MANEJO DE EXCEPCIONES EN JAVA

EXCEPCIONES

- Una excepción es una situación anómala que surge en un bloque de código, es decir, es un error que normalmente aparece en tiempo de ejecución y que impide que continúe la ejecución de un método.

- Qué ocurre cuando aparece una excepción?

EXCEPCIONES

- **Una excepción es una situación anómala que surge en un bloque de código, es decir, es un error que normalmente aparece en tiempo de ejecución y que impide que continúe la ejecución de un método.**

- **Qué ocurre cuando aparece una excepción?**

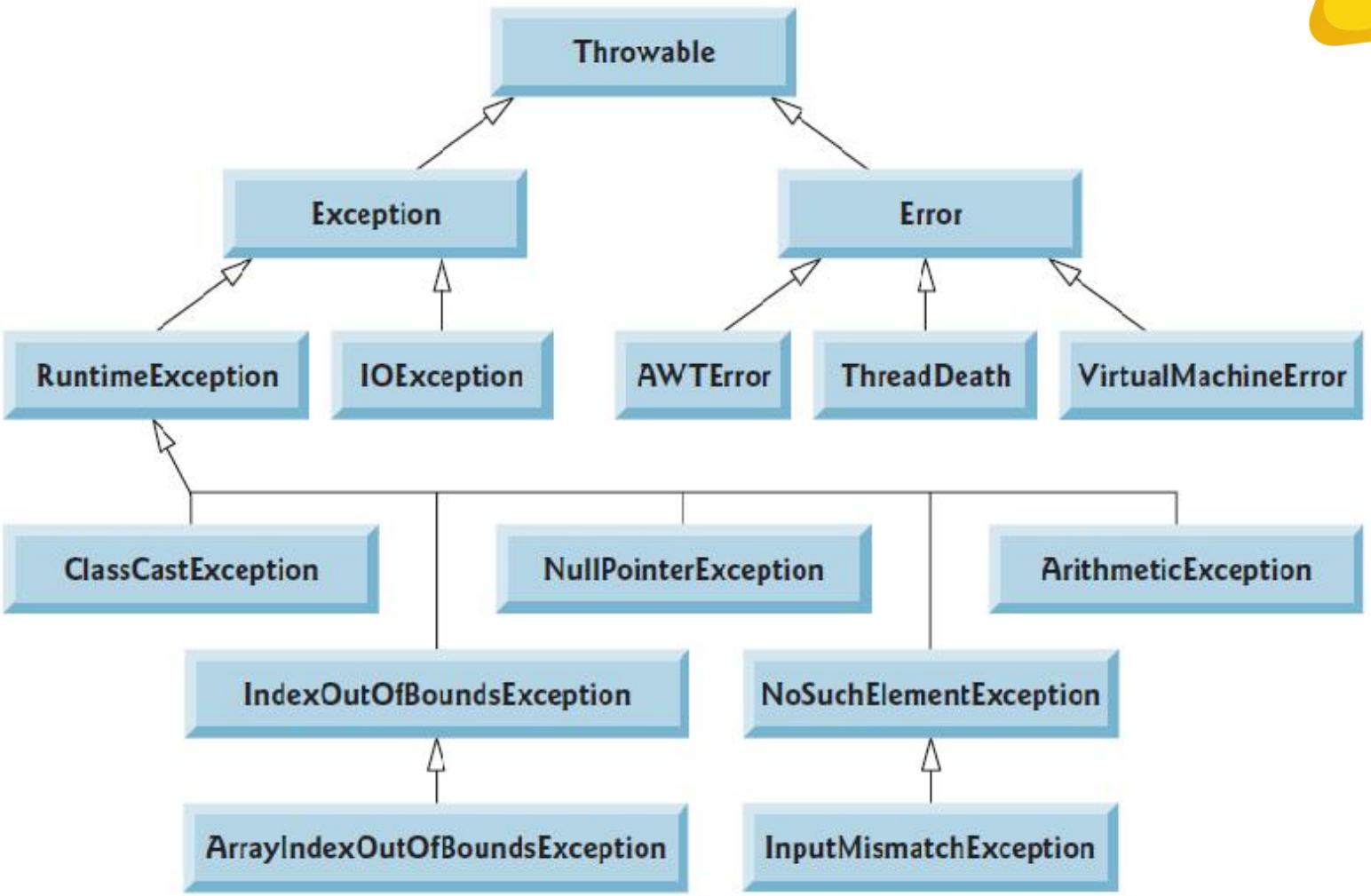
Cuando se genera una excepción el interprete **crea un objeto para representar la excepción y** envía el objeto (la excepción) al método que la ha provocado.

Si el método no captura la excepción, entonces

el interprete la captura y realiza las acciones pertinentes (**detención del programa y avisos** o mensajes por pantalla).

Java tiene un sistema para que el programador defina la captura y gestión de las excepciones

EXCEPCIONES: JERARQUIA



https://docs.oracle.com/javase/7/docs/api/java/lang/RuntimeException.html

Tipos de excepciones

- ***Error***: Excepciones que indican problemas muy graves, que suelen ser no recuperables y no deben casi nunca ser capturadas. **Ej**: Memoria agotada, error interno de la JVM

- ***Exception***: La clase Exception indica las condiciones que una aplicación querría poder atrapar. Esta clase maneja excepciones que se lanzan dentro y fuera del tiempo de ejecución.
  - **IOException (fuera de ejecución):** Errores que están por fuera del alcance del programador. Ej. No se encuentra un archivo que se intenta leer.

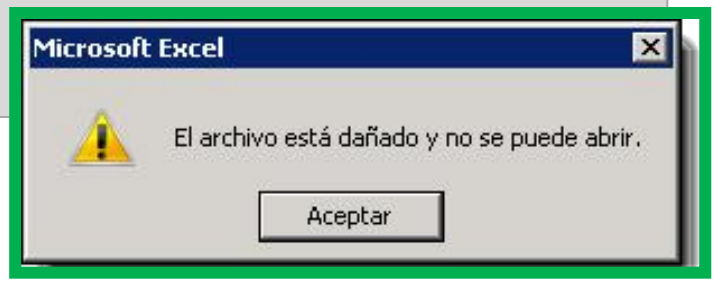
- ***RuntimeException***: Excepciones que se dan durante la ejecución del programa.
  - **Ejs**: una división por cero o acceder un elemento fuera de los limites de un *array*.

Checked vs Unchecked Exceptio

Checked

Excepciones que se deben manejar durante el proceso de compilación. Si no se manejan sale un error de compilación:

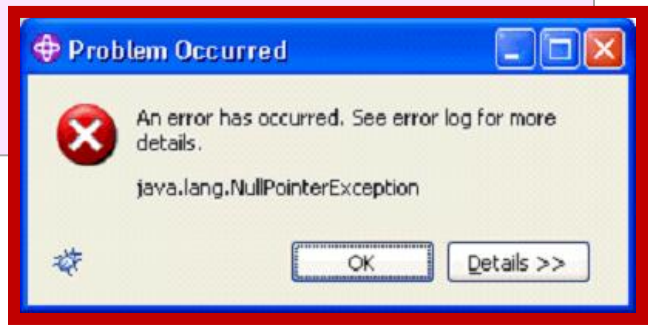
- IOException
- SQLException
- DataAccessException
- ClassNotFoundException



UnChecked

Excepciones que **no se verifican en el proceso de compilación**. Generalmente
son errores del programador y se lanzan en tiempo de ejecución

- ✓ NullPointerException
- ✓ ArrayIndexOutOfBound
- ✓ ArithmeticException
- ✓ IllegalArgumentException



EXCEPCIONES: EJEMPLO

```
public static double division(int x, int y){
    return x/y;

}
```

```
Numerador: 5
Denominador: 0
Exception in thread "main" java.lang.ArithmeticException: / by zero
        at lab1pi.ExceptionManeger.division(ExceptionManeger.java:18)
        at lab1pi.ExceptionManeger.main(ExceptionManeger.java:29)
Java Result: 1
```

CAPTURA Y GESTIÓN DE EXCEPCIONES

```
try {

// bloque de código en el que gestionamos la excepción
}

catch ( tipo_excepción1 objeto ) {

}


catch ( tipo_excepción2 objeto ) {

}
```

try: ejecuta un bloque de código y lanza la excepción a la primera sentencia ***catch*** que gestione ese tipo de excepción

catch: captura la excepción y realiza las acciones correspondientes

EXCEPCIONES: EJEMPLO

```
public double division(int num, int den)
{
    double resul;
    try
    {
        resul = num/den;

    }catch(ArithmeticException ae){
        JOptionPane.showMessageDialog(null, "Error al dividir por 0"+ae);
        resul = -1;
        menu();
    }
    return resul;

}
```

CLAUSULA FINALLY

- La clausula **Finally** ejecutar un fragmento de código **independientemente de si se produce o no una excepción**.

Ej: *Cerrar un archivo que se está manipulando.*

- ***finally***: se ejecuta al final de ***try*** y después de ***catch***, además captura las excepciones no capturadas por una sentencia ***catch***.

CLAUSULA FINALLY

```
try {

// bloque de código en el que gestionamos la excepción
}

catch ( tipo_excepción1 objeto ) {

}


catch ( tipo_excepción2 objeto ) {

}

finally {

}
```

Si el cuerpo ***try*** comienza la ejecución, ***finally*** **siempre se ejecutará**

Tipos de excepciones

Tipo Excepción	Descripción
ArithmeticException	Errores Matemáticos, como división por cero
ArrayIndexOutOfBoundsException	Un programa tratando de almacenar un dato en una posición no válidos de un arreglo
FileNotFoundException	Un intento de acceder a un archivo que no existe
IOException	Fallas de entrada/salida, tal como la inhabilidad de leer desde un archivo
NullPointerException	Referencia a un objeto NULL
NumberFormatException	Una conversión fallida entre String y números

Excepciones: Mensajes

Método	Descripción
printStackTrace	Imprime la pila de llamadas a métodos.
getStackTrace	Obtiene la información que imprime printStackTrace.
getMessage	Devuelve la cadena descripta almacenada en una excepción.

```
Exception in thread "main" java.lang.ArithmeticException: / by zero
        at lab1pi.ExceptionManeger.division(ExceptionManeger.java:18)
        at lab1pi.ExceptionManeger.main(ExceptionManeger.java:29)
Java Result: 1
```

Excepciones: Mensajes

Método	Descripción
printStackTrace	Imprime la pila de llamadas a métodos.
getStackTrace	Obtiene la información que imprime printStackTrace.
getMessage	Devuelve la cadena descripta almacenada en una excepción.

Tipo de excepción

Mensaje de la excepción

```
Exception in thread "main" java.lang.ArithmeticException: / by zero
        at lab1pi.ExceptionManeger.division(ExceptionManeger.java:18)
        at lab1pi.ExceptionManeger.main(ExceptionManeger.java:29)
Java Result: 1
```

Pila de llamados

ArithmeticException

```
public static double division(int x, int y){
    return x/y;

}
```

```
Numerador: 5
Denominador: 0
Exception in thread "main" java.lang.ArithmeticException: / by zero
        at lab1pi.ExceptionManeger.division(ExceptionManeger.java:18)
        at lab1pi.ExceptionManeger.main(ExceptionManeger.java:29)
Java Result: 1
```

ArrayIndexOutBounds

Ocurre cuando se trata de acceder a una posición que no existe en un arreglo.

```
public class PruebaException3 {
   public void produceExcepcion(){
     int numero[]=new int[5];

     try{
          numero[7]=10;
         System.out.println("Accesando a una posicion fuera del vector");
     }

     catch (ArrayIndexOutOfBoundsException excep){
        System.out.println("Ocurrio una excepcion");
        System.out.println(excep.getMessage() );
         //excep.printStackTrace();
     }

}
```

ArrayIndexOutBounds

```
public class ArrayException {
     double notas[] =  new double[10];

     public void setNotas(){

          for (int i=0; i<= notas.length; i++){
               notas[i]= Math.round(Math.random()*10);
          }
     }
     public static void main (String a[]){

          ArrayException ar=  new ArrayException();
          ar.setNotas();

     }
}
```

```
Exception in thread "main" java.lang.ArrayIndexOutOfBoundsException: 10
        at piEjemplos.ArrayException.setNotas(ArrayException.java:19)
        at piEjemplos.ArrayException.main(ArrayException.java:25)
Java Result: 1
```

ArrayIndexOutBounds

```
public class ArrayException {
     double notas[] =  new double[10];

     public void setNotas(){

          for (int i=0; i<= notas.length; i++){
               notas[i]= Math.round(Math.random()*10);
          }
     }
     public static void main (String a[]){

          ArrayException ar=  new ArrayException();
          ar.setNotas();

     }
}
```

```
Exception in thread "main" java.lang.ArrayIndexOutOfBoundsException: 10
        at piEjemplos.ArrayException.setNotas(ArrayException.java:19)
        at piEjemplos.ArrayException.main(ArrayException.java:25)
Java Result: 1
```

NumberFormatException

- Cuando se intenta procesar como un dato numérico algo que en realidad es un alfanumérico (por ejemplo una letra).

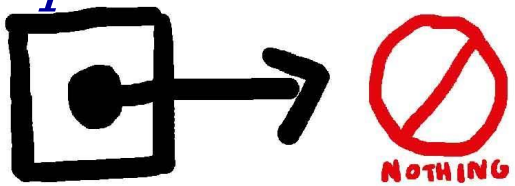
```
public class NumberFormatExample {
    public static void main(String a[]){
        try{
            int edad = Integer.parseInt(JOptionPane.showInputDialog("Digite su edad"));
            System.out.println("Edad leida exitosamente");
        }
        catch (NumberFormatException exc){
            System.out.println("Ha ocurrido una excepcion");
             System.out.println("Digite un numero");

        }
    }
}
```

NullPointerException

- Todos los valores en Java, **excepto las primitivas**, son **referencias a objetos** y todos tienen un valor por defecto, el valor *null*.

- A un objeto con valor *null*, que no referencia ningún objeto, no se le puede aplicar ningún método

- **Si se aplica algún métodos se lanza una excepción** de tipo *NullPointerException*



NullPointerException

```
public class NullPointerExceptionExample {

        public static void main(String[] args){

            String cadenaPrueba = null;
            try{

                if (cadenaPrueba.equals("")) {
                    System.out.println("Cadena es vacia");
                }else{
                    System.out.println("Cadena no vacia");
                }

            } catch(NullPointerException ex){
                System.out.println("Una excepción: "+ex);
            }
        }
}
```

NullPointerException

```java
public class Casa {

    private long precio;

    public Casa (long precio){
        this.precio = precio;
    }

    public long getPrecio(){
        return precio;
    }
}
```

```java
public class VentaCasa {


    private Casa casa;

    public static void main(String[] args){

        VentaCasa vender = new VentaCasa();
        System.out.println(vender.casa.getPrecio());
    }

}
```

NullPointerException

```java
public class Casa {

    private long precio;

    public Casa (long precio){
        this.precio = precio;
    }

    public long getPrecio(){
        return precio;
    }
}
```

```java
public class VentaCasa {


    private Casa casa;

    public static void main(String[] args){

        VentaCasa vender = new VentaCasa();
        System.out.println(vender.casa.getPrecio());
    }

}
```

```
Exception in thread "main" java.lang.NullPointerException
        at piExcepciones.VentaCasa.main(VentaCasa.java:21)
Java Result: 1
```

Múltiples Excepciones

```
static void divide(){
    int num[]={4,8,16,32,64,128,256};
    int den[]={2,0,4,4,0,8};

    for (int i=0;i<num.length+1;i++){
        try {
            System.out.println(num[i]+ "/" + "=" + num[i]/den[i]);
        }

        catch(ArithmeticException e){
            System.out.println("Excepción dividiendo por cero");
        }

        catch(java.lang.ArrayIndexOutOfBoundsException e){
            System.out.println("Excepción  al acceder al arreglo");
        }
    }
}
```

Múltiples Excepciones

```
for (int i=0;i<num.length+1;i++){
      try {
          System.out.println(num[i]+ "/" + "=" + num[i]/den[i]);
      }

      catch(ArithmeticException e){
         System.out.println("Excepción dividiendo por cero");
      }

      catch(ArrayIndexOutOfBoundsException e){
        System.out.println("Excepción  al acceder al arreglo");
      }

      catch (Exception e){
          System.out.println ("Captura cualquier excepción");
      }
}
```

EJERCICIOS

- Implemente las excepciones a los programas creados de Empleado, y calculadora