

CURSO Java Herencia

GRATIS

APUNTATE!!

El uso de Java Finally como clausula que cierra recursos , es algo obligatorio a conocer . Muchas veces surge la pregunta de para qué sirve exactamente Java Finally en nuestro código. Vamos a ver un ejemplo elemental de la división de dos números enteros. Recordemos que es una operación que no se puede realizar y la maquina virtual de Java lanzará una excepción que podemos capturar con un bloque try/catch (hubiera sido mejor comprobar el valor).

```
package com.arquitecturajava;

public class Principal {

    public static void main(String[] args) {

        int a =5;
        int b=0;

        try {
            int resultado=a/b;

            System.out.println(resultado);
        } catch (Exception e) {
```

```
        System.out.println("la aplicacion fallo");

    }
    System.out.println("se cierran los recursos");

    System.out.println("la aplicacion finalizado");

}

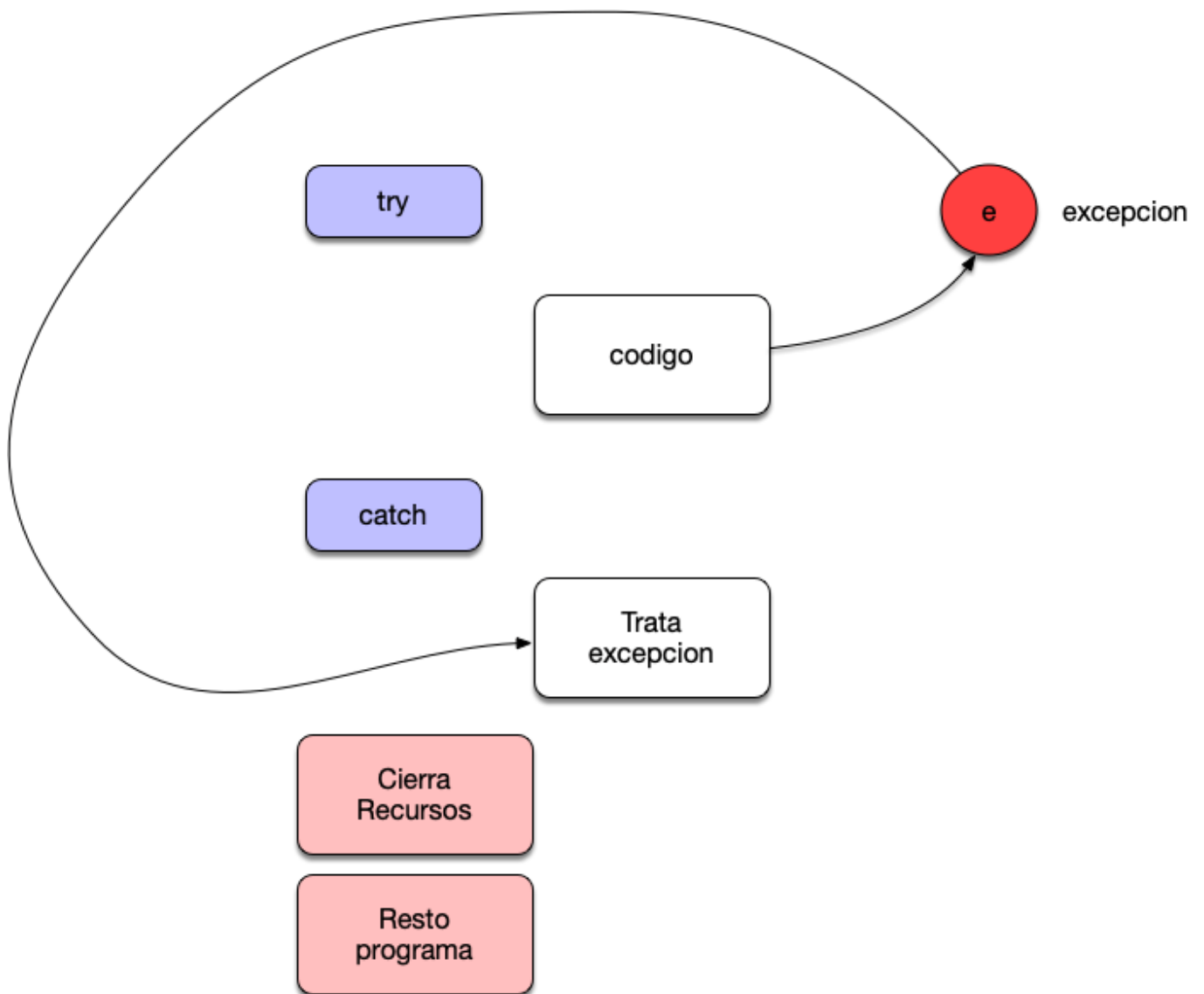
}
```

Java sin Finally

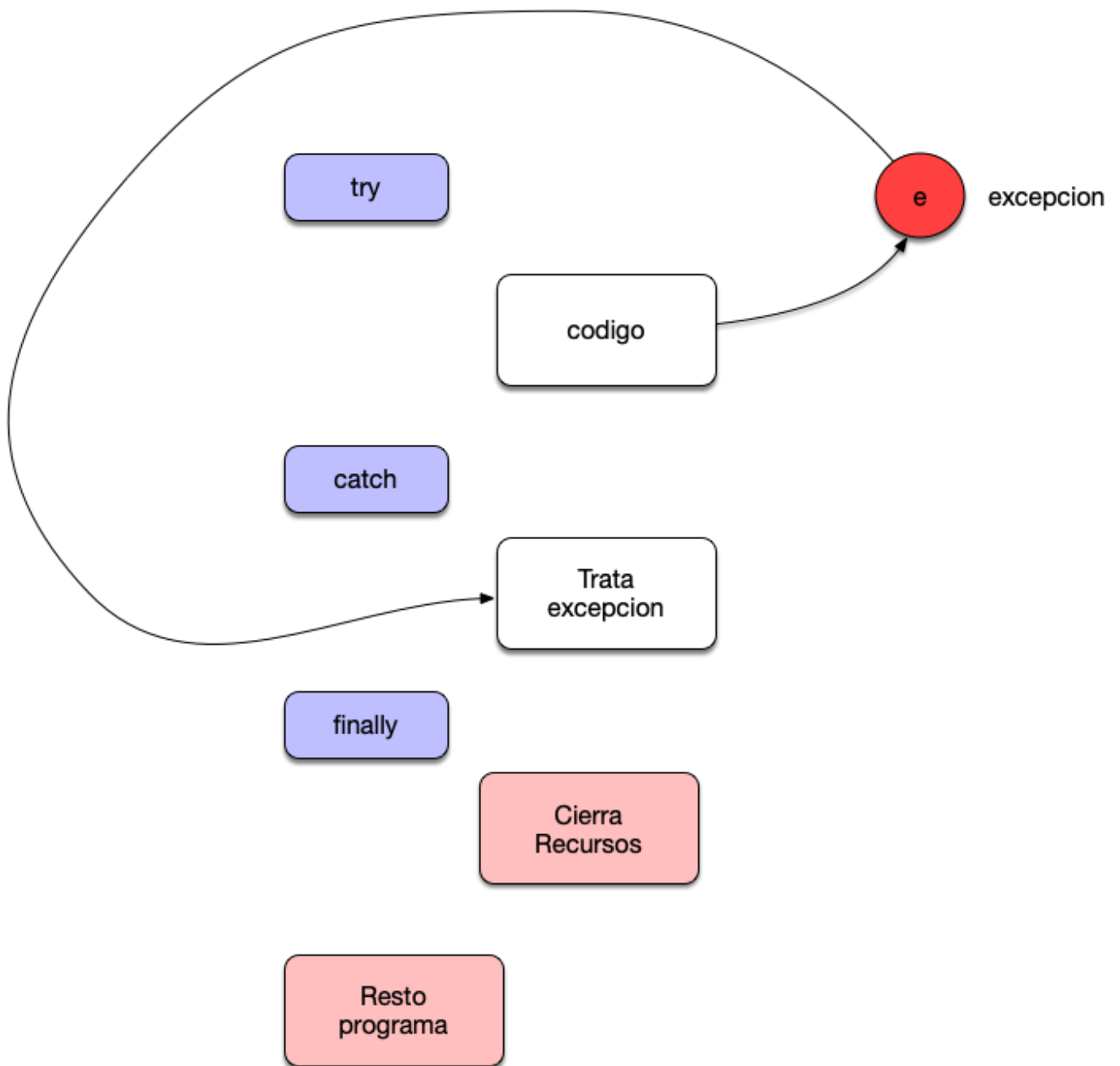
El resultado lo podemos ver en la consola:

```
la aplicacion fallo
se cierran los recursos
la aplicacion finalizado
```

Una excepción salta y es capturada se emite un mensaje de error se cierran los recursos y la aplicación finaliza



Podemos realizar una operación similar con la clausula finally



Todo parece quedar un poco más ordenado

```
package com.arquitecturajava;
```

```
public class Principal2 {  
  
    public static void main(String[] args) {  
  
        int a =5;  
        int b=0;  
  
        try {  
            int resultado=a/b;  
  
            System.out.println(resultado);  
        } catch (Exception e) {  
            // TODO Auto-generated catch block  
            System.out.println("la aplicacion fallo");  
  
        }finally {  
            System.out.println("se cierran los recursos");  
  
        }  
  
        System.out.println("la aplicacion finalizado");  
  
    }  
  
}
```

Sin embargo el resultado es el mismo:

```
la aplicacion fallo  
se cierran los recursos  
la aplicacion finalizado
```

Java Finally

En muchas ocasiones la gente suele preguntar para qué es necesario usar `java finally` . La respuesta es que `finally` asegura siempre el cierre de los recursos y se ejecuta ocurra lo que ocurra en el programa. Por ejemplo supongamos que se produce una excepción dentro de la clausula `catch`.

```
package com.arquitecturajava;
```

```
public class Principal1 {
```

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

```
        int a =5;
```

```
        int b=0;
```

```
        try {
```

```
            int resultado=a/b;
```

```
            System.out.println(resultado);
```

```
        } catch (Exception e) {
```

```
            System.out.println("la aplicacion fallo");
```

```
            throw new NullPointerException();
```

```
        }
```

```
        System.out.println("se cierran los recursos");
```

```
        System.out.println("la aplicacion finalizado");
```

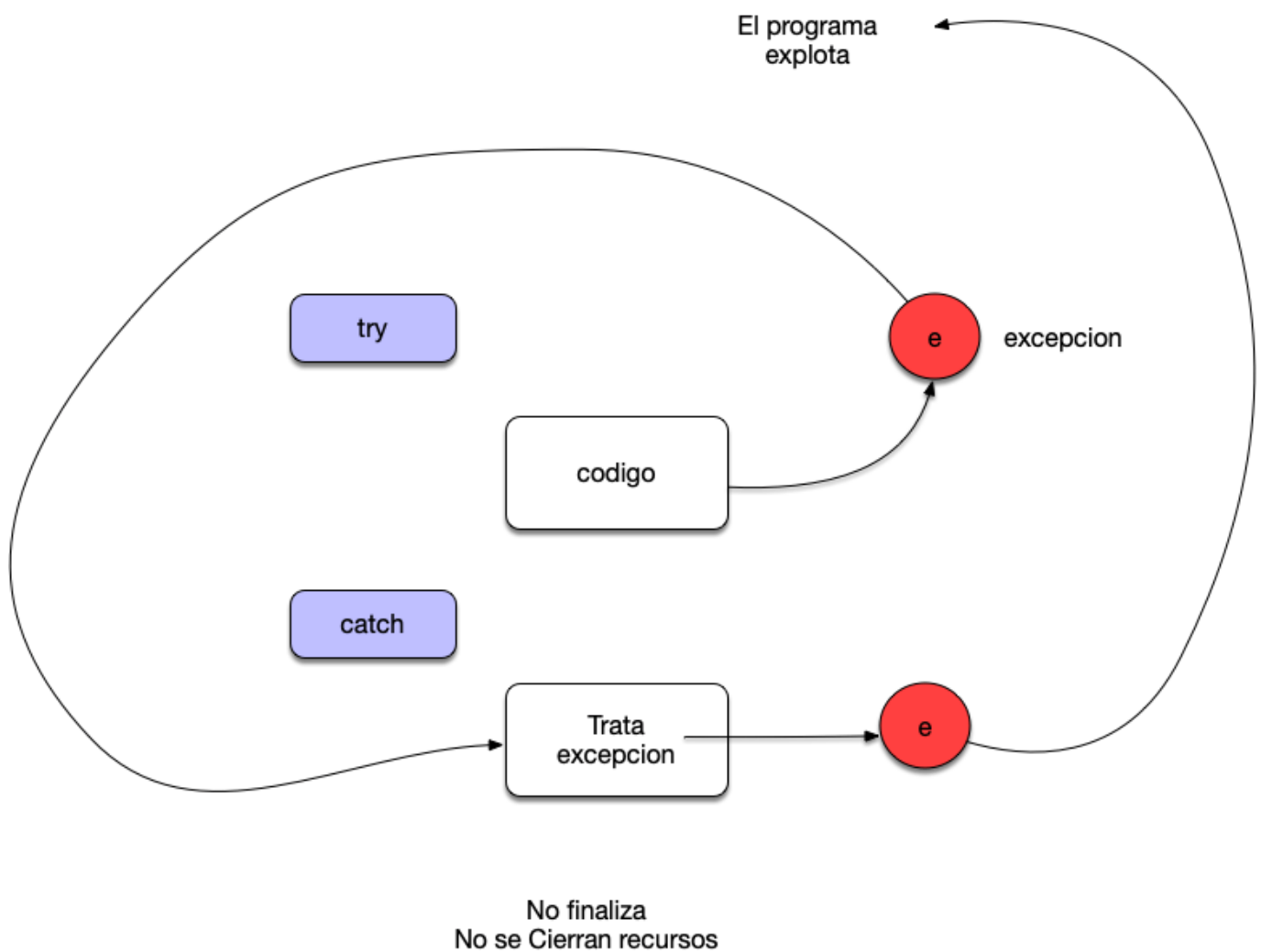
}

}

En este caso el programa finaliza.

```
la aplicacion fallo  
Exception in thread "main" java.lang.NullPointerException  
    at com.arquitecturajava.Principali.main(Principali.java:19)
```

pero no llega a cerrar los recursos y eso supone un problema.



Es algo a veces difícil de ver pero son cosas que ocurren. Si una excepción es lanzada en la

cláusula catch los recursos no se cerrarán . Ahora bien si hemos hecho uso de java finally la situación cambia.

```
package com.arquitecturajava;
```

```
public class Principal3 {
```

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

```
        int a =5;
```

```
        int b=0;
```

```
        try {
```

```
            int resultado=a/b;
```

```
            System.out.println(resultado);
```

```
        } catch (Exception e) {
```

```
            // TODO Auto-generated catch block
```

```
            System.out.println("la aplicacion fallo");
```

```
            throw new NullPointerException();
```

```
        }finally {
```

```
            System.out.println("se cierran los recursos");
```

```
        }
```

```
        System.out.println("la aplicacion finalizado");
```

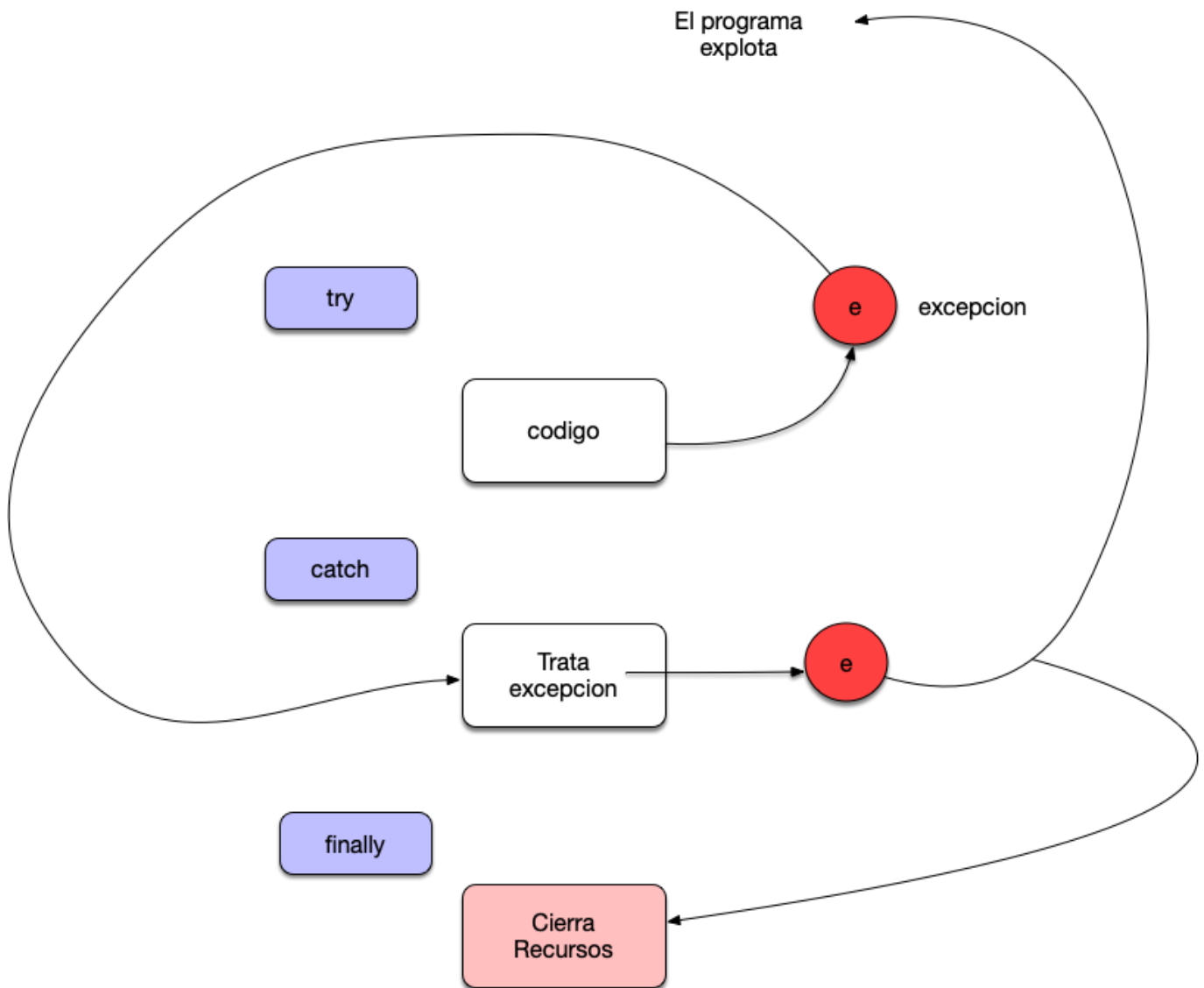
```
    }
```

}

Es cierto que la aplicación ha fallado y revienta

```
la aplicacion fallo  
Exception in thread "main" se cierran los recursos  
java.lang.NullPointerException  
at com.arquitecturajava.Principal3.main(Principal3.java:19)
```

Pero también es cierto que la cláusula finally se ejecuta y cierra los recursos que están abiertos .



De ahí la importancia de hacer uso de la cláusula y entender como funciona correctamente.

Otros artículos relacionados

1. [JDBC, Java try with resources](#)
2. [JDBC Prepared Statement y su manejo](#)
3. [REST HTTP return codes y sus curiosidades](#)
4. [Java 8 Optional y NullPointerExceptions](#)