

Tipus d'errors en Java

En Java (i en general en programació) els errors es poden dividir en **tres grans tipus**

1 Errors de compilació

- **Quan passen:** abans d'executar el programa.
- **Qui els detecta:** el compilador (javac).
- **Exemples:** errors de sintaxi o de tipus.
- **Conseqüència:** el programa **no es pot executar** fins que es corregeixen.

Exemple:

```
int x = "hola"; // ✗ Error de tipus: no pots assignar un String a un int
System.out.println("Hola"
```

L'última línia falta un parèntesi — també seria un error de **sintaxi**.

2 Errors de temps d'execució (Runtime errors)

- **Quan passen:** mentre el programa s'està executant.
- **Conseqüència:** el programa **peta** si no estan controlats.
- **Exemples comuns:**
 - `ArithmeticException` → divisió entre zero
 - `NullPointerException` → accedir a un objecte nul
 - `ArrayIndexOutOfBoundsException` → accedir fora de l'array

Exemple:

```
int a = 10;
int b = 0;
int c = a / b; // ✗ ArithmeticException: / by zero
```

3 Errors lògics

- **Quan passen:** el programa s'executa sense errors, però **no fa el que hauria de fer**.
- **Conseqüència:** resultats incorrectes.
- **El compilador no els detecta.**

Exemple:

```
int preu = 100;
```

```
int descompte = 20;
int total = preu + descompte; // ❌ lògic — hauria de restar, no sumar
```

Gestió d'errors amb try-catch

El mecanisme try-catch serveix per **controlar els errors d'execució** (les excepcions).

Estructura bàsica:

```
try {
    // Codi que pot llançar una excepció
} catch (TipusExcepcio e) {
    // Codi per manejar l'error
} finally {
    // (opcional) Codi que sempre s'executa, passi el que passi
}
```

Exemple 1: capturar una divisió per zero

```
public class ExempleTryCatch {
    public static void main(String[] args) {
        try {
            int x = 10;
            int y = 0;
            int resultat = x / y; // genera ArithmeticException
            System.out.println(resultat);
        } catch (ArithmeticException e) {
            System.out.println("⚠️ No es pot dividir per zero!");
        } finally {
            System.out.println("Bloc finally: això s'executa igualment.");
        }
    }
}
```

Sortida:

```
⚠️ No es pot dividir per zero!
Bloc finally: això s'executa igualment.
```

Exemple 2: múltiples tipus d'excepcions

```
try {
    int[] nums = {1, 2, 3};
    System.out.println(nums[5]); // ArrayIndexOutOfBoundsException
} catch (ArrayIndexOutOfBoundsException e) {
    System.out.println("Error: accés fora dels límits de l'array!");
} catch (Exception e) {
    System.out.println("Error desconegut: " + e.getMessage());
}
```

Si no saps exactament quin tipus d'excepció pot aparèixer, pots capturar la classe **Exception**, que és la mare de totes les excepcions.

Exemple 3: crear una excepció pròpia

Pots definir les teves pròpies excepcions per controlar errors específics del teu programa.

```
class EdatInvalidaException extends Exception {
    public EdatInvalidaException(String missatge) {
        super(missatge);
    }
}

public class ExempleExcepcioPersonalitzada {
    static void comprovarEdat(int edat) throws EdatInvalidaException {
        if (edat < 18) {
            throw new EdatInvalidaException("Edat menor de 18!");
        }
    }

    public static void main(String[] args) {
        try {
            comprovarEdat(15);
        } catch (EdatInvalidaException e) {
            System.out.println("Error: " + e.getMessage());
        }
    }
}
```

Exemple complet: “Calculadora senzilla amb control d’errors”

```
import java.util.Scanner;

public class CalculadoraSegura {

    public static void main(String[] args) {

        Scanner entrada = new Scanner(System.in);

        try {
            // --- 1 Error de lògica (pot passar si el codi està mal dissenyat) ---
            // Aquí suposem que volem sumar, però l'usuari diu "divideix" — si no controlem això,
            farem un càlcul erroni.
            System.out.println("Introdueix el primer nombre:");
            int a = entrada.nextInt();

            System.out.println("Introdueix el segon nombre:");
```

```

int b = entrada.nextInt();

System.out.println("Tria una operació (+, -, *, /):");
char op = entrada.next().charAt(0);

int resultat = 0;

// --- 2 Possibles errors d'execució controlats amb try-catch ---
switch (op) {
    case '+':
        resultat = a + b;
        break;
    case '-':
        resultat = a - b;
        break;
    case '*':
        resultat = a * b;
        break;
    case '/':
        resultat = a / b; // ⚠️ pot llançar ArithmeticException si b == 0
        break;
    default:
        System.out.println("❌ Operació no reconeguda!");
        return;
}

System.out.println("✅ Resultat: " + resultat);
}

// --- 3 Captura d'excepcions específiques ---
catch (ArithmeticException e) {
    System.out.println("⚠️ Error: no pots dividir per zero!");
}
catch (java.util.InputMismatchException e) {
    System.out.println("⚠️ Error: només pots introduir números enters!");
}

// --- 4 Captura genèrica per qualsevol altre error ---
catch (Exception e) {
    System.out.println("⚠️ S'ha produït un error inesperat: " + e.getMessage());
}

// --- 5 Bloc finally (opcional, sempre s'executa) ---
finally {
    System.out.println("Programa finalitzat. Gràcies per utilitzar la calculadora!");
    entrada.close(); // Tanquem l'objecte Scanner
}
}
}

```

Explicació per línies


| Línia | Què fa | Tipus d'error possible |
|---|---|--|
| Scanner entrada = new Scanner(System.in); | Crea l'entrada per llegir dades de l'usuari | (cap) |
| int a = entrada.nextInt(); | Llegeix el primer número | InputMismatchException si escrius text |
| int resultat = a / b; | Divideix dos números | ArithmeticException si b és 0 |
| switch (op) | Decideix quina operació fer | pot tenir error lògic si l'usuari tria un operador no controlat |
| catch (ArithmeticException e) | Captura divisió per zero | controlat |
| catch (InputMismatchException e) | Captura si l'usuari posa una lletra | controlat |
| catch (Exception e) | Captura qualsevol altra excepció | controlat |
| finally | Sempre s'executa (tant si hi ha error com no) | — |

Exemple de sortida

Cas correcte:

Introdueix el primer nombre:
8
Introdueix el segon nombre:
2
Tria una operació (+, -, *, /):
/
Resultat: 4
Programa finalitzat. Gràcies per utilitzar la calculadora!

Cas d'error (divisió per zero):

Introdueix el primer nombre:
8
Introdueix el segon nombre:
0
Tria una operació (+, -, *, /):
/
 Error: no pots dividir per zero!
Programa finalitzat. Gràcies per utilitzar la calculadora!

Cas d'error (entrada no numèrica):

Introdueix el primer nombre:

hola

⚠ Error: només pots introduir números enters!
Programa finalitzat. Gràcies per utilitzar la calculadora!

Exemple amb excepció pròpia

```
import java.util.Scanner;

// Excepció pròpia
class OperacioInvalidaException extends Exception {
    public OperacioInvalidaException(String msg) {
        super(msg);
    }
}

public class Calculadora {
    public static void main(String[] args) {
        Scanner in = new Scanner(System.in);

        try {
            System.out.print("Primer nombre: ");
            int a = in.nextInt();
            System.out.print("Segon nombre: ");
            int b = in.nextInt();
            System.out.print("Operació (+, -, *, /): ");
            char op = in.next().charAt(0);

            int res;
            switch (op) {
                case '+': res = a + b; break;
                case '-': res = a - b; break;
                case '*': res = a * b; break;
                case '/':
                    if (b == 0) throw new ArithmeticException("Divisió per zero");
                    res = a / b;
                    break;
                default:
                    throw new OperacioInvalidaException("Operació no reconeguda");
            }

            System.out.println("Resultat: " + res);

        } catch (ArithmeticException e) {
            System.out.println("⚠ " + e.getMessage());
        } catch (OperacioInvalidaException e) {
            System.out.println("⚠ " + e.getMessage());
        } catch (Exception e) {
            System.out.println("⚠ Error inesperat");
        } finally {
            System.out.println("Programa acabat");
        }
    }
}
```

```

        in.close();
    }
}

```

Resum:

- `try` → on pot passar un error
- `catch` → controla errors concrets (`ArithmeticException`, `OperacioInvalidaException`, etc.)
- `finally` → sempre s'executa
- `throw new ...` → per **llençar** una excepció pròpia

| Bloc | Què fa | Exemple |
|----------------|--|---|
| try | Conté el codi que pot provocar errors. | <code>try { int x = a / b; }</code> |
| catch | Captura i tracta l'error (excepció) si passa dins del <code>try</code> . | <code>catch (ArithmeticException e)</code> <code>{ ... }</code> |
| finally | S'executa sempre , hi hagi error o no (ideal per tancar fitxers, escàners, etc.). | <code>finally { in.close(); }</code> |
| throw | Llença una excepció (fem que passi un error expressament). | <code>throw new</code> <code>ArithmeticException("Divisió per zero");</code> |