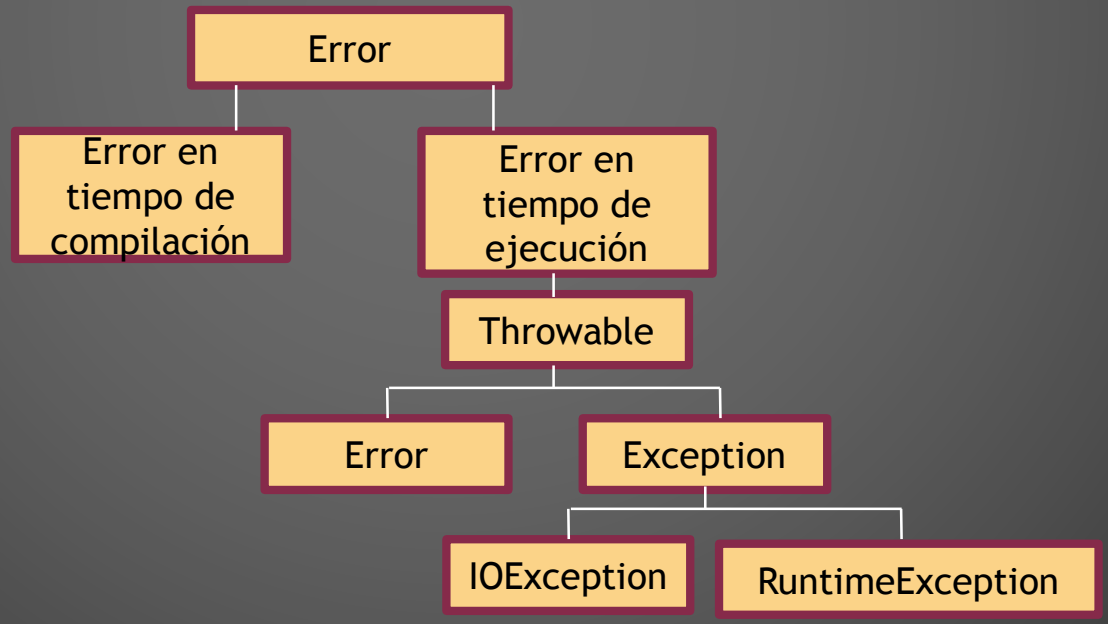
USO DE EXCEPCIONES EN JAVA

- Una excepción es una situación no esperada en la ejecución de un programa.
- En java se cuenta con varias clases de Excepciones ya creadas en la API para resolver varios problemas, sin embargo si no existiese podemos crear una a la medida por nosotros mismos.
- Al programar nos pueden ocurrir diversos errores y hay que clasificarlos

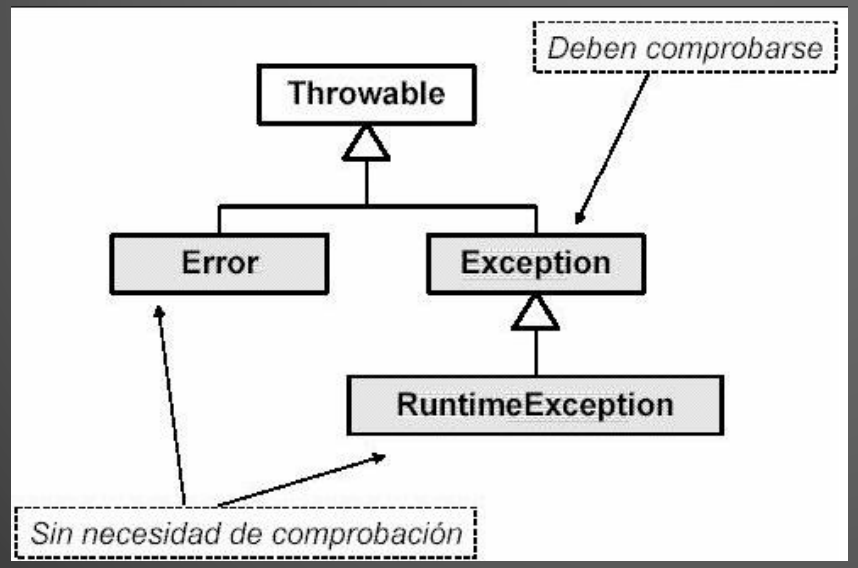


USO DE EXCEPCIONES EN JAVA

- Existen dos tipos de excepciones: controladas y no controladas.

• Cualquier clase que derive de Error o de RuntimeException es NO CONTROLADA. No es necesario capturarlas ni declararlas aunque si se puede.

- Cualquier clase que derive directamente de Exception excepto RuntimeException ES CONTROLADA. Es obligatorio capturarlas o más bien dicho tratarlas.

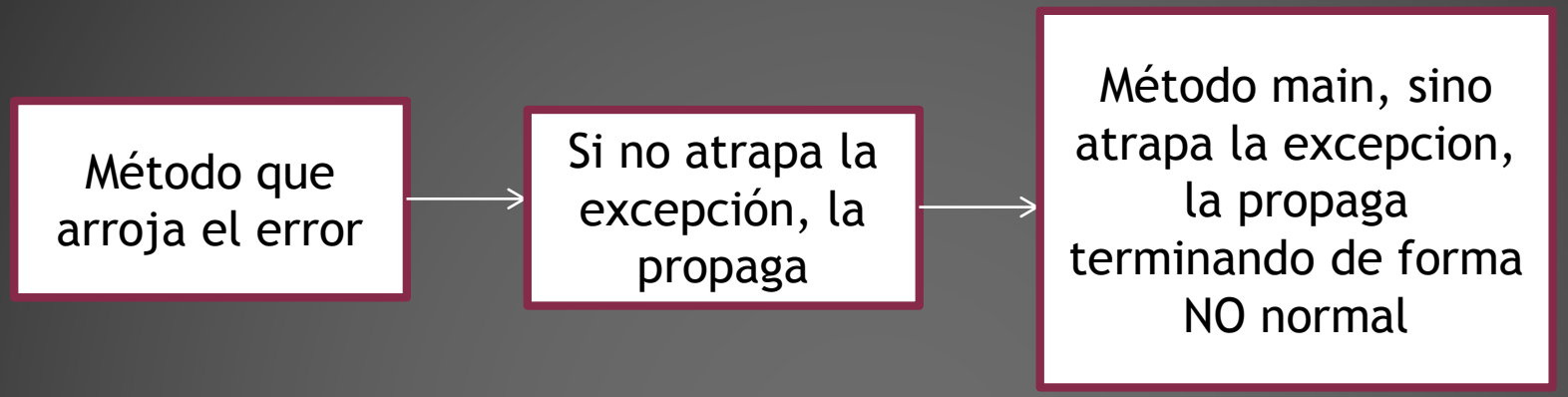


USO DE EXCEPCIONES EN JAVA

- Para manejar una excepción en java utilizamos el bloque try/catch
- El bloque finally es opcional este siempre se va a ejecutar ocurra o no la excepción.

```
try {
    int budget = 1000;
    System.out.println("Success");
}
catch(Exception ex) {
    System.out.println(ex);
}
finally {
    System.out.println("This always runs");
}
```

USO DE EXCEPCIONES EN JAVA
PROPAGACION DE EXCEPCIONES



- Si en el cuerpo de un método se lanza una excepción (de un tipo derivado de la clase Exception), en la cabecera del método hay que añadir una clausula throws que incluye una lista de los tipos de excepciones que se pueden producir al invocar al método.

Ejemplo

```
public String leerFichero (String nombreFichero)
                throws IOException
...
```

USO DE EXCEPCIONES EN JAVA
PROPAGACION DE EXCEPCIONES

1. Un método que propaga una excepción:

```
public void f() throws IOException
{
  // Fragmento de código que puede
  // lanzar una excepción de tipo IOException
}
```

2. Un método equivalente que no propaga la excepción:

```
public void f()
{
  // Fragmento de código libre de excepciones

  try {
      // Fragmento de código que puede
      // lanzar una excepción de tipo IOException
      // (p.ej. Acceso a un fichero)
  } catch (IOException error) {
      // Tratamiento de la excepción
  } finally {
      // Liberar recursos (siempre se hace)
  }
}
```

- La clausula throws nos dice el tipo de excepción que arroja cierto método.
- Una excepción que extiende de Exception SI obliga a declararla en la firma del método.
- Una excepción que extiende de RuntimeException NO estamos obligados a declararla en la firma del método.

USO DE EXCEPCIONES EN JAVA
CREACION DE EXCEPCIONES PROPIAS

```
public DivideByZeroException
       extends ArithmeticException
{
    public DivideByZeroException(String Message)
    {
        super(message);
    }
}
```

```
public double dividir(int num, int den)
       throws DivideByZeroException
{
  if (den==0)
     throw new DivideByZeroException("Error!");

  return ((double) num/(double)den);`
}
```

- Podemos crear clases de excepción, las cuales pueden extender de la clase Exception, RuntimeException o una ya existente.
- La intención de crear nuestras propias clase de excepción es personalizar los mensajes de error.