Unidad 7: anexo

Excepciones

Diseño y manejo de excepciones

Contenido

- Excepciones en Java
- Jerarquía
- Diseño o declaración de nuevas excepciones
- Lanzamiento de excepciones
- Gestión de excepciones

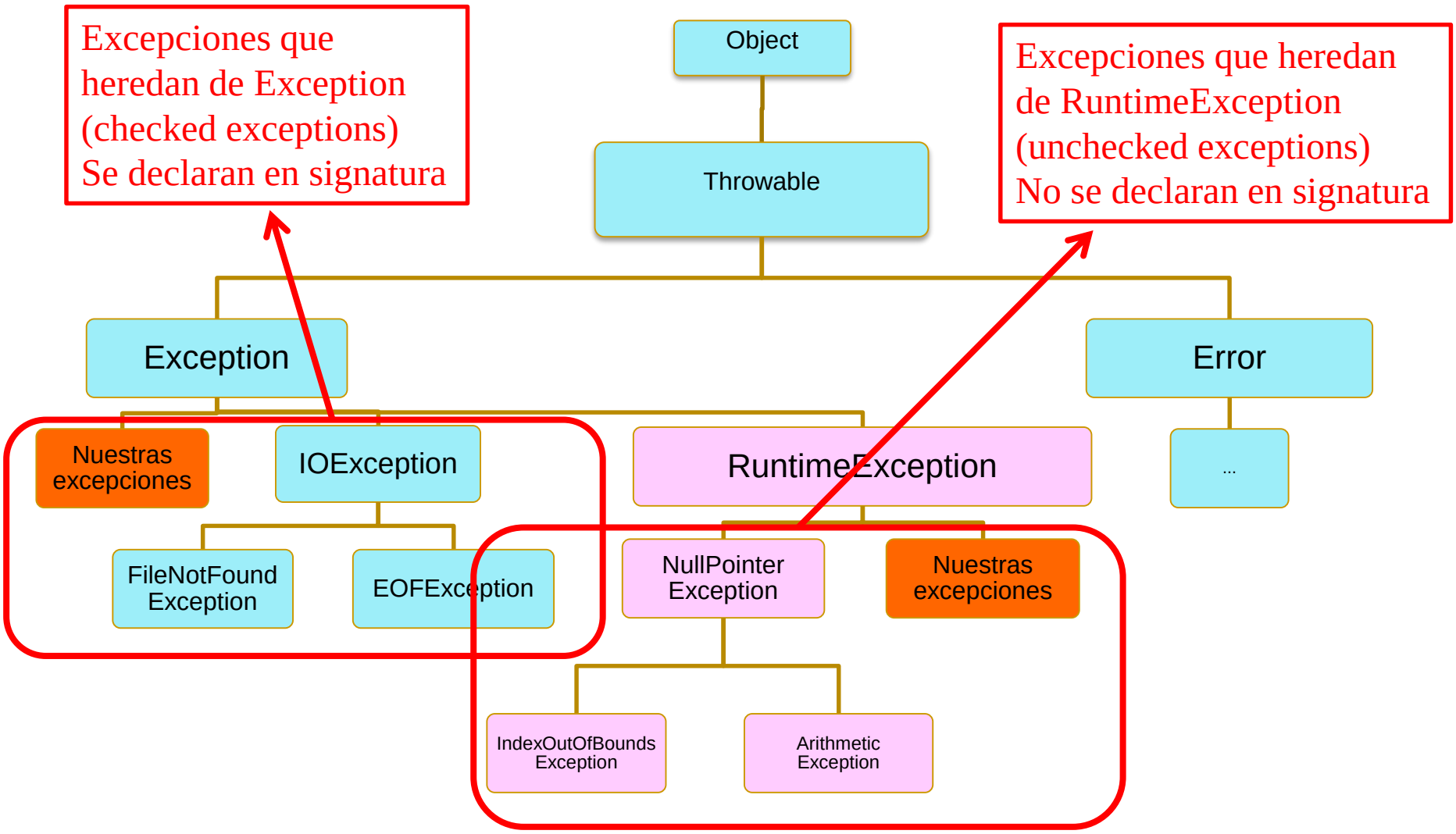
Excepciones en Java

- Son estructuras de control
  - Alteran el flujo del programa en que se ejecutan.
- Destinadas a la detección y recuperación de errores
  - Ante una situación inesperada en la aplicación se *lanzará/elevará* una excepción.
- Las excepciones se representan mediante objetos
  - El API de Java tiene clases definidas para muchas de las excepciones más comunes (error en el manejo de ficheros, índice fuera de los límites de array o vector, error aritmético como división por cero…)
  - También podemos definir nuestras propias excepciones mediante la declaración de clases con tal fin.
- Las excepciones son *lanzadas* por aquellos métodos en los que ocurre una situación anómala
  - Unas tienen que ser gestionadas *obligatoriamente* y otras no.

Excepciones en Java: *Throwable*

- Todas las excepciones tienen como clase base *Throwable*, incluida en el paquete *java.lang*, y sus métodos son:
  - *Trowable(String mensaje)*; Constructor. La cadena es opcional
  - *Throwable fillInStackTrace();* Llena la pila de traza de ejecución.
  - *String getLocalizedMessage();* Crea una descripción local de este objeto.
  - *String getMessage();* Devuelve la cadena de error del objeto.
  - *void printStackTrace(PrintStream_o_PrintWriter s)*; Imprime este objeto y su traza en el flujo del parámetro s, o en la salida estándar (por defecto).
  - *String toString;* Devuelve una breve descripción del objeto.

Jerarquía de excepciones



Declaración de nuevas excepciones

- Implementaremos las clases excepción basándonos en las ya definidas en el API mediante herencia (extends),

```
public class EdadNegativaException extends RuntimeException {
        public  EdadNegativa ( ) {
                super();
        }
        public  EdadNegativa (String texto ) {
                super(texto);
        }
}
```

Lanzamiento o disparo de excepciones

- ***throw***: sirve para disparar.

```
if(condicion_de_disparo) throw new MiExcepcion("textoExplicativo");
```

- ***throws***: sirve para declarar en las signaturas de los métodos las excepciones que se lanzarán (las que extiendan de Exception)

```
Tipo nombre-método (parám-formales) throws ClaseException1, ClaseException1,…
{
        cuerpo
}
```

Lanzamiento de excepciones

- Lanzamiento automático por parte del sistema
  - Ante una situación de error el sistema lanzará una excepción

```
public static void main(String[] args) {
        Integer n = 3/0;
}
```

lanza la excepción ArithmeticException

```
Exception in thread "main" java.lang.ArithmeticException: / by zero
at main(java:15)
```

Lanzamiento de excepciones

- Lanzamiento de excepciones propias
  - Definimos nuestras propias excepciones e indicamos cuándo lanzarlas. La sintaxis para el lanzamiento de excepciones es:

```
throw objetoExcepcion
```

```
public void setEdad(Integer edad){
   if (edad < 0){
         throw new EdadNegativaException("Edad menor que cero");
   }
   this.edad = edad;
}
```

objeto excepción

Lanzamiento de excepciones

- Ejemplo de lanzamiento de la excepción

```
public static void main(String[] args) {
        Persona p = new PersonaImpl();
        p.setEdad(-3);
}
```

como no se ha gestionado el programa acaba así

```
Exception in thread "main" EdadNegativaException:
Edad menor que cero
at Persona.Edad(java:67) at main(java:20)
```

Mensaje que se pasa como parámetro al constructor

Gestión de excepciones

- Distinguimos entre excepciones que hay que capturar obligatoriamente (heredan de Exception (checked exceptions)) y las que no (heredan de RuntimeException (unchecked exceptions))
- La captura de excepciones se hará mediante el bloque *try/catch* con la siguiente sintaxis:

```
try{
  sentencias susceptibles de generar excepciones
}
catch (ExceptionType1 e){
   sentencias a ejecutar en caso de que se haya lanzado ExceptionType1
}
 catch (ExceptionType2 e){
   sentencias a ejecutar en caso de que se haya lanzado ExceptionType2
}
....
finally {
   sentencias a ejecutar con cualquier excepción
}
```

Cláusula catch múltiple

- Característica multi-`catch` de JavaSE7
  - Si después de un bloque try hay varios bloques catch que manejan excepciones distintas pero tienen cuerpos idénticos, se puede usar la característica multi-catch para atrapar esos tipos de excepciones en un solo manejador catch:

```
try{
  sentencias susceptibles de generar excepciones
}
catch (ExceptionType1 | ExceptionType2 | ExceptionType3 e){
   sentencias a ejecutar en caso de que se haya lanzado ExceptionType1 o
ExceptionType2 o ExceptionType3
}
....
finally {
   sentencias a ejecutar con cualquier excepción
}
```

Gestión de excepciones

```
public static void main(String[] args) {
        Persona p = new PersonaImpl();
        try {
            p.setEdad(-3);
            p.setAnioNacimiento(2025);
        }
        catch(EdadNegativaException e){
                mostrar("Se usó una edad no válida: " + e);
        }
        catch(FechaNoValidaException e){
                mostrar("Se usó un anio incorrecto: " + e);
        }
        finally{
                mostrar("Se usaron valores incorrectos para la persona p");
        }
}
```

Gestión de excepciones

- Para aquellas excepciones que hay que gestionar obligatoriamente (las que heredan de *Exception*) usaremos el bloque *try/catch* o bien la sentencia *throws* en la cabecera del método.

```
public Punto clone(){
        Punto copia=null;
        try{
                copia = (Punto)super.clone();
        }catch(CloneNotSupportedException e){e.printStackTrace();}
        return copia;
    }
```

```
public Punto clone() throws CloneNotSupportedException {
        Punto copia=null;
        copia = (Punto)super.clone();
        return copia;
    }
```

Resumen

- Si el método ***m1*** llama a ***m2*** entonces la gestión de las excepciones disparadas por ***m2*** se hace en el código de ***m1***, mediante las cláusulas ***try/catch/finally***
- Si ***m1*** no gestiona las excepciones generadas (disparadas) por ***m2*** las debe propagar al método que llamó a ***m1***.
- Si ***m2*** declara en su signatura la posibilidad de generar excepciones mediante ***throws***, entonces m1 tiene obligación en su código de gestionar todas las excepciones declaradas.
- Las sentencias del bloque ***finally*** se ejecutan siempre al final se haya disparado excepción o no.
- ***finally*** y ***catch*** son opcionales, pero al menos debe haber un bloque ***finally*** o uno ***catch*** detrás del bloque ***try***.
- **Nota:** En los constructores es mejor lanzar las excepciones para evitar construir objetos mal formados.

Recomiendación

- Se recomienda no abusar del sistema de excepciones como control de flujos simples, sino utilizarlo sólo en aquellos estados del programa que realmente creen un problema de ejecución que pueda ser letal para el programa.