UNIVERSIDAD DEL BÍO-BÍO

Paradigmas de la Programación

Manejo de excepciones

UNIVERSIDAD DEL BÍO-BÍO

Excepciones

Al ejecutar código, pueden ocurrir diferentes tipos de errores. Cada error que se produzca puede deberse a factores internos (Errores de codificación/Lógica) o externos (Acceso denegado/Intento de envío o recepción de mensaje sin éxito/Etc.).

Cuando se produce un error, normalmente la ejecución del código se detendrá y generará un mensaje de error.

El término técnico se define como: Lanzamiento de excepción.

UNIVERSIDAD DEL BÍO-BÍO

Excepciones

En el siguiente código se intenta obtener un elemento de una posición que aún no existe.

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

    ArrayList<String> textos = new ArrayList<>();
    textos.add("Hola");
    System.out.println(textos.get(10));

}
```

```
Exception in thread "main" java.lang.IndexOutOfBoundsException: Index: 10, Size: 1
        at java.util.ArrayList.rangeCheck(ArrayList.java:653)
        at java.util.ArrayList.get(ArrayList.java:429)
        at javaapplication25.JavaApplication25.main(JavaApplication25.java:23)
```

UNIVERSIDAD DEL BÍO-BÍO

Excepciones comunes

Excepción	Descripción
ArrayIndexOutOfBoundsException	Se intenta acceder a un arreglo con un índice inválido (menor que 0 o mayor a la longitud del array).
ClassCastException	Intento fallido de transformar (Cast) un objeto en otro tipo.
NullPointerException	Intento de acceder o utilizar un objeto con referencia nula.
IllegalArgumentException	Método que recibe un argumento formateado de diferente manera a lo que se espera que ingrese.
NumberFormatException	Intento fallido de convertir un String a un número.
StackOverflowError	Desbordamiento de Stack en una llamada recursiva incorrecta(Ej.: Ciclo infinito).

Manejo de excepciones

Gracias al manejo de excepciones, se tiene completo control de los errores que pudiesen ocurrir, manipulando el flujo del programa.

Para el control de excepciones se utiliza la siguiente estructura:

```
try
{

}
catch(Exception e)
{

}
```

Manejo de excepciones - try

El bloque try contiene un conjunto de sentencias donde puede producirse una excepción. Un bloque try siempre va seguido de un bloque catch, que controla la excepción originada.

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

    ArrayList<String> textos = new ArrayList<>();
    textos.add("Hola");

    try
    {
        int i= 0;
        i+= 10;
        System.out.println(textos.get(i));
    }
    catch (Exception e)
    {

    }

}
```

Manejo de excepciones - catch

El bloque catch se utiliza para manejar la condición incierta del bloque try.

En el caso que se produzca una excepción (Error) en el bloque try, automáticamente se ejecuta el bloque catch.

Observación: Si se produce un error y aún hay líneas por ejecutar en el bloque try, estas son ignoradas para ejecutar directamente el bloque catch.

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

    ArrayList<String> textos = new ArrayList<>();
    textos.add("Hola");

    try
    {
        int i= 0;
        i+= 10;
        System.out.println(textos.get(i));
    }
    catch (Exception e)
    {
        System.out.println("El error está controlado\n"+
                "Todo está bien :)");
    }
}
```

UNIVERSIDAD DEL BÍO-BÍO

Bloque catch múltiple

Un bloque try puede tener múltiples bloques catch asociados a el.

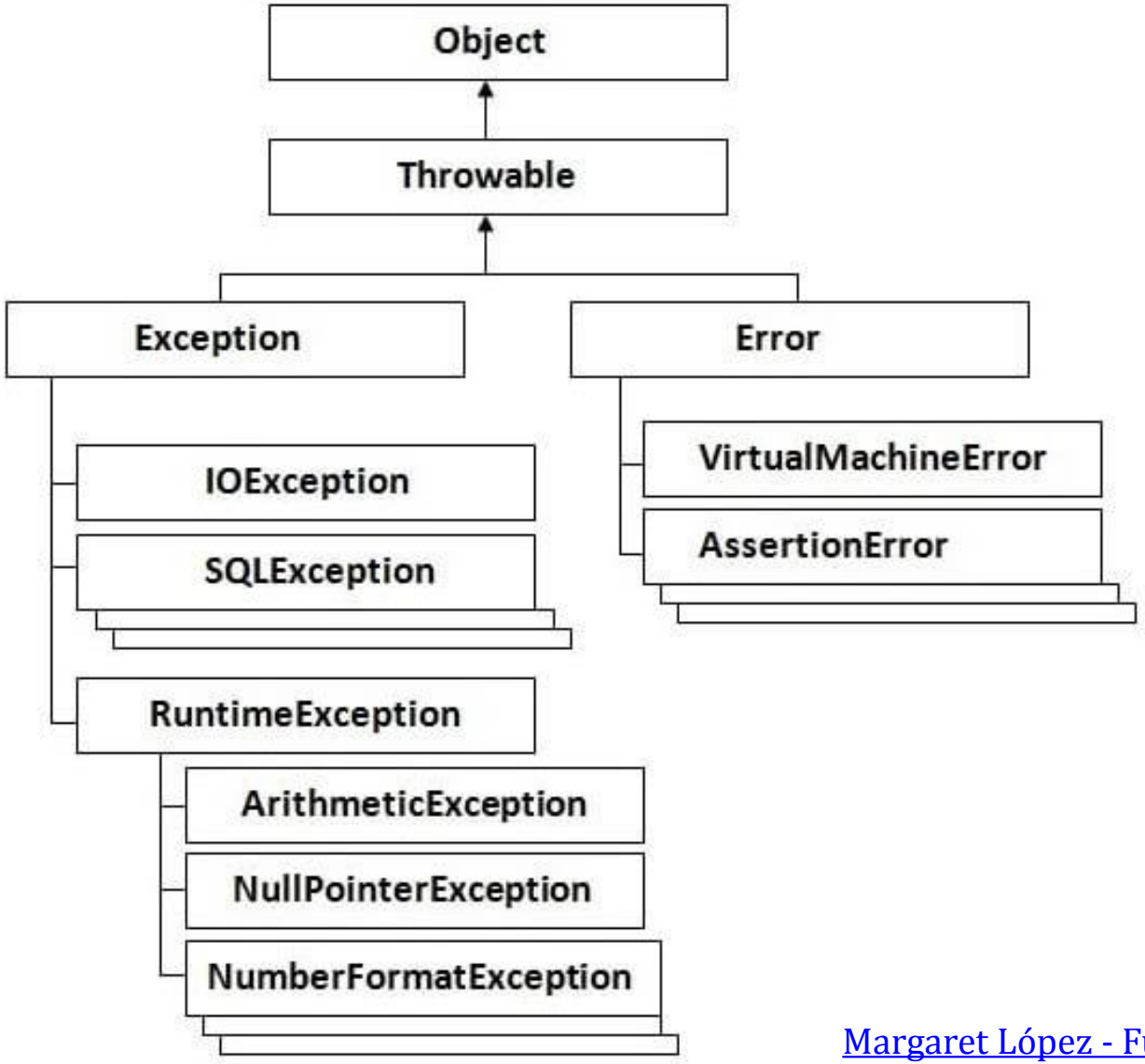
Observación 1: El bloque catch que se ejecute será el **más** cercano al tipo de excepción lanzada.

Observación 2: El orden de los bloques catch, debe ir desde el **más** específico al **menos** específico.

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

    ArrayList<String> textos = new ArrayList<>();
    textos.add("Hola");

    try {
        int i = 0;
        i = i + 10 / 0;
        System.out.println(textos.get(i));
    }
    catch (IndexOutOfBoundsException e) {
        System.out.println("Index inválido en el Arreglo.");
    }
    catch (ArithmeticException e) {
        System.out.println("No es posible dividir por cero.");
    }
    catch(Exception e) {
        System.out.println("Hay un error de algo.");
    }

}
```

UNIVERSIDAD DEL BÍO-BÍO

Jerarquía de Excepciones



Margaret López - Fuente de Imagen

UNIVERSIDAD DEL BÍO-BÍO

Bloque finally

El bloque finally se ejecuta al termino del bloque try o catch, sin importar lo que suceda.

Observación 1: Su uso es opcional.

Observación 2: Si el bloque try termina toda su ejecución se ejecutará el bloque finally. Si se produce una excepción se ejecutará el bloque catch y por último el bloque finally.

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

    ArrayList<String> textos = new ArrayList<>();
    textos.add("Hola");

    try {
        int i = 0;
        i = i + 10 / 0;
        System.out.println(textos.get(i));
    }
    catch (Exception e) {
        System.out.println("Index inválido en el Arreglo.");
    }
    finally
    {
        System.out.println("Algo se puede hacer aquí.");
    }

}
```

Delegación de excepciones

En el caso que nuestra implementación pueda producir errores, se podrá forzar el uso de try/catch desde el punto en que se llame a un método o constructor.

La implementación se realiza al termino de la declaración del constructor o método, agregando la palabra clave throws seguido de las excepciones que pudiesen ocurrir.

Observación: Las excepciones que heredan de **RuntimeException** o **Error**, no obligarán el uso de try/catch. Las que heredan directamente de Exception si lo harán.

Observación: Se pueden agregar tantas excepciones sean necesarias después de la palabra clave throws.

Clase Base

```
public class A {

    public A() throws Exception {

    }

    public void proceso() throws IOException, ArithmeticException {
        System.out.println("Proceso muy riesgoso :S");
    }

}
```

No se podrá ejecutar el código hasta que dicho constructor y método estén dentro de un try/catch.

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

    A a = new A();

    a.proceso();

}
```

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

    A a = null;
    try
    {
        a = new A();
    } catch (Exception ex) {

    }

    try
    {
        a.proceso();
    }
    catch (IOException ex) {

    }
    catch (ArithmeticException ex) {

    }

}
```

Lanzamiento de excepciones

Hasta el momento cuando se detecta un error, lo máximo que se podía hacer era imprimir un mensaje por consola o terminar la ejecución del método. Bajo el sistema excepciones, es posible detener la ejecución, lanzando una excepción de manera explicita o ejecutando directamente bloques catch.

Para el lanzamiento de excepciones se ocupará la palabra clave throw (No throws) seguido de una instancia de Excepción.

Observación: Al lanzar una excepción, la ejecución de código termina en ese punto, en el caso que hubiera un bloque catch, será re-dirigido a dicho bloque.

```
throw new Exception("Error :S");
```

Lanzamiento dentro de un try/catch

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

    int i = 5;

    try {
        if (i == 5)
            throw new ArithmeticException("No puede ser cinco. - Error");

        if (i == 0)
            throw new Exception("Error :S");

        i++;
    }
    catch (ArithmeticException e)
    {
        System.out.println("Error - Cinco");
    }
    catch (Exception e)
    {
        System.out.println("Error - Cero");
    }

}
```

Si se llega a ejecutar se re-dirige a este bloque.

Si se llega a ejecutar se re-dirige a este bloque.

Lanzamiento dentro de un método

```
public class A {

    private int dato;

    public A(int dato) {
        this.dato = dato;
    }

    public void proceso() {
        if (dato == 0)
            throw new ArithmeticException("¡Esto no está bien! ¡Lanzar error!");

        dato += 1;
    }

}
```

Observación: En el caso que se desee lanzar una excepción que hereda directamente de **Exception**, será necesario indicar los tipos posibles de excepciones que pudiesen ocurrir en el método(**throws**). Para excepciones de tipo **RuntimeException** o **Error** no será necesario.

Creando tipos de excepciones

Para poder crear nuestro propio tipo de excepción, solamente será necesario crear una clase que herede de algún tipo de Excepción (Exception /RuntimeExcepcion /Error /Etc).

```
public class MiExcepcion extends Exception{

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

}
```

Usando nueva excepción

```
public class A {

    private int dato;

    public A(int dato) {
        this.dato = dato;
    }

    public void proceso() throws MiExcepcion {
        if (dato == 0)
            throw new MiExcepcion("Notificar error.");

        dato += 1;
    }

}
```