

0101010
0100101
1101010

UD9.1- GESTIÓ D'EXCEPCIONS - Captura i propagació

Programació –1er DAW/DAM

0. CONTINGUTS

- Errors i excepcions
- Excepcions a Java
- Captura d'excepcions
- Propagació d'excepcions

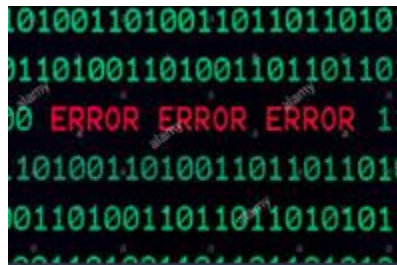
1. Errors i excepcions (I)

- **Error**

- Esdeveniment o situació que es produeix al llarg de l'execució d'un programa impedit-ne dur a terme la seva tasca.
 - *Accedir a una posició d'un array fora de rang*
 - *Referenciar un mètode d'una variable que apunta **a null***

- **Excepció**

- Objecte generat com a conseqüència d'un error; **conté informació** sobre les **característiques de l'error**.

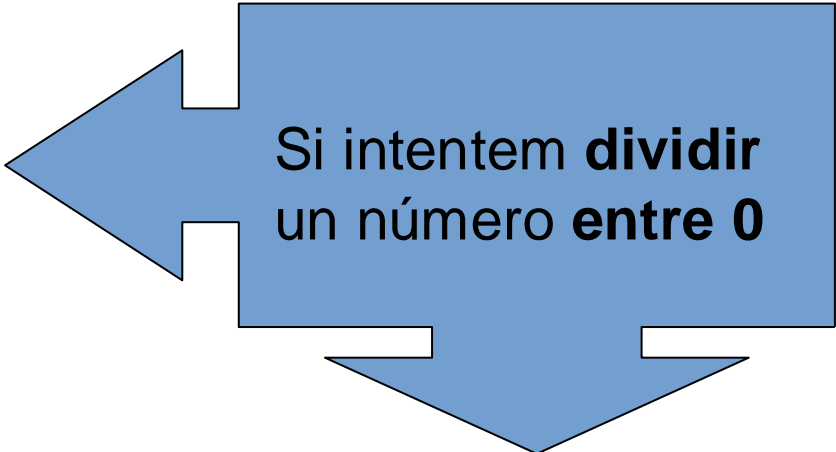


```
Exception in thread "main" java.util.InputMismatchException
    at java.util.Scanner.throwFor(Scanner.java:864)
    at java.util.Scanner.next(Scanner.java:1485)
    at java.util.Scanner.nextInt(Scanner.java:2117)
    at java.util.Scanner.nextInt(Scanner.java:2076)
    at es.coloma.Actividad7.getEdad(Actividad7.java:25)
    at es.coloma.Actividad7.main(Actividad7.java:16)
```

1. Errors i excepcions. Exemple I

- Si el nostre **programa no controla les excepcions**:
 1. S'informarà per consola del problema (*es mostrarà la traça d'execució*)
 2. I es detindrà l'execució del programa.

```
public class TestException {  
    public static void main(String[] args )  
    {  
        System.out.println(1/0);  
    }  
}
```



Si intentem **dividir**
un número **entre 0**

```
/Library/Java/JavaVirtualMachines/jdk1.8.0_221.jdk/Contents/Home/bin/java ...  
Exception in thread "main" java.lang.ArithmeticException: / by zero  
    at es.coloma.App.main(App.java:15)
```

1.1 Errors i excepcions. Exemple II

- Si executem:

```
Scanner lector = new Scanner(System.in);  
System.out.print("Valor:");  
int valor = lector.nextInt();  
System.out.print("Hem llegit:" + valor);
```

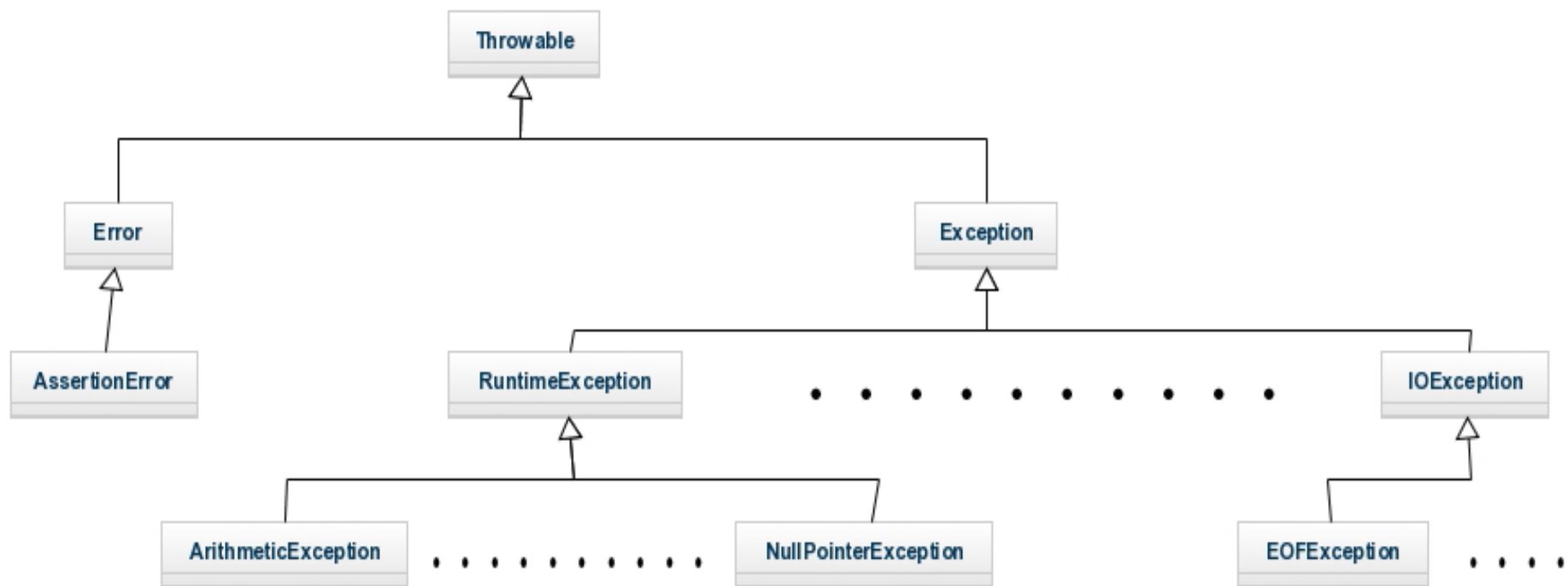
- Si **introduïm** caràcters **no numèrics** es llança una **Excepció**
`InputMismatchException`

```
Valor: hola
```

```
Exception in thread "main" java.util.InputMismatchException  
    at java.util.Scanner.throwFor(Scanner.java:909)  
    at java.util.Scanner.next(Scanner.java:1530)  
    at java.util.Scanner.nextInt(Scanner.java:2160)  
    at java.util.Scanner.nextInt(Scanner.java:2119)  
    at unidad07.SinExcepcion1.main(SinExcepcion1.java:16)
```

2. Excepcions a Java

- Java llança excepcions com a resposta a situacions poc habituals.
- Les excepcions també són classes:



2.1. Tipus d'excepcions

La classe *Exception* és una classe genèrica que representa totes aquelles **excepcions** que **poden ser tractades** (*manejades pel programador*).

- S'agrupa en dos grups principals:
 - **Excepcions implícites:** el programador **NO té obligació** de capturar i gestionar. Estan agrupades a la classe `RuntimeException`.
 - **Excepcions explícites:** Estem **obligats** a tractar-les.

Atenció: Les subclasses `java.lang.Error`, representen un **problema seriós** en l'aplicació que mai haurem de tractar al nostre programa Ex. `VirtualMachineError`

2.2 Excepcions implícites

- **Java** les llança de **forma automàtica**.
- Es tracta de subclasses de `RuntimeException`
- **No cal** realitzar un **tractament explícit** (Java no obliga a tractar-les).
- Aquestes excepcions **indiquen al programador** quins tipus d'**error** té el **programa** perquè ho solucione abans de continuar.
- Exemples:
 - Quan se sobrepassa la dimensió d'un array, es llança una excepció ***ArrayIndexOutOfBoundsException***.
 - Quan s'utilitza una referència a un objecte que encara no ha estat creat, es llança l'excepció ***NullPointerException***.

2.2 Excepcions implícites. Exemples

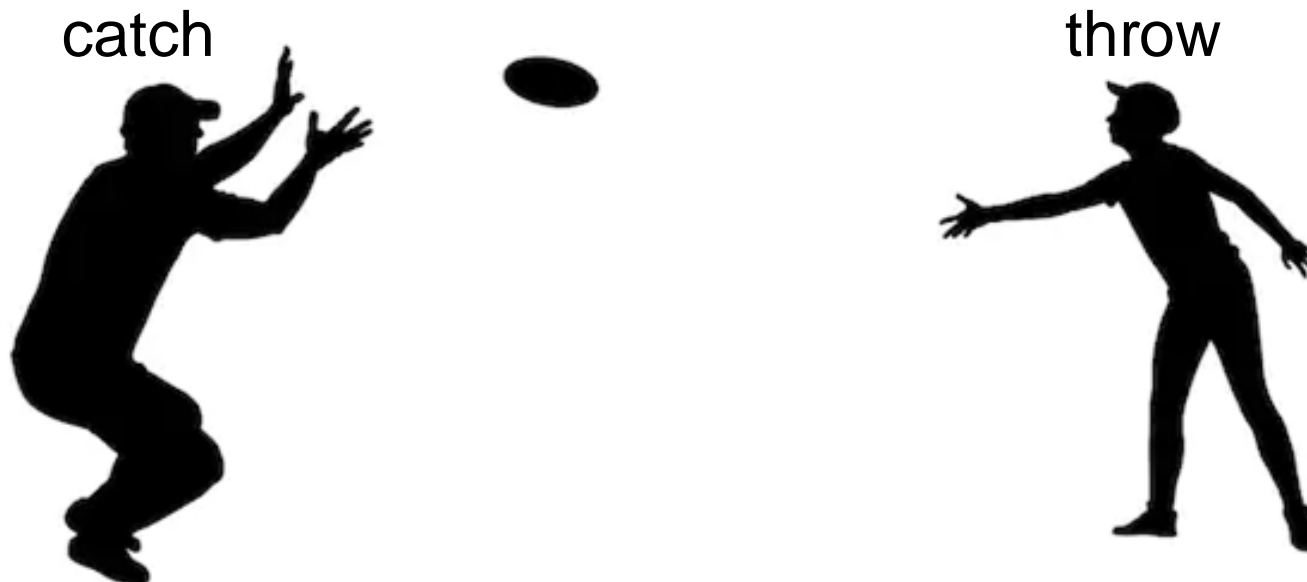
classe	Situació d'excepció
NumberFormatException	Una aplicació intenta convertir una cadena a un tipus numèric, però no té el format apropiat. <code>Integer.parseInt("Un");</code>
IndexOutOfBoundsException	Indica que un índex d'algun tipus (un matriu, cadena) està fora de rang. <code>int[] números = new int[10];</code> <code>números[11] = 25;</code>
NegativeArraySizeException	Si una aplicació intenta crear un array amb mida negativa. <code>int[] números = new int[-10];</code>
NullPointerException	Quan una aplicació intenta utilitzar <i>null</i> on es requereix un objecte. <code>Persona alumne;</code> <code>alumne.saluta();</code>
InputMismatchException	Llançada per Scanner per indicar que el valor recuperat no coincideix amb el patró esperat. <code>Scanner scanner = new Scanner(System.in);</code> <code>scanner.nextInt(); // << "cinc"</code>

2.2 Excepcions explícites

- Si es produeixen, és **obligat** el seu **tractament**.
- **No** es tracta d'un **error de programació** sinó que representen una **situació anòmla** que ha de ser **tractada** per l'**aplicació**.
- Exemple:
 - ***FileNotFoundException***: no hi ha el fitxer al qual s'intenta accedir.
 - ***IOException***: es produeix un error d'entrada/eixida(exemple de *FileInputStream* o de *BufferedReader*).
 - ***TimeoutException***: el servei o l'operació que s'està fent ha superat el temps màxim assignat.
 - ...

3. Captura d' excepcions

- El tractament d'excepcions permet **controlar** els possibles **errors** d'una aplicació **amb elegància**.
- Quan es produeix una **Excepció** podem definir **manejadors** que s'encarreguen de **capturar-la** i realitzar les accions necessàries que permeten **recuperar el flux d'execució normal** de l'aplicació (evitant que es 'trenque')



3. Captura d' excepcions

- Exemples:
 - **Introduir una dada errònia** → Tornar a sol·licitar una dada.
 - **No poder connectar-se a un servei de pagament** → Reintentar connexió.
 - **Realitzar una acció no permesa** → Escriure en un **arxiu Log**.
 - **Enviar un formulari on falten dades** → Mostrar un **missatge a l'usuari** i tornar a **executar una acció**.
- Avantatges:
 - Separació de la **lògica de l'aplicació** del codi de **gestió d'errors**.
 - Facilita la **lectura, depuració i manteniment** del codi.

3.1 L'estructura `try...catch`

- Permet encapsular i protegir els **blocs de codi** que podrien **provocar errors**:
 - Cada bloc (`try`) té un o més controladors (*capturadors*) associats (`catch`).
 - Cada **controlador** especifica el **tipus d'excepció** que controla.

```
Scanner input =new Scanner(System.in);
try{
    int num = input.nextInt();
    int aux = 1/num;
} catch(InputMismatchException e) {    //controlador 1
    System.out.println("Ha d'introduir un número");
} catch(ArithmeticException e) {      //controlador 2
    System.out.println("No és possible dividir per zero");
}
```

3.1.1 Com utilitzar l'estructura *try ... catch*

- El bloc *catch* **tracta l'excepció**: és on podem especificar les accions que s'han de dur a terme:
 - cada *catch* tracta un **tipus d'excepció**.

```
...  
  
}catch(InputMismatchException e) {  
    System.out.println("Ha d'introduir un número");  
}catch(ArithmeticException e) {  
    System.out.println("No és possible dividir per zero");  
}
```

- Quan es produeix una excepció, es cerca **el primer catch** coincident amb el **tipus d'excepció** que s'ha produït.
 - > L'**últim catch** ha de ser el que capture les excepcions **més genèriques**.

3.2 Exemples (I)

```
public static void main(String[] args) {  
  
    String [] text = {"Un", "Dos", "Tres", "Quatre", "Cinc"};  
  
    for (int i = 0; i < 10; i++) {  
        try {  
            System.out.println("index" + i + "=" + text[i]);  
        } catch (ArrayIndexOutOfBoundsException e) {  
            System.out.println ("Fallada a l'índex" + i);  
        }  
    }  
}
```

3.2 Exemples (II)

```
Scanner scanner = new Scanner(System.in);  
try{  
    System.out.print("Valor:");  
    int valor = scanner.nextInt();  
    int auxiliar = 8 / valor;  
    System.out.println(auxiliar);  
} catch(ArithmeticException e) {  
    System.out.println("Divisió per zero");  
}
```


3.3 El bloc *finally*

- És **opcional**: si s'inclou, el codi que conté **s'executa sempre**.
- **Exemple d'ús**: codi de "neteja", *com el que s'utilitza per tancar arxius o una connexió xarxa*.

```
try {  
    //Codi que pot provocar errors  
} catch(ExceptionA a) {  
    //Controlador Excepcions de tipus A  
} catch(ExceptionB b) {  
    //Controlador Excepcions de tipus B  
} finally{  
    //Codi que s'executa sempre  
}
```

3.3 El bloc `finally`. Exemple (I)

```
Scanner lector = new Scanner(System.in);
try{
    System.out.print("Valor:");
    int valor = lector.nextInt();
    int auxiliar = 8 / valor;
    System.out.println(auxiliar);
} catch(ArithmeticException e1) {
    System.out.println("Divisió per zero");
} catch(InputMismatchException e2) {
    System.out.println("No s'ha llegit un sencer....");
} catch(Exception e3) {
    System.out.println("Error general");
} finally{
    lector.nextLine(); //Netegem la darrera lectura
}
```

3.4 Alguns mètodes de la classe *Exception*


- `String getMessage()`
 - Recupera el missatge descriptiu de l'excepció o una indicació específica de l'error produït.
- `String toString()`
 - Escriu una cadena sobre l'error. Normalment indica el nom de la classe d'excepció i el text de `getMessage()`.
- `void printStackTrace()`
 - Escriu el mètode i el missatge de l'excepció (és el que s'anomena **informació de pila**)
 - És el **mateix missatge** que mostra l'executor (*JVM*) quan **no es controla l'excepció**.

```
try {  
    ...  
} catch(IOException ioe) {  
    System.out.println(ioe.getMessage());  
    System.out.println(ioe);  
    ioe.printStackTrace();  
}
```

3.5 Directrius per capturar excepcions

1. **Manejar excepcions específiques:** no declarar manejadors per a la classe `Exception` ja que estarem **ocultant qualsevol error** que es produeix, **impossibilitant** així la seua **detecció**.
2. **Organitzar els blocs *catch*** des dels més específics fins als més generals.

```
Try {  
    Integer number = null;  
    number.toString();  
} catch (Exception i) {  
    System.out.println("Captura qualsevol tipus d'excepció");  
} catch (NullPointerException npe) {  
    System.out.println("Ja ha estat capturada");  
} finally {  
    System.out.println("S'executa sempre independentment de si s'ha executat algun bloc catch");  
}
```



3.5 Directrius per capturar excepcions

- No utilitzar **excepcions** per implementar la lògica del programa.
- **No ignorar excepcions:** tot bloc **catch** ha de tenir una implementació (no hi ha que deixar blocs buits).

```
try {  
    int i = 0;  
    int[] númerosPrimos = {1,2,34,12,56};  
    while(true) {  
        System.out.println(numerosPrimos[i++]);  
        // no l'hem d'utilitzar perquè acabe el bucle  
    }  
} catch(ArrayIndexOutOfBoundsException ex) {  
    // ha de tenir una implementació  
}
```



3.6 Com saber si un mètode llança excepcions

a) Consultant la documentació oficial ([exemple](#))

nextInt

```
public int nextInt()
```

Scans the next token of the input as an int.

An invocation of this method of the form `nextInt()` behaves in exactly the same way as the invocation `nextInt(radix)`, where `radix` is the default radix of this scanner.

Returns:

the int scanned from the input

Throws:

`InputMismatchException` - if the next token does not match the *Integer* regular expression, or is out of range

`NoSuchElementException` - if input is exhausted

`IllegalStateException` - if this scanner is closed

b) Observant els errors que es produeixen

```
/Library/Java/JavaVirtualMachines/jdk1.8.0_221.jdk/Contents/Home/bin/java ...
```

```
Exception in thread "main" java.lang.ArithmeticException: / by zero
    at es.coloma.Actividad3.main(Actividad3.java:13)
```

```
Process finished with exit code 1
```

```
|
```

Activitats prèvia

- **Activitat 1.** Realitza un programa que demane 6 números per teclat i ens diga quin és el màxim. Si l'usuari introdueix una dada errònia (que no siga un **nombre sencer**) el programa ha d'indicar-ho i ha de tornar a demanar el número.

4. Propagació d' excepcions

- Quan es produeix una excepció podem:
 1. **Capturar-ho al propi mètode** on s'ha produït.
 2. **O deixar que es propague** cap al **mètode invocador** perquè ho capture.
 - Si no el captura, es propagarà successivament fins al mètode *main()* .
 - Si no és capturat per ningú, es propaga fins al sistema operatiu i **finalitza l'execució del programa**.

Recorda. Només es propaguen **automàticament** les excepcions que deriven de la classe ***RuntimeException***.

4.1 Exemple I

```
public class TestException {  
  
    public static void main (String [] args) {  
        try {  
            int n = llegirNumero();  
            System.out.print("Número introduït: " + n);  
        } catch(NumberFormatException e) {  
            System.out.println ("Error. Captura de l'excepció en main.");  
        }  
    }  
  
    public static int llegirNumero () {  
        Scanner reader = new Scanner(System.in);  
        int num = 0;  
        try {  
            System.out.print("Insereix un número:");  
            num = Integer.parseInt(reader.nextLine());  
            System.out.println ("Número correcte");  
        } catch(NumberFormatException e) {  
            System.out.println("Error. Només es poden introduir números.");  
        }  
        return num;  
    }  
}
```

Es necessari aquest catch?

Comprova que aquesta excepció, si no es captura ací, es propagaria automàticament

Què mostra aquest programa si introduïm una lletra?

4.1 Exemple II

```
public static void main(String [] args) {  
    try {  
        int n = llegirNumero();  
        System.out.print("Número introduit: " + n);  
    } catch(NumberFormatException e) {  
        System.out.println ("Error. Captura de l'excepció en main.");  
    }  
}  
  
public static int llegirNumero() {  
    Scanner reader = new Scanner(System.in);  
    int num = 0;  
    System.out.print("Insereix un número:");  
    num = Integer.parseInt(reader.nextLine());  
    System.out.println ("Número correcte");  
    return num;  
}
```

Què mostra aquest programa si introduïm una lletra?

Activitat Prèvia

Activitat 2. Modifica l'**activitat 1** de manera que es dispose d'un mètode `obtenirEnter()` que demane a l'usuari que introdisca un número enter. Aquest mètode serà cridat des del mètode principal `main()` tantes vegades com calga perquè l'usuari introdisca 6 números enters. Al final s'haurà de mostrar el més gran.

Nota: El tractament de l'excepció per controlar que l'usuari no introdisca una dada errònia ha de realitzar-se en el mètode `main()`

Això és tot... de moment :-)