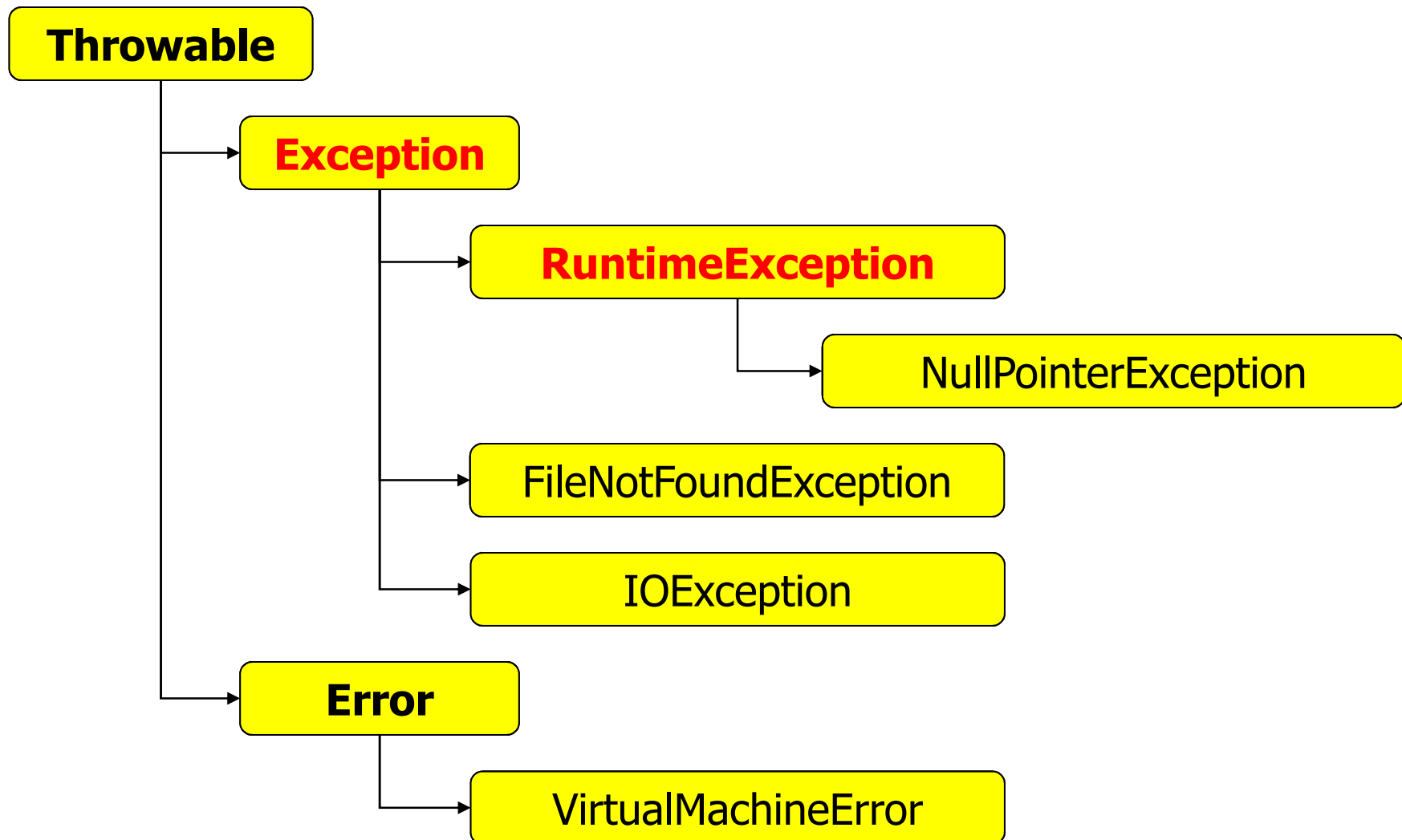




Excepciones y Finally en Java

Eduardo Ostertag Jenkins, Ph.D.
OBCOM INGENIERIA S.A. (Chile)
Eduardo.Ostertag@obcom.cl

Tipos de excepciones en Java



Ejemplo: leer propiedades de un archivo

```
public Properties readProperties(String fileName)
    throws FileNotFoundException, IOException
{
    // Nos aseguramos que fileName no sea null
    if (fileName == null)
        throw new NullPointerException("fileName es null");

    // Abrimos archivo de propiedades y cargamos contenido
    Properties properties = new Properties();
    FileInputStream in = new FileInputStream(fileName);
    properties.load(in);
    in.close();

    // Retornamos propiedades
    return properties;
}
```

Ahora queremos usar readProperties

- Ahora queremos usar readProperties, pero vamos a tener que hacer algo con **FileNotFoundException** y **IOException**
- Podemos declararlas (throws) y redispararlas (throw) a quien nos llamó para que allí se preocupen del problema
- Con esta técnica, un método debe declarar todas las excepciones de los métodos a los cuales invoca
- ¿Vamos a declarar todos los nombres de las excepciones, o vamos a decir simplemente "throws Exception"?
- Pero con "throws Exception" se pierde el detalle de las excepciones que pueden producirse, y el que nos llama no puede atraparlas usando "catch" selectivos

getUserName versión 1 y 2: declarativas

```
public String getUserNameV1(String fileName)
    throws FileNotFoundException, IOException
{
    Properties props = readProperties(fileName);
    return props.getProperty("user.name");
}
```

```
public String getUserNameV2(String fileName)
    throws Exception // ← ¡flojo! ☹️
{
    Properties props = readProperties(fileName);
    return props.getProperty("user.name");
}
```

- Esto está bien, pero ¿podemos evitar el **throws**?

getUserName versión 3: asquerosa

- Técnica asquerosa: **tragarse la excepción**

```
public String getUserNameV3(String fileName)
{
    try {
        Properties props = readProperties(fileName);
        return props.getProperty("user.name");
    } catch (Exception ex) {
        ex.printStackTrace(); // ← poco útil
        return "";           // ← ¡asqueroso!
    }
}
```

getUserName versión 4: aceptable

- Técnica aceptable: **empaquetar la excepción**

```
public String getUserNameV4(String fileName)
{
    try {
        Properties props = readProperties(fileName);
        return props.getProperty("user.name");
    } catch (Exception ex) {
        throw new RuntimeException(ex);
    }
}
```

getUserName versión 5: nice

- Técnica nice: **empaquetar con valor agregado**

```
public String getUserNameV5(String fileName)
{
    try {
        Properties props = readProperties(fileName);
        return props.getProperty("user.name");
    } catch (Exception ex) {
        throw new RuntimeException(
            "No se pudo leer nombre del usuario" +
            " desde el archivo: " + fileName, ex);
    }
}
```

getUserName versión 6: very nice

- Técnica very nice: **utilizar excepción propia**

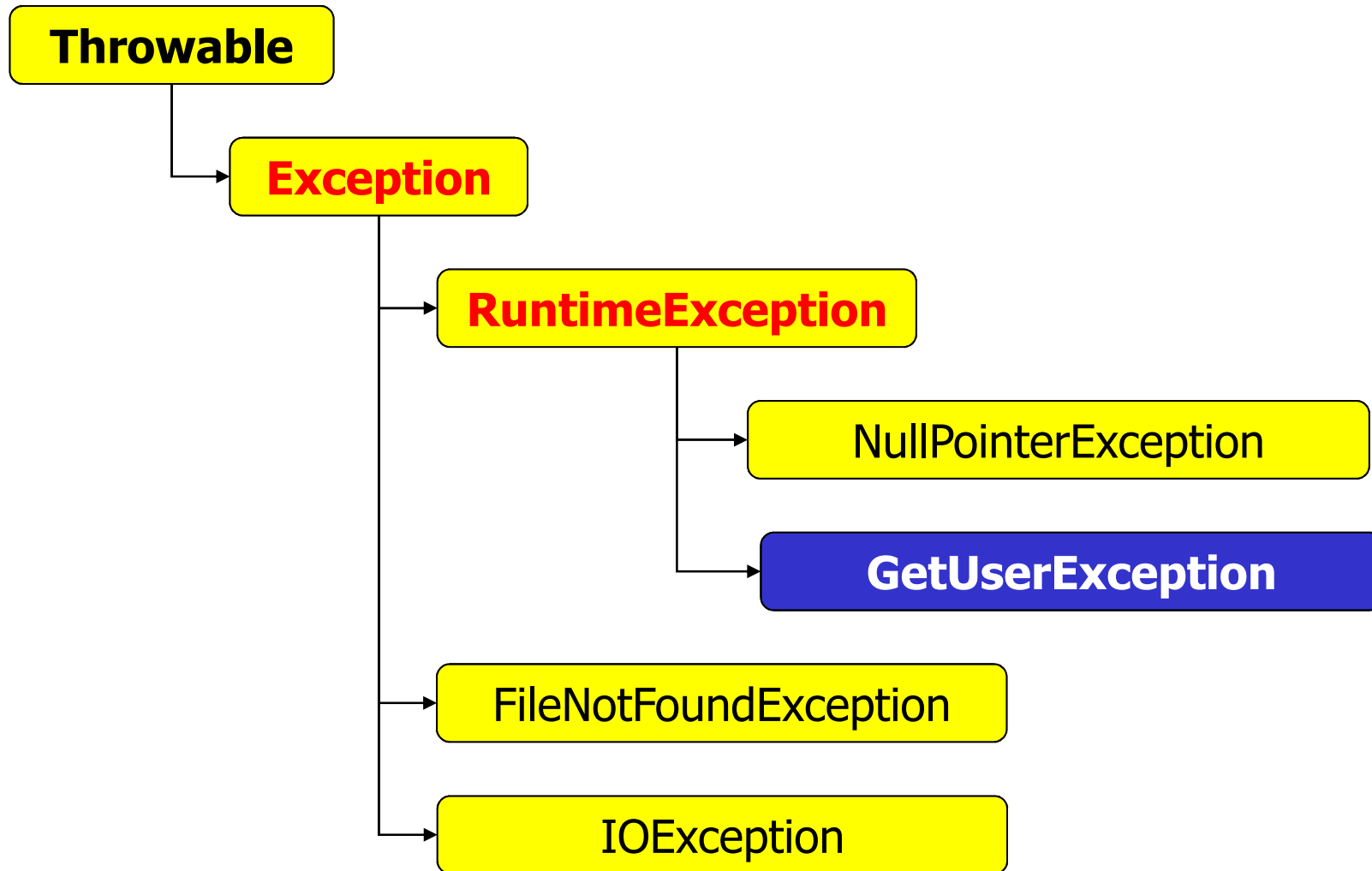
```
public String getUserNameV6(String fileName)
{
    try {
        Properties props = readProperties(fileName);
        return props.getProperty("user.name");
    } catch (Exception ex) {
        throw new GetUserException(fileName, ex);
    }
}
```

Declaración de GetUserException

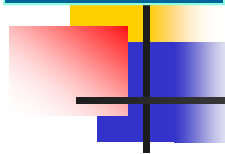
```
public class GetUserException extends RuntimeException
{
    public GetUserException(String fileName, Throwable cause)
    {
        super(fileName, cause);
    }
}
```

```
public class GetUserException extends RuntimeException
{
    public GetUserException(String fileName, Throwable cause)
    {
        super("No se pudo leer nombre del usuario" +
            " desde el archivo: " + fileName, cause);
    }
}
```

Excepción propia GetUserException



OBCOM



Finally

Finally

¿Está liberando siempre los recursos?

```
public Properties readProperties(String fileName)
    throws FileNotFoundException, IOException
{
    // Nos aseguramos que fileName no sea null
    if (fileName == null)
        throw new NullPointerException("fileName es null");

    // Abrimos archivo de propiedades y cargamos contenido
    Properties properties = new Properties();
    FileInputStream in = new FileInputStream(fileName);
    properties.load(in);
    in.close(); ←

    // Retornamos propiedades
    return properties;
}
```

¿Siempre se ejecuta este "close"?

Siempre libere los recursos con "finally"

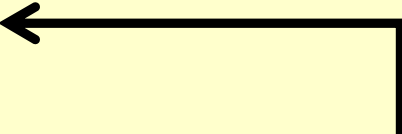
```
public Properties readProperties(String fileName)
    throws FileNotFoundException, IOException
{
    // Nos aseguramos que fileName no sea null
    if (fileName == null)
        throw new NullPointerException("fileName es null");

    // Abrimos archivo de propiedades y cargamos contenido
    Properties properties = new Properties();
    FileInputStream in = new FileInputStream(fileName);
    try {
        properties.load(in);
    } finally {
        in.close();
    }

    // Retornamos propiedades
    return properties;
}
```

¿Programó correctamente el "finally"?

```
public void saveClientName(DataSource dsource, String clientName)
    throws SQLException
{
    Connection conn = null;
    CallableStatement call = null;
    try {
        conn = dsource.getConnection();
        call = conn.prepareCall("{call SaveName(?)}")
        call.setString(1, clientName);
        call.execute();
    } finally {
        if (call != null)
            call.close();
        if (conn != null)
            conn.close();
    }
}
```



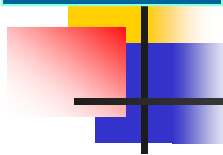
¿Siempre se ejecuta este "close"?

Un "finally" debe liberar sólo un recurso

```
public void saveClientName(DataSource dsource, String clientName)
    throws SQLException
{
    Connection conn = dsource.getConnection();
    try {
        CallableStatement call = conn.prepareCall("{call SaveName(?)}");
        try {
            call.setString(1, clientName);
            call.execute();
        } finally {
            call.close();
        }
    } finally {
        conn.close();
    }
}
```

- ¡Nunca se trague una excepción!
- Alternativas de tratar una excepción:
 - Atrape (**catch**) y resuelva la excepción
 - Declare (**throws**) y dispare (**throw**)
 - Empaquete (**RuntimeException**) y dispare
- Siempre libere los recursos (**finally**)

OBCOM



Muchas gracias

Muchas

Gracias