

# Captura de excepciones



# Bloques try catch

➤ Las excepciones se capturan en un programa java a través de los bloques try catch:

```
try{  
    //instrucciones  
}  
catch(TipoExcepcion1 ex){  
    //tratamiento excepción  
}  
catch(TipoExcepcion2 ex){  
    //tratamiento excepcion  
}
```

# Consideraciones

- No puede haber ninguna instrucción entre los bloques try y catch:

```
try{..}  
System.out.println("hello"); //error de compilación  
catch(...){}
```

- Si se capturan varios tipos de excepciones que tienen relación de herencia entre ellas, los catch de las subclases deben ir antes que los de las superclases:

## Compilación correcta

```
catch(FileNotFoundException ex){  
..  
}  
catch(IOException ex){  
..  
}
```

## Error compilación

```
catch(RuntimeException ex){  
..  
}  
catch(ArithmeticException ex){  
..  
}
```

# Multicatch

➤ Si los catch de varias excepciones van a realizar la misma tarea, podemos agruparlos en un multicatch:

```
catch(IOException ex){  
    System.out.println("error");  
}  
catch(SQLException ex){  
    System.out.println("error");  
}
```



```
catch(IOException|SQLException ex){  
    System.out.println("error");  
}
```

➤ Las excepciones del multicatch no pueden tener relación de herencia, se produciría un error de compilación

# Métodos de exception

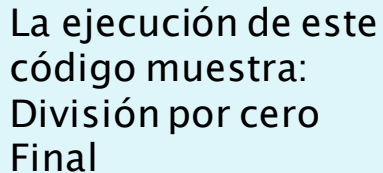
➤ Todas las clases de excepción heredan los siguientes métodos de Exception:

- `String getMessage()`. Devuelve una cadena de caracteres con un mensaje de error asociado a la excepción
- `void printStackTrace()`. Genera un volcado de error que es enviado a la consola

# Bloque finally

➤ Se ejecuta siempre, tanto si se produce la excepción como si no.

```
try{
    int n=4/0;
}
catch(ArithmeticException ex){
    System.out.println("División por cero");
    return;
}
finally{System.out.println("Final");}
```



La ejecución de este código muestra:  
División por cero  
Final

➤ Si se produce una excepción y no hay ningún catch para capturarla, se propagará la excepción al punto de llamada, pero antes ejecutará el bloque finally