

try con recursos

Cierre de objetos

- Algunos objetos utilizados para acceder a datos (Connection, PrintStream, BufferedReader, etc.) deben ser cerrados después de su uso.
- Estos objetos exponen el método `close()` para realizar el cierre de los mismos.
- Para garantizar el cierre, la llamada al método `close()` se debe realizar en el bloque `finally`

Cierre clásico de objetos

➤ Ejemplo clásico de uso:

Gran cantidad de código extra para garantizar el cierre de los objetos

```
Connection con=null;
try{
    con=DriverManager.getConnection(...);
    ...
}
catch(SQLException ex){
    //tratamiento excepción
}
finally{
    if(con!=null){
        try{
            con.close();
        }catch(SQLException ex){
            ...
        }
    }
}
```

Interfaz AutoCloseable

- Interfaz del paquete `java.lang` que incorpora un único método llamado `close()`.
- Implementada desde Java 7 por las clases de objetos que gestionan recursos, como las de entrada y salida de `java.io`, o los objetos JDBC de acceso a datos.
- Objetivo: Que sea el propio entorno de ejecución el que llame automáticamente al método `close()` de cualquier objeto `AutoCloseable`, ahorrando código al programador

try con recursos

- Variante del try en la que se crean objetos autocerrables al principio del mismo.
- Al salir del bloque try, tanto de forma natural como por una excepción, los objetos son cerrados automáticamente.
- Sintaxis:

Llamada implícita
variable.close();
antes de abandonar
el bloque try

```
try(Tipo variable=new ...){  
}  
catch(TipoException ex){  
}
```

```
Tipo variable =new ...;  
try(variable){  
}  
catch(TipoException ex){  
}
```

Cierre con try con recursos

➤ Ejemplo de uso:

No se requiere bloque finally para cierre de objetos. Importante ahorro de código

```
try(Connection con=DriverManager.getConnection(...))
...
}
catch(SQLException ex){
    //tratamiento excepción
}
```

➤ Cierre de múltiples objetos:

```
try(FileReader fr=new FileReader("datos.txt");
    BufferedReader bf=new BufferedReader(fr))
...
}
catch(IOException ex){
    //tratamiento excepción
}
```

Son cerrados en **orden inverso a su creación**:
bf.close()
fr.close()

Consideraciones

- La creación del objeto puede realizarse antes del try, indicando después la variable entre paréntesis. En este caso, la variable se trata como constante efectiva:

correcto

```
Connection con=DriverManager.getConnection(...);  
try(con){  
    ...  
}
```

error de compilación

```
Connection con=DriverManager.getConnection(...);  
con=DriverManager.getConnection(...); //error  
try(con){  
    ...  
}
```

- El método `close()` es llamado nada más abandonar el bloque try, antes de entrar en un posible bloque catch o finally.