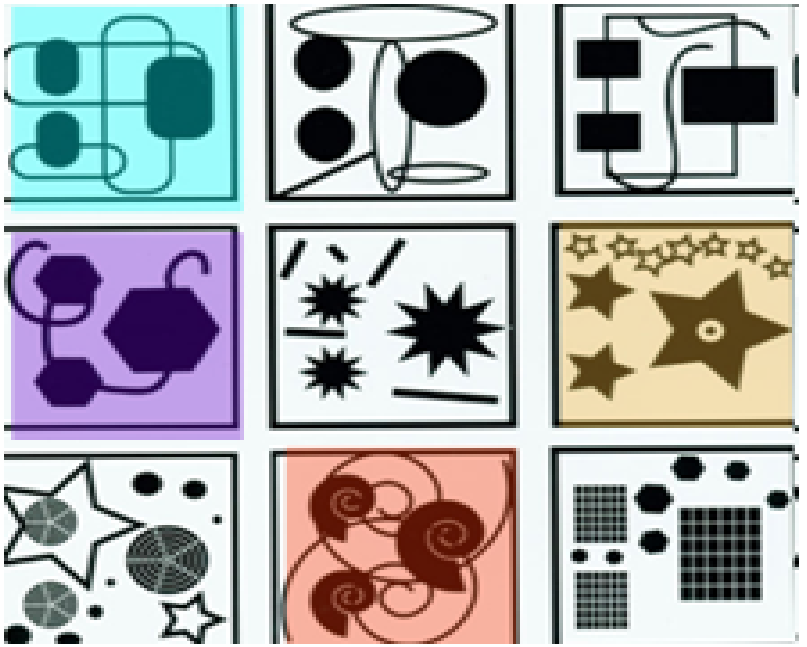
Tema 4



Conceptos complementarios en Orientación a Objetos

Lección 4.2

Manejo de excepciones

Objetivos de aprendizaje

- Conocer el concepto de excepción
- Conocer los mecanismos para el tratamiento de excepciones
- Ser capaz de usar excepciones tanto en Java como en Ruby

Contenidos

1. Definiciones
2. Envío de mensajes entre objetos
3. Esquema del tratamiento de excepciones
4. Mecanismos en los lenguajes OO
5. Sintaxis en Java
6. Sintaxis en Ruby
7. Tipos de excepciones
8. Ejemplo en Java
9. Ejemplo en Ruby

1. Definiciones

Excepción:

Situación provocada por un problema durante la ejecución, que implica que el programa termine de forma anormal, a no ser que se maneje la excepción.

Ejemplos:

- El usuario introduce un valor erróneo en una variable.
- Se necesita un fichero que no existe.
- Un objeto no puede responder un mensaje en su estado actual.

Tratamiento/manejo de la excepción:

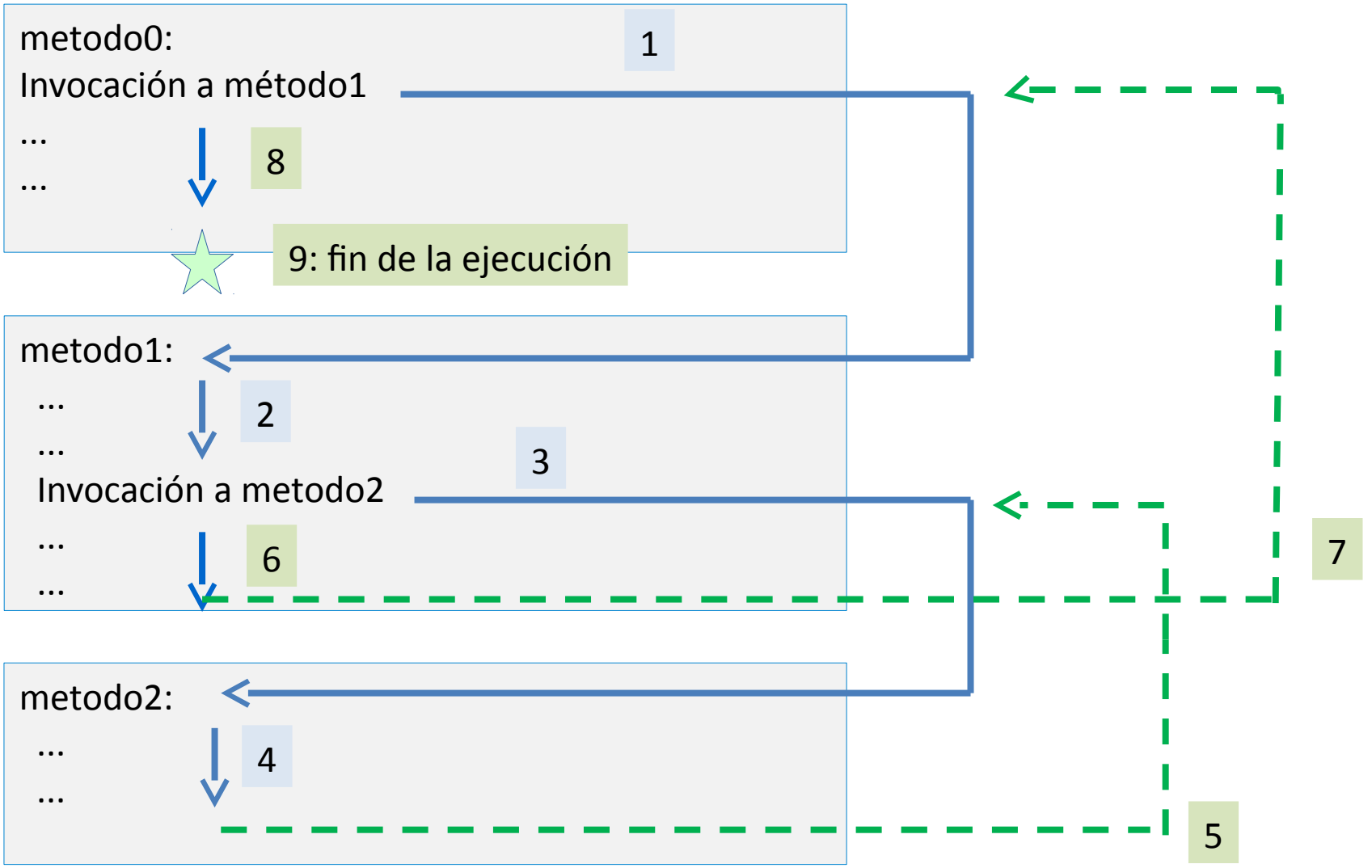
Evita la terminación anormal, detectando el problema y dando una terminación alternativa para recuperarse del error.

Ejemplos:

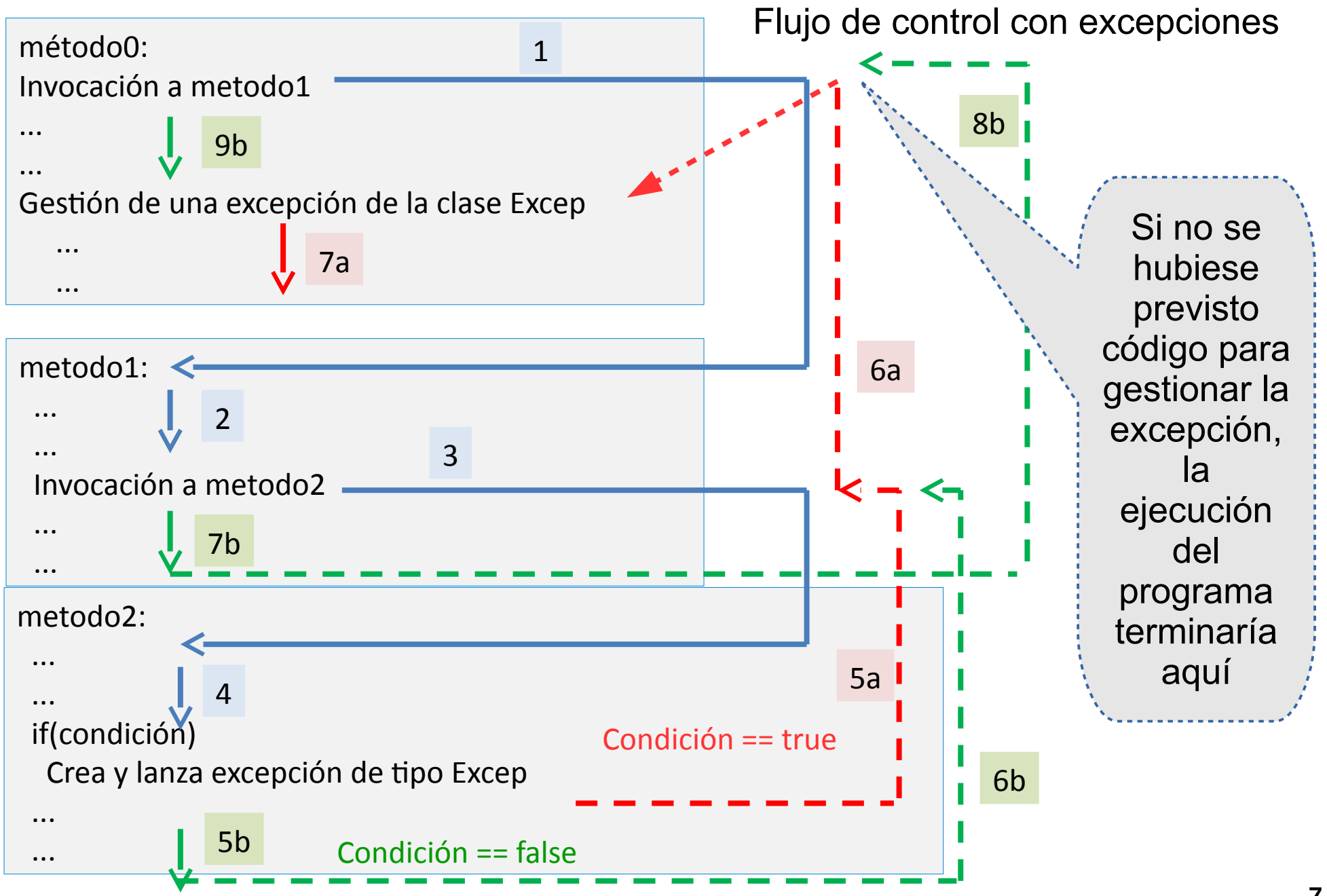
- Se muestra de forma comprensible un mensaje de error.
- Se cambia el valor de la variable o el estado del objeto.

2. Envío de mensajes entre objetos

Flujo de control normal que se produce durante el envío de mensaje



3. Esquema del tratamiento de excepciones



4. Mecanismos en los lenguajes OO

Los lenguajes de programación OO proveen de mecanismos para:

- La definición e instanciación de clases/tipos de Excepciones.
- El lanzamiento de excepciones, cuando se producen.
- La captura y tratamiento de excepciones.
- En algunos casos: avisar en la cabecera que un método propaga determinado tipo de excepciones.

Ejemplo de Excepciones
EjemplosTema4.zip[Java|Ruby]
Paquete Excepciones

5. Sintaxis en Java: 2 alternativas

```
try{ ...
    a.metodo();
    ...

}catch (UnTipoDeExcepcion e) {
    // Código cuando se produzca una excepción de tipo UnTipoDeExcepcion
       dentro del código try
}catch (OtroTipoDeExcepcion o){
    //Código cuando se produzca una excepción de tipo OtroTipoDeExcepcion
       dentro del código try
}finally {
    //Código que se ejecuta independientemente de que se lance o no una
       excepción dentro del código try
}
```

```
void metodo() throws UnTipoDeExcepcion, OtroTipoDeExcepcion {
    …
    if (algopasa)
        throw new UnTipoDeExcepcion("mensaje_error1");
    …
    if (otracosapasa)
        throw new OtroTipoDeExcepcion("mensaje_error2");
    …
}
```

6. Sintaxis en Ruby

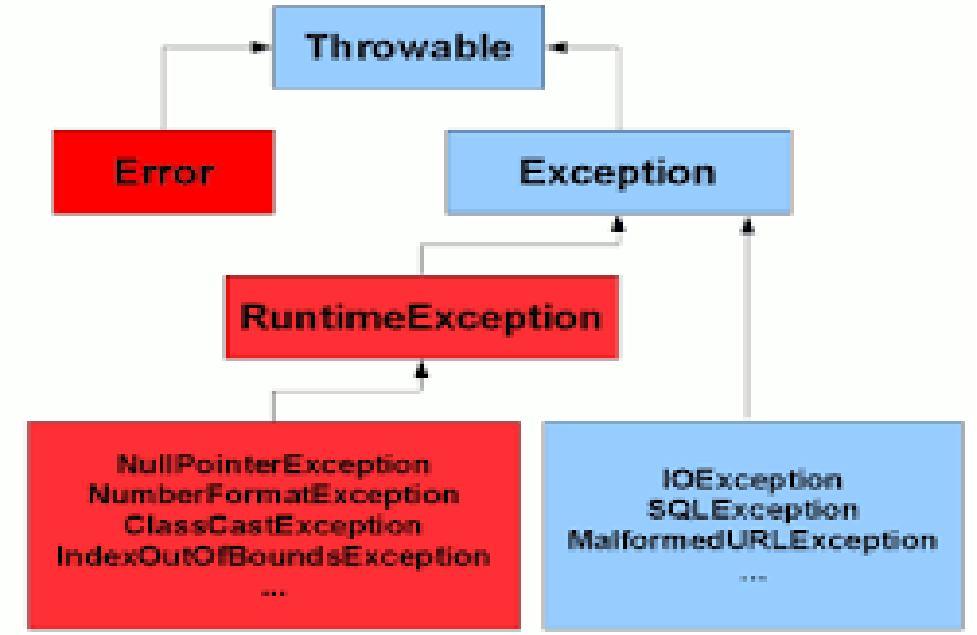
```ruby
begin
    ...
    a.metodo
    ...

rescue UnTipoDeExcepcion => e
    # Código cuando se produzca una excepción de tipo UnTipoDeExcepcion
    # dentro del begin anterior
rescue OtroTipoDeExcepcion => o
    # Código cuando se produzca una excepción de tipo OtroTipoDeExcepcion
    # dentro del begin anterior
else
    # Código que se ejecuta si no hay errores dentro del begin
ensure
    # Código que se ejecuta independientemente de que se lance o no una
    # excepción
end
```

```ruby
def metodo
    ...
    if (algopasa)
        raise UnTipoDeExcepcion, 'mensaje_error1'
    ...
    if (otracosapasa)
        raise OtroTipoDeExcepcion, 'mensaje_error2'
    ...
end
```

7. Tipos de excepciones

Además, los lenguajes de programación proveen clases correspondientes a los tipos de excepciones más comunes.

- Ejemplos de algunas clases de excepciones de Java:
  - ArrayStoreException
  - NullPointerException
  - ArrayIndexOutOfBoundsException
  - NumberFormatException
  - ClassNotFoundException

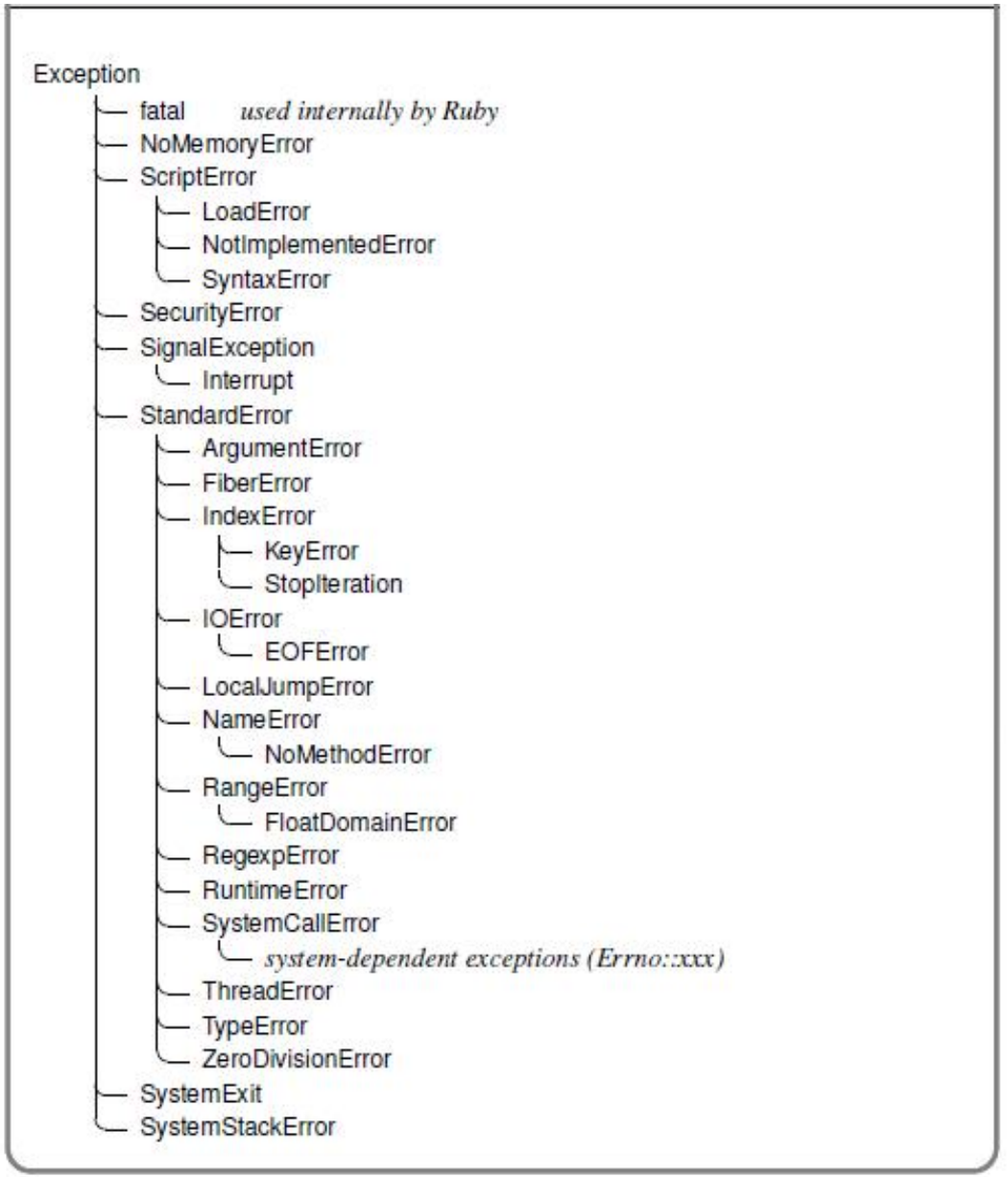


Jerarquía de excepciones en Java

7. Tipos de excepciones

Ejemplos de algunas clases de excepciones de Ruby:

- RuntimeError
- NoMethodError
- ArgumentError
- IOError



8. Ejemplo en Java

Código con excepción **sin** su tratamiento definido:

```
int divisor = 0;
   int resultado = 100/divisor;
   System.out.println("Resultado: " + resultado);
      /* Salida: Exception in thread "main"
         java.lang.ArithmeticException: / by zero */
```

Se detecta el error y se termina la ejecución con un mensaje por defecto

Código con excepción **con** su tratamiento definido:

```
   try {
         int divisor = 0;
         int resultado = 100/divisor;
         System.out.println("Resultado: " + resultado);
   } catch (ArithmeticException ex) {
         System.out.println("Error, dividiendo por cero");
            // Salida: Error, dividiendo por cero
         System.out.print(ex.getMessage());
           // Salida: / by zero
   }
```

Se detecta el error y se pasa a su tratamiento

9. Ejemplo en Ruby

Código con error **sin** su tratamiento definido:

```
divisor = 0
resultado = 100/divisor
puts "Resultado: " + resultado
    # Salida: ZeroDivisionError: divided by 0
```

Se produce el error y se termina la ejecución con un mensaje de error por defecto

Código con error **con** su tratamiento definido:

```
begin
   divisor = 0
   resultado = 100/divisor
   puts "Resultado:" + resultado
rescue ZeroDivisionError => ex
   ERR.puts "Error, dividiendo por cero"
end
    # Salida:  Error, dividiendo por cero
puts ex.message
    #  Salida: divided by 0
```

Se detecta el error y se pasa a su tratamiento, al estar éste definido