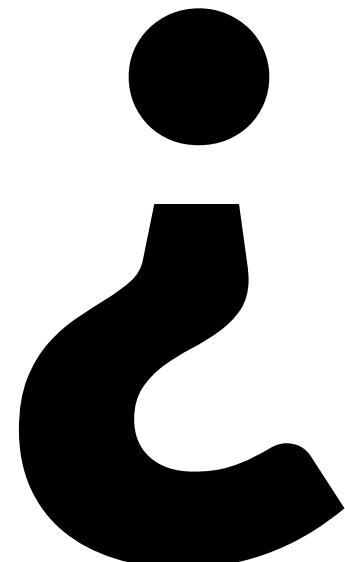
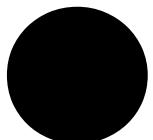


Excepciones

UNRN

Universidad Nacional
de Río Negro





Como les fue con el TP

**No hace falta
cargar los reportes
al repositorio**

Se *regeneran* al momento de
corregir

**Recuerden que al
informe le falta la
corrección.**



El resumen de lo visto en la semana

Por Dredd



Checkstyle

**Se solucionan
con autoformato**

1

`` en la columna 40 debería tener salto de línea después.

`` no está seguido de espacio en blanco.

Line has trailing spaces.

y muchos más

2

**La línea es mayor de 90 caracteres
(encontrado 92).**

3

Falta el comentario Javadoc.

La primera frase debería finalizar con un punto.

Se esperaba la etiqueta @param para `numeroBase`.

4

‘3’ es un número mágico.

**iRename al
rescate!**

5

El nombre `factor_a` debe coincidir con el patrón `^[a-z][a-zA-Z0-9]*\$`.

El nombre `EsPrimo` debe coincidir con el patrón `^[a-z][a-zA-Z0-9]*\$`.

**Háganle caso al
check en su
computadora**

**El autoformato
arregla bastante**

PMD

**Como lo verían para
que nos ayude más**

Mandatorio

1

CommentRequired de Documentation

2

Ignorar

**CloseResource, Ensure that resources
like this InputStream object are
closed after use**

**Esto va a aparecer siempre que usemos
Scanner**

3

OnlyOneReturn, A method should have only one exit point, and that should be the last statement in the method

Mandatorio

**Opcional
pero recomendado**

4

AvoidReassigningParameters
Avoid reassigning parameters such as
'dividendo'

AvoidReassigningParameters

```
public static int suma(int sumando, int sumador) {  
    sumando = sumando + sumador;  
    return sumando;  
}
```

Opcional
pero recomendado

5

ShortVariable
Avoid variables with short names like
i

**Opcional
pero recomendado**

6

AvoidLiteralsInIfCondition Avoid using Literals in Conditional Statements

AvoidLiteralsInIfCondition

```
if (valor == 6)
```

Prefieran

```
if (valor == TOPE)
```

Opcional

7

**UnusedAssignment
The initializer for variable
'divisor' is never used (overwritten
on line 32)**

UnusedAssignment

```
int variable = 10;  
// program happens  
variable = funcion();
```

```
int variable;  
// program happens  
variable = funcion();
```

**Opcional
pero recomendado**

8

ConfusingTernary

Avoid if (x != y) ..; else ..;

Si el if tiene else, eviten negar la condición.

ConfusingTernary

¡Si tiene sentido!

```
if (divisor != 0) {  
    // dividimos  
} else {  
    // no podemos  
}
```

Prefieran

```
if (divisor == 0) {  
    // no podemos  
} else {  
    // dividimos  
}
```

9

Useless Parentheses

Mejor que sobren a que falten

Ignorar

1

Ignorar

LongVariable
Avoid excessively long variable names
like numeroDeOperaciones

0

**Es mejor usar nombres descriptivos de
todas formas.**

1

Mandatorio

UseUnderscoresInNumericLiterals

Number 16545645 should separate every
third digit with an underscore

1

1

Mandatorio

**ControlStatementBraces
This statement should have braces**

2

Todos los bloques van con llaves.

Control Statement Braces

```
if (variable == 0)  
    variable = variable
```

Debe ir

```
if (variable == 0) {  
    variable = variable + 1  
}
```

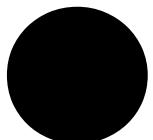


¿Preguntas?



**Abran
hilo**

Como armar uno sobre este tema



Como preguntar sobre esto

Agreguen contexto

Me encontré un:

ShortVariable de Code Style en la línea 53
Avoid variables with short names like a

```java  
codigo que contiene el mensaje

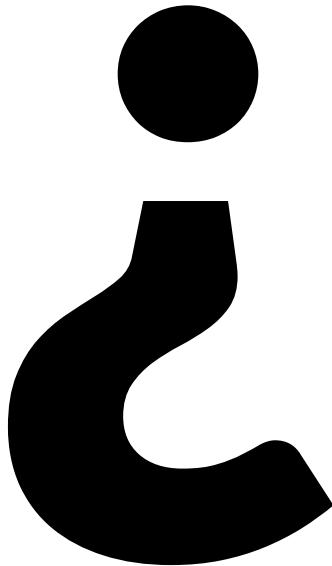
```

¿Le hacemos caso?



¿Preguntas?

Gestión de errores excepciones



Qué son



**Cualquier
interrupción al flujo
de instrucciones
normal del
programa**

¿Que son?

¿Cuál es la salida acá?

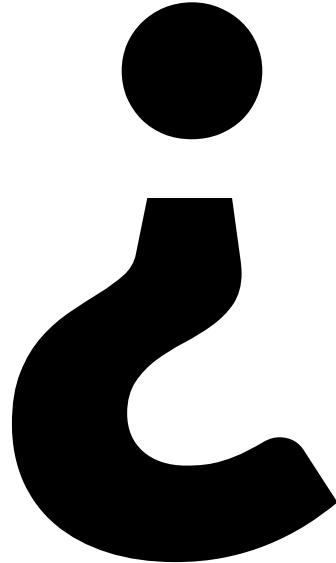
```
public class ArregloApp {  
    public static void main(String[] args) {  
        int[] numeros = {1, 2, 3};  
        System.out.println(numeros[3]);  
    }  
}
```



¡BOOM!

El Stacktrace

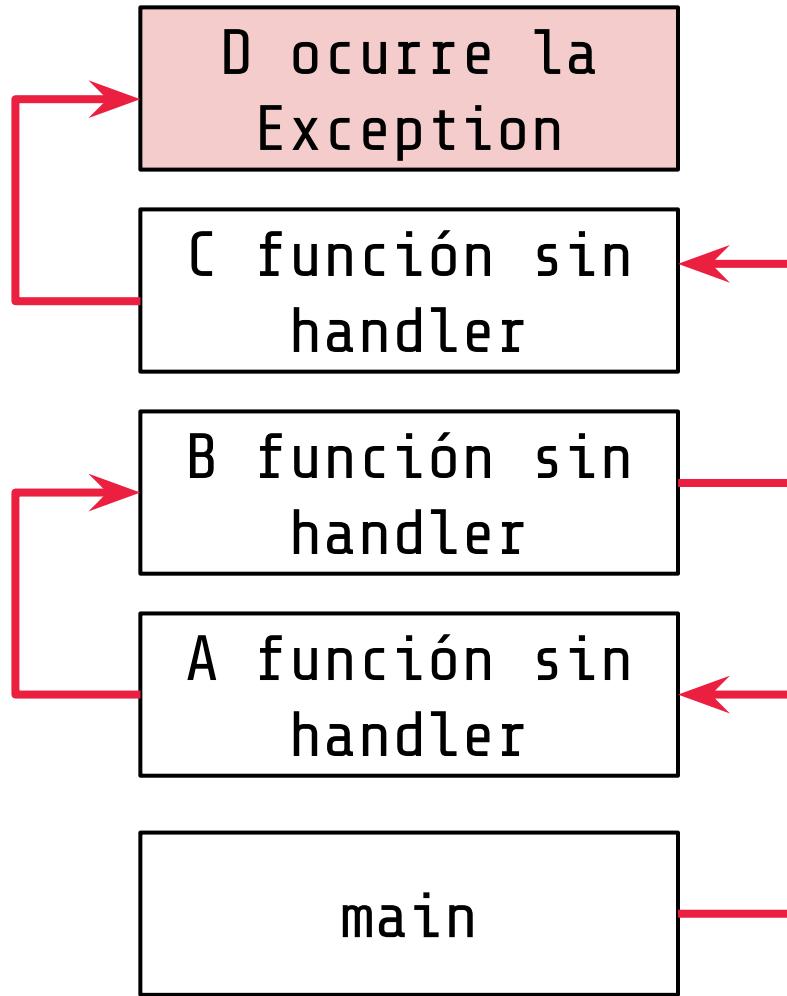
```
Exception in thread "main"
java.lang.ArrayIndexOutOfBoundsException: Index 3 out of bounds
for length 3
at ar.unrn.ArregloApp.main(ArregloApp.java:21)
```



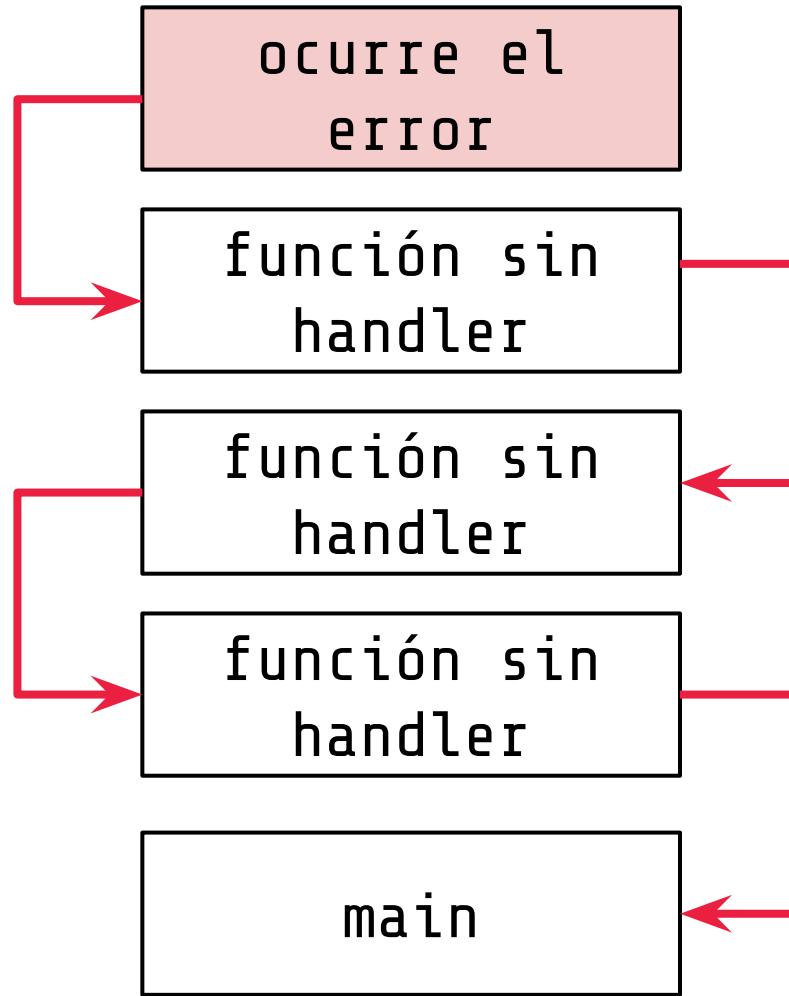
Como funcionan



**Todo viene bien
hasta que...**



Buscando quien se haga cargo



Si nadie lo ataja

Un ejemplo concreto (y simple)

```
public class ExplosionApp {  
  
    public static void main(String[] args) {  
        a();  
    }  
    static void a() {  
        b();  
    }  
    static void b() {  
        c();  
    }  
    static void c() {  
        d();  
    }  
    static void d() {  
        1/0;  
    }  
}
```



¡BOOM!

Esto es lo que veríamos

```
Exception in thread "main" java.lang.ArithmetricException: / by zero
at ar.unrn.ExplosionApp.d(ExplosionApp.java:23)
at ar.unrn.ExplosionApp.c(ExplosionApp.java:19)
at ar.unrn.ExplosionApp.b(ExplosionApp.java:15)
at ar.unrn.ExplosionApp.a(ExplosionApp.java:11)
at ar.unrn.ExplosionApp.main(ExplosionApp.java:7)
```

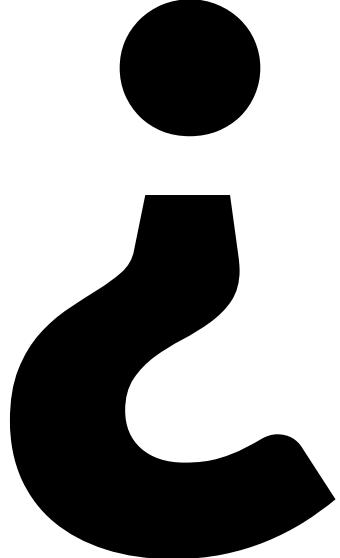


**Esto aplica a cualquier
situación 'fuera de lo
normal'**



¿Preguntas?

**Hasta ahí, es solo otro
ejemplo de como vemos
un error**



**Pero ¿qué
podemos hacer?**



1

Atajar

2

Delegar

Atajadas



Como atajar Excepciones

¡Pavada de atajada!

```
try {  
    argentina.getWorldCup(2023).getFinal().ganar();  
} catch (MbappeException exc){
```



}

**Por ahora, para
saber que falla es ver
la documentación o
*hacerlo fallar***

Y ahora, ¿cuál es la salida?

```
public class ArregloApp {  
    public static void main(String[] args) {  
        int[] numeros = {1, 2, 3};  
        try {  
            System.out.println(numeros[3]);  
            System.out.println("Hola Mundo");  
        } catch (ArrayIndexOutOfBoundsException exc) {  
            System.out.println("Te pasaste");  
        }  
    }  
}
```

Y ahora, ¿cuál es la salida?

```
public class ArregloApp {  
    public static void main(String[] args)  
        int[] numeros = {1, 2, 3};  
        try {  
            System.out.println(numeros[3]);  
            System.out.println("Hola Mundo");  
        } catch (ArrayIndexOutOfBoundsException exc) {  
            → System.out.println("Te pasaste");  
        }  
}
```

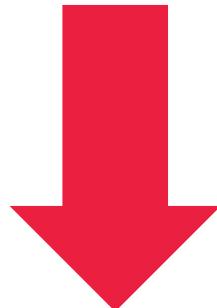


**Por donde va el
programa es
fundamental con las
Excepciones**

Atajar significa que se puede hacer algo

```
try {  
    // Código que puede fallar  
    // ¡Pero también lo que depende!  
}catch (TipoExcepction exc){  
    // Que hacer cuando falle, intentamos recuperarnos  
}
```

Y sigue su curso



**Esta estructura
admite variaciones**

Cuando el código puede fallar de múltiples formas

```
try {  
    argentina.getWorldCup(2023).getFinal().ganar();  
} catch (FranceException exc){  
    // Messi  
} catch (MbappeException exc){  
    // Dibu Martinez  
}
```

Cuando queremos responder de la misma forma

```
catch (MbeppException | FranceException exc) {
```



```
}
```

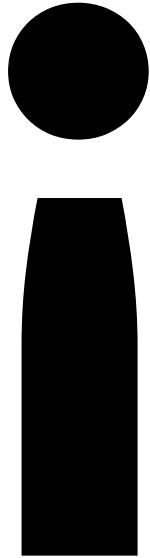
Delegar

En este caso, ¿que podemos hacer?

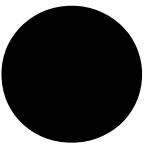
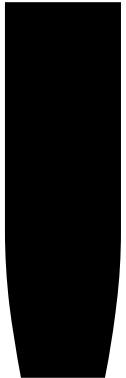
¿Pero quién
puede hacer
algo?

```
int divide(int dividendo, int divisor){  
    return dividendo / divisor;  
}
```





**Son una forma de
dar más
información**





¿Preguntas?

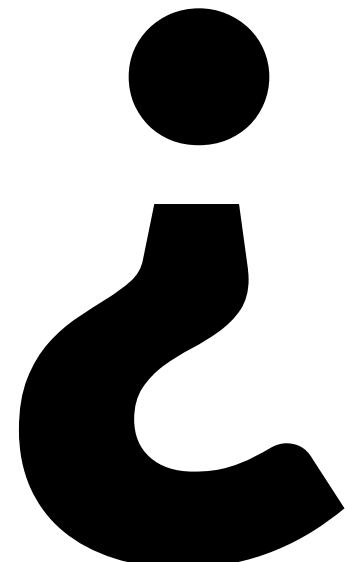
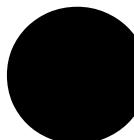
Como regla general

-

Es mejor prevenir
que atajar

**Un
ArrayIndexOutOfBoundsException
es un condicional
que no funcionó**

Ya vamos a ver situaciones que si o si necesitan ataje



Como se prueba esto

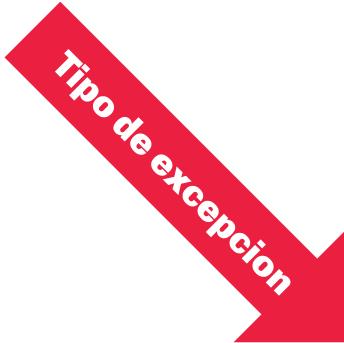
Como para mejorar la División Lenta

```
@Test
void debieraFallar(){
    try{
        var esperado = DivisiónApp.dividir(1,0);
        fail("Debió saltar al catch");
    } catch (ArithmeticException exc){
        ; // un print tambien vá
    }
}
```

Lanzamiento de excepciones



Lanzamiento



```
throw new ArithmeticException("division por cero");
```

Pero

```
public static int divisionLenta(int dividendo, int divisor)  
throws ArithmeticException
```

**De momento vamos
a usar Exceptions
ya creadas**

**En el TP3 hay un
ejemplo**

Clasificación de excepciones

De tiempo de ejecución (RuntimeException)

Que son (a grandes rasgos) prevenibles

ArithmeticException

No dividir por cero

ArrayIndexOutOfBoundsException

No pasarse de los límites

StringIndexOutOfBoundsException

No convertir algo incorrecto

NumberFormatException

**Ver una de este tipo
es *generalmente* un
bug**

de tipo (Exception)

**A grandes rasgos,
inevitables y
tenemos que estar
preparados**

**El uso de archivos
es un buen ejemplo**

Es necesario declarar su lanzamiento

```
public static String[] leerArchivo(String nombre) throws ArchivoException
```

**Pero requiere a que
quien llame se haga
cargo**

Hay más opciones

```
try {  
    String[] datos = leerArchivo("dredd");  
} except (ArchivoException exc) {  
    // ¿podemos cambiar de archivo?  
}
```

**En algunos casos,
no podemos hacer
nada.**

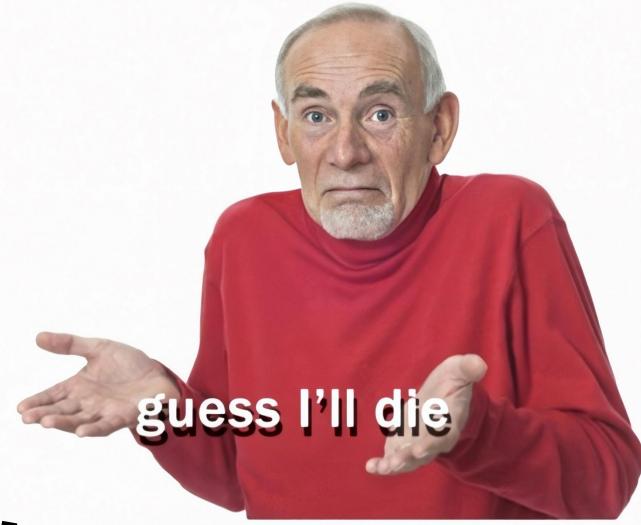
**Que iremos viendo
gradualmente**

-

Errores (Error)

Situaciones que no podemos recuperar

OutOfMemoryError
StackOverflowError



guess I'll die

—

¿Que tipo elegir?

**Qué tipo elegir es
un tema de debate**

**Es preferible que
sean con tipo**

**De momento, todas
de tiempo de
ejecución.**

Continuará...



¿Preguntas?



TP3

Arreglos



try/catch/finally

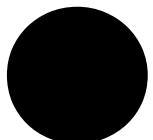
```
try {  
    argentina.getWorldCup(2023).getFinal().ganar();  
} catch (MbappeException exc){  
    // Dibu Martinez  
} finally {  
    // Que hacer en cualquiera de los dos caminos  
}
```

-

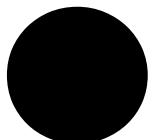
Un ejemplo mas interes

¡Entrada garantizada!

```
Scanner scan = Scanner(System.in, "UTF-8");
String entrada = scan.nextLine();
int numero;
try{
    numero = Integer.parseInt(entrada);
} catch (NumberFormatException exc){
    System.out.printf("%s no era un numero\n, entrada");
}
```

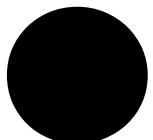


**atajar
o
indicar**

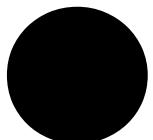


**Quien quiere
hacerlo función**

coding time



**Cuántas veces
piden el ingreso**



**Y si la función
tiene una
cantidad de
intentos**

Lanzamiento

```
throw new NoMasIntentosException();
```

Creación de Excepciones

(por ahora) dentro de la clase

```
public static class NoMasIntentosException extends Exception{  
    public NoMasIntentosException(){  
        super();  
    }  
}
```

Pero, es necesario declararla

```
/**  
 *Pide un número entero, con un mensaje personalizado  
 * y una cantidad limitada de intentos  
 * @throws NoMasIntentosException cuando agotamos intentos  
 */  
public static int pideInt(String mensaje, int intentos)  
    throws NoMasIntentosException {  
    // completar  
}  
}
```



¿Preguntas?

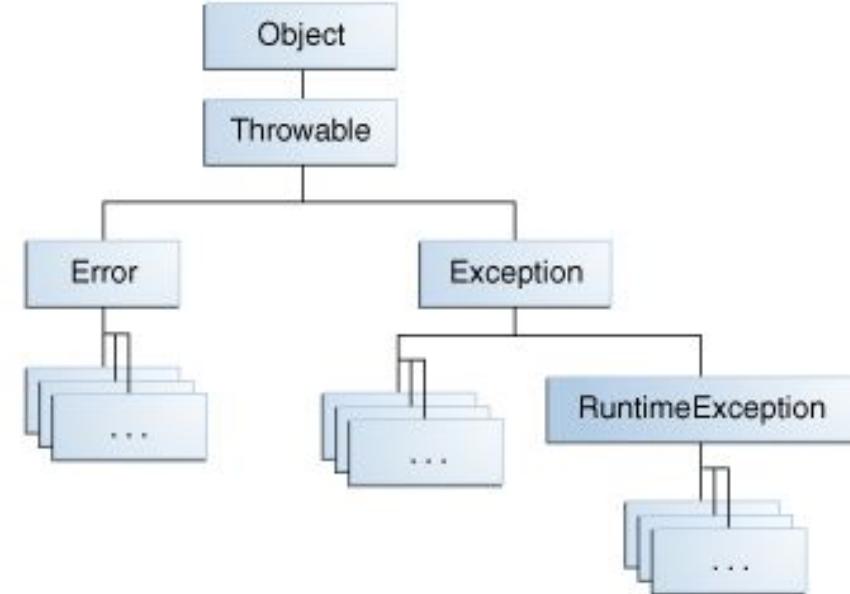
—

¿Qué otras opciones tenemos?



There's three, actually

La familia de Excepciones



RuntimException

Por ejemplo

```
public static class PreconditionException
    extends RuntimeException{
    public NoMasIntentosException(String mensaje){
        super(mensaje);
    }
}
```

```
if (numero < 0){  
    throw new PreconditionException("no puede ser negativo");  
}
```

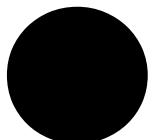
—

Esto no quita que pueda ser atajado

siendo
particularmente
útil para los tests



¿Preguntas?



Observaciones sobre excepciones

**Si solo hacen un
print/printStackTrace**

**Dejen que la
excepción siga su
camino**

**Ya que solo la
“silencian”**

**Con archivos es
particularmente
visibles**

¿Esta función puede cumplir con su objetivo siempre?

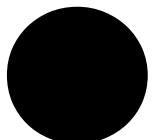
```
public static File crearArchivo(String nombre) {  
    File archivo = new File(nombre);  
    try {  
        archivo.createNewFile();  
    } catch (IOException exc) {  
        exc.printStackTrace();  
    }  
    return archivo;  
}
```

¿Esta función puede cumplir con su objetivo siempre?

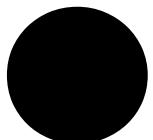
```
public static File crearArchivo(String nombre) {  
    File archivo = new File(nombre);  
    try {  
        archivo.createNewFile();  
    } catch (IOException exc) {  
        exc.printStackTrace();  
    }  
    return archivo;  
}
```

¿Y acá? ¿Que pasa si ‘archivo’ ya existía?

```
public static void escribirInforme(String archivo, String[] informe){  
    File destino = crearArchivo(archivo);  
    escribir(destino, informe);  
}
```



**¿Y si el archivo ya
existía?**



**¿Y si el archivo no
se puede
escribir?**

**Es importante
pensar en el
“usuario” de la
función**

uno mismo

Necesito saber **que** falló y **como** fallo para tomar la decisión correcta

**No apuren la
captura de la
excepción**

Da una falsa sensación de seguridad

```
public static File crearArchivo(String nombre) {  
    File archivo = new File(nombre);  
    try {  
        archivo.createNewFile();  
    } catch (IOException exc) {  
        exc.printStackTrace();  
    }  
    return archivo;  
}
```

**Simplemente, no
hay nada que hacer**

¿Esta función puede cumplir con su objetivo siempre?

```
public static File crearArchivo(String nombre)
                                throws IOException{
    File archivo = new File(nombre);
    archivo.createNewFile();
    return archivo;
}
```

De esta manera cuando lleguen a escribir...

```
public static void escribirInforme(String archivo, String[] informe)
                                    throws IOException{
    File destino;
    try{
        destino = crearArchivo(archivo);
    }catch (FileNotFoundException exc){
        throw new ArchivoNoEncontrado(exc);
    }
    escribir(destino, informe);
}
```



¿Preguntas?

**Como vamos a usar
archivos a futuro,
este tema lo
podemos dejar para
luego**

Aparte

printStackTrace

es un `print` con pasos adicionales
Y como tales, técnicamente no van dentro de las funciones

**Mantengan
separado lo que
interactúa con el
archivo**

**De lo que hace el
trabajo con su
información**

**Es más fácil de
armar tests**

Son mas simples

**Y no siempre
necesitan
interactuar con
archivos**

Excepciones III

Una forma mas completa de armar excepciones

```
public static class LecturaArchivoException extends Exception{  
    public LecturaArchivoException(){  
        super();  
    }  
    public LecturaArchivoException(String razon){  
        super(razon);  
    }  
    public LecturaArchivoException(Throwable excepcionOrigen){  
        super(excepcionOrigen);  
    }  
}
```



¿Preguntas?

Un uso netamente erroneo

Para ‘suavizar’ una excepción

```
try{
    //código que puede fallar
catch (IOException exc){
    throw new RuntimeException(exc);
}
```



¿Preguntas?

<https://docs.oracle.com/javase/tutorial/essential/exceptions/tryResourceClose.html>

Try with resources

Encadenamiento

unrn.edu.ar

UNRN

Universidad Nacional
de Río Negro



| unrnionegro