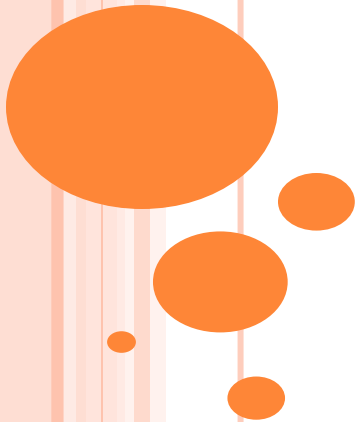


# CAPTURA DE EXCEPCIONES



# Concepto

- Situación anómala que se produce durante la ejecución de un programa
- Cuando se produce una excepción en un punto del programa, se crea un objeto de este tipo de excepción y se "lanza" al programa
- Si este no la trata y la excepción termina llegando a la máquina virtual Java, se interrumpe la ejecución y se muestra en la consola la traza de error:

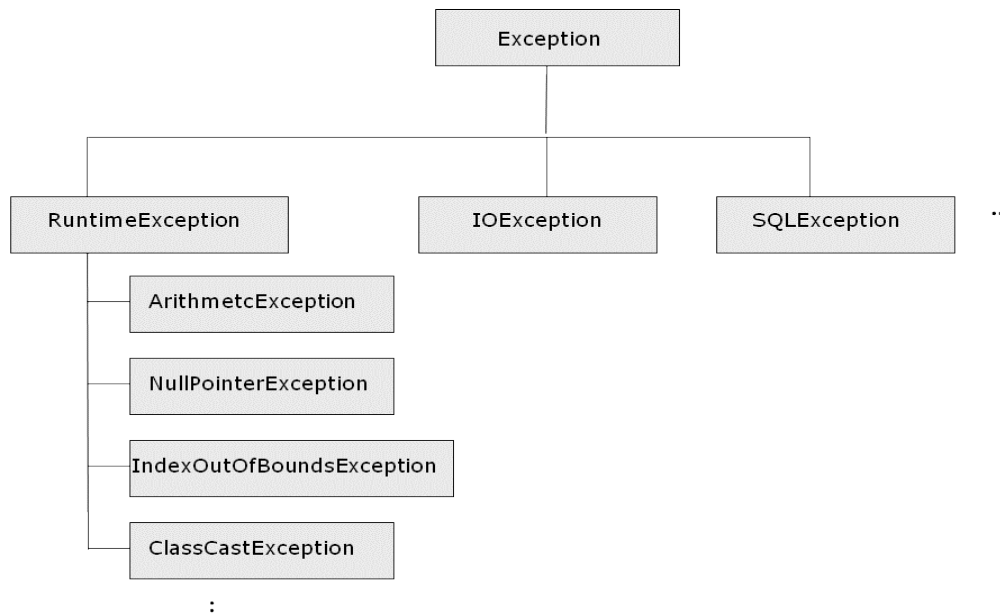
---

```
Exception in thread "main" java.lang.ArrayIndexOutOfBoundsException: 6
    at principal.Primitiva.ordenar(Primitiva.java:35)
    at principal.Primitiva.main(Primitiva.java:18)
```



# Clases de excepciones

➤ Todas las clases de excepción son subclases de **Exception**



# Tipos de excepciones

- **Checked (marcadas).** Excepciones propias de determinadas APIs, y su captura es obligatoria. Ejemplos: `SQLException`, `IOException`..
- **Unchecked (no marcadas).** Excepciones generales que se pueden producir en cualquier programa. No es obligatorio su captura, son subclases de `RuntimeException`. Ejemplos: `NullPointerException`, `ArrayIndexOutOfBoundsException`,..

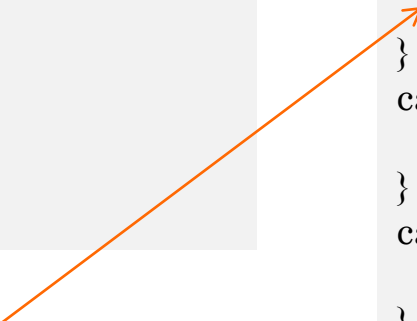


# Bloques try catch

➤ 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  
}
```

La instrucción se ejecuta solo si no hay errores



```
try{  
    Scanner sc=new Scanner(System.in);  
    int a=sc.nextInt();  
    int b=sc.nextInt();  
    int s=a/b;  
    System.out.println("División: "+s);  
}  
catch(InputMismatchException ex){  
    System.out.println("Número incorrecto");  
}  
catch(ArithmeticException ex){  
    System.out.println("División por 0");  
}
```

# 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

➤ Apropiado para cierre de objetos