacemos dentro de un try/catch para manejar la
posible excepción, además le pasamos la lista al método  y el número.

List<Integer> lista = new ArrayList();


try{
        agregarNumeroLista(lista, 1);
}catch (Exception e){
        System.out.println(e.getMessage);
        //Usamos el metodo getMessage, para obtener el mensaje que pusimos en el  throw
}
```

Podríamos crear nuestra propia excepción, para ello bastará con crear una clase que herede de Exception.

```
public class MiExcepcion extends Exception {


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

Entonces nuestro método podría lanzar una excepción personalizada:

```
public void agregarNumeroLista(List<Integer> lista, int numero) throws  Exception {
        // Validamos si la lista ya tiene el numero a agregar
        if(lista.contains(numero){
                // Si lo tiene tiramos un excepción de tipo Exception, poniéndole un  mensaje
                entre los paréntesis
                throw new MiExcepcion("El numero ya está en la lista);
        }
        lista.add(numero); // Si no contiene el numero, lo agregamos a la lista Main:
}
```

BIBLIOGRAFÍA

Información sacada de las páginas:

- https://www.oracle.com/ar/database/what-is-a-relational-database/

- https://www.geeksforgeeks.org/sql-tutorial/