Introducción a la Programación en Java

Excepciones

- Es un evento que ocurre durante la ejecución del programa que **interrumpe el flujo normal de las sentencias**.
- Algunos ejemplos de errores y situaciones excepcionales son:
  - No hay memoria disponible para asignar
  - Acceso a un elemento de un array fuera de rango
  - Leer por teclado un dato de un tipo distinto al esperado
  - Error al abrir un fichero
  - División por cero
  - Problemas de Hardware

Excepciones (cont.)

```
public static void main(String[] args) {
    int a = 4, b=0;
    System.out.println(a/b);
}
```

provoca:

```
Exception in thread "main" java.lang.ArithmeticException: / by zero
```

```
public static void main(String[] args) {
    int [] array = new int [5];
    array[5] = 1;
}
```

provoca:

```
Exception in thread "main" java.lang.ArrayIndexOutOfBoundsException: 5
```

```
public static void main(String[] args) {
    Scanner sc = new Scanner(System.in);
    System.out.print("Introduce un número entero: ");
    int n = sc.nextInt();
    System.out.print("Número introducido: " + n);
}
```

Se espera un int. Si por ejemplo se lee 4,5 provoca:

```
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 excepciones1.Excepciones1.main(Excepciones1.java:7)
```

Excepciones (cont.)

- ¿Qué ocurre cuando se produce una excepción?

La Máquina Virtual Java crea un objeto excepción y lo lanza. Este objeto excepción contiene información sobre el error.

El sistema busca en el método donde se ha producido la excepción un manejador de excepciones que trate la excepción.

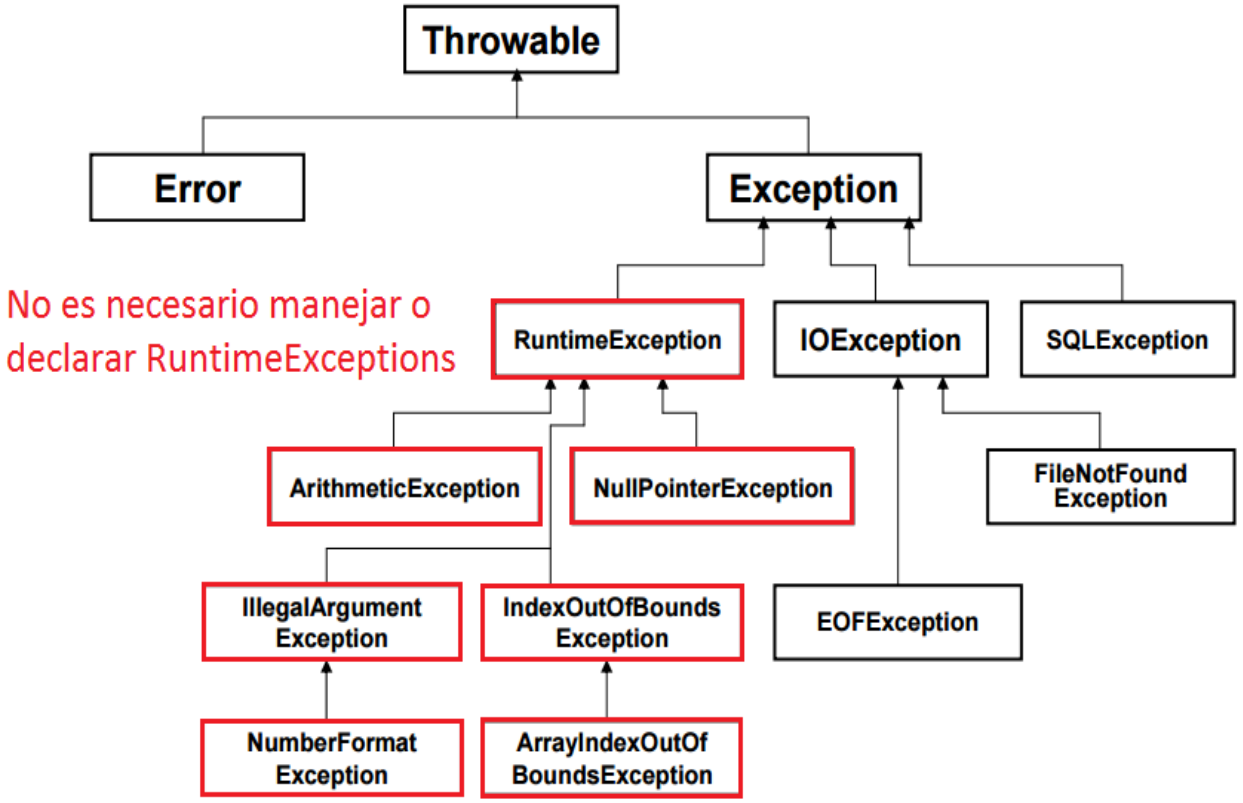
Si el método no contiene un manejador para la excepción se busca en el método que llamó a este y así sucesivamente en toda la pila de llamadas.

Cuando se encuentra un manejador apropiado este manejará la excepción.

Si no se encuentra un manejador adecuado la Máquina Virtual Java muestra el error y acaba el programa.

Excepciones (cont.)

- Todas las excepciones en Java se representan, a través de objetos que heredan, en última instancia, de la clase **java.lang.Throwable**.



Excepciones (cont.)

- Tipos de excepciones
  - Error
  - Checked
  - Unchecked

Excepciones (cont.)

```
String[] lista2 = {"1", "2", "3", "4", "cinco"};
int sum = 0;
for (String s : lista2) {
    sum += Integer.parseInt(s);
}
System.out.println("El resultado es:" + sum);
```

```
run:
Exception in thread "main" java.lang.NumberFormatException: For input string: "cinco"
        at java.lang.NumberFormatException.forInputString(NumberFormatException.java:65)
        at java.lang.Integer.parseInt(Integer.java:580)
        at java.lang.Integer.parseInt(Integer.java:615)
        at clase.pkg4.Clase4.main(Clase4.java:37)
Java Result: 1
BUILD SUCCESSFUL (total time: 0 seconds)
```

Excepciones (cont.)

- Try – catch – finally
  - El código dentro del try puede generar más de una excepción, y se pueden capturar todas ellas añadiendo varios bloques catch

```
try {
    // código que podría provocar una excepción.
} catch (Exception e) {
    // código que se ejecuta si se provoca una excepción
} finally {
    // código que se ejecutará siempre
}
```

Ver ejemplo

Excepciones (cont.)

```
try {
        // Instrucciones que pueden generar una excepción
}
catch(tipoExcepcion e){

        // Instrucciones para tratar esta excepción
}
catch(otroTipoExcepcion e){

        // Instrucciones para tratar esta otra excepción
}
finally{

    // Instrucciones que se ejecutarán siempre
}
```

Excepciones (cont.)

- Ejemplo 1:

```
try {

    for (String s : lista2) {
        sum2 += Integer.parseInt(s);
    }
    System.out.println("El resultado es:" + sum2);
} catch (NumberFormatException e) {

    System.err.println("Se ha producido la siguiente excepcion: " + e.getMessage()
}
```

```
run:
Se ha producido la siguiente excepcion: For input string: "cinco"
BUILD SUCCESSFUL (total time: 0 seconds)
```

Ver ejemplo

Excepciones (cont.)

- Ejemplo 2:

```
String[] lista3 = {"1", "2", "3", "4", "cinco"};
int sum3 = 0;
for (String s : lista3) {

    try {
        sum3 += Integer.parseInt(s);
    } catch (NumberFormatException e) {

        System.err.println("Se ha producido la siguiente excepcion: " + e.getMessage());
    }
}
System.out.println("El resultado es:" + sum3);
```

```
run:
El resultado es:10
Se ha producido la siguiente excepcion: For input string: "cinco"
BUILD SUCCESSFUL (total time: 0 seconds)
```

Ver ejemplo 1 y 2

Lanzar Excepciones

- Parte del tratamiento de excepciones puede incluir la propagación de la misma. Imagine que existen dos métodos A y B, donde el método A invoca al método B, si B genera una excepción (de tipo checked) puede lanzarla al método A. Entonces A deberá gestionarla mediante try-catch o lanzarla nuevamente.
- **Throws**

Ver ejemplo: excepción lanzada.

Crear clases que deriven de Exception

Se pueden crear clases que hereden de la clase Exception, implementando así **excepciones personalizadas**.

Ver ejemplo: customException

Hay otro modo de lanzar excepciones mediante la palabra reservada **throw**

```
public void ValidarPass(String pass, String rpass) throws CustomException{

    if(!pass.equals(rpass)){
        //Lanzar el error, pero es una nueva excepcion por lo tanto debe crearse
        throw new CustomException("Las contraseñas no coinciden");
    }
}
```

Ver ejemplo: validarPass

Base de datos y Java

- El API JDBC es el estándar de conexión entre el lenguaje de programación Java y un amplio rango de bases de datos
- El uso del API JDBC hace posible llevar a cabo lo siguiente:
  - Establecer una conexión con una base de datos o acceder a cualquier fuente de datos tabular
  - Enviar enunciados SQL.
  - Procesar los resultados

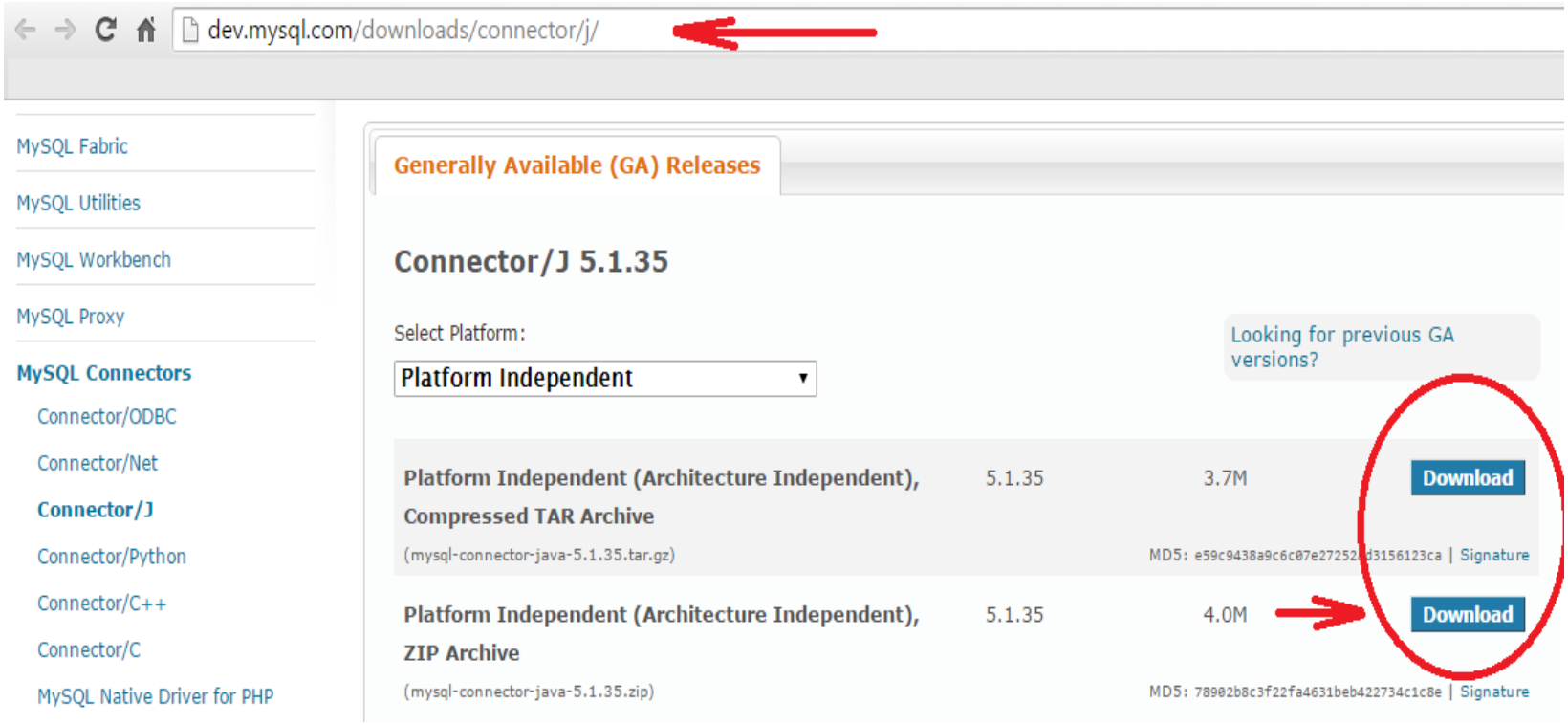
Base de datos y Java (cont.)

Las interfaces principales de la API JDBC se encuentran en el paquete java.sql

DriverManager	Proporciona métodos para cargar drivers
Connection	Representa una conexión a una BD
Statement	Permite enviar sentencias SQL a la BD
PreparedStatement	Para sentencias SQL que toma parámetro(s).
CallableStatement	se usan para ejecutar procedimientos almacenados
ResultSet	Almacena los resultados de un comando ejecutado

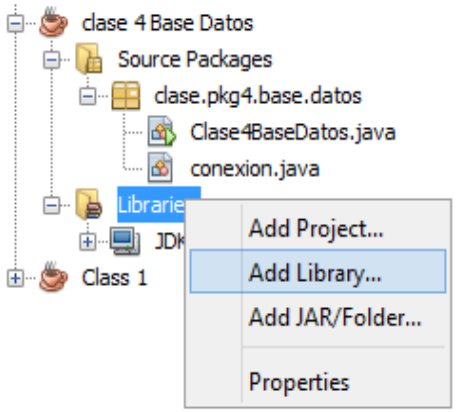
Base de datos y Java (cont.)

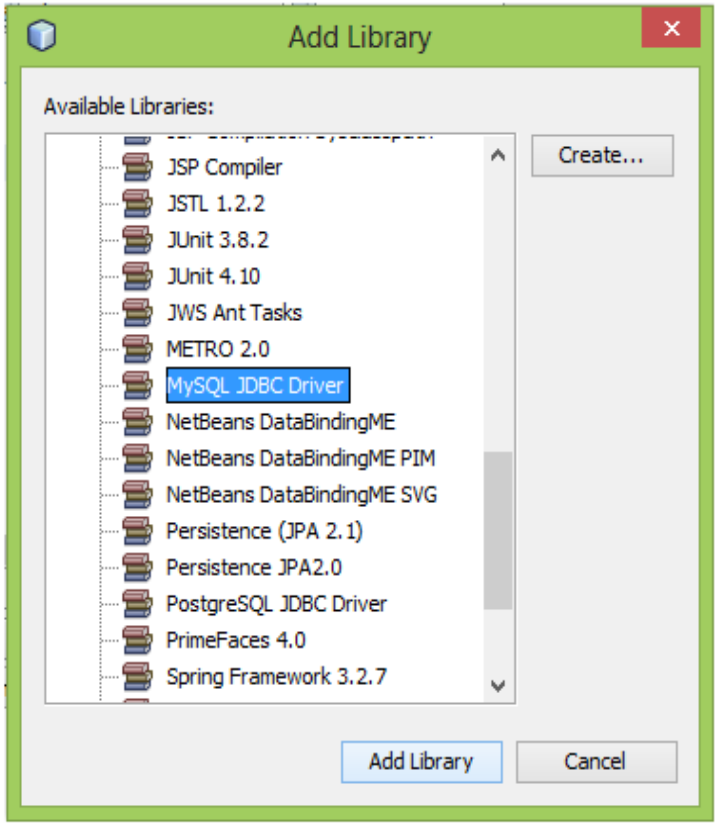
1. Descargando el jar para la conexión.



Base de datos y Java (cont.)

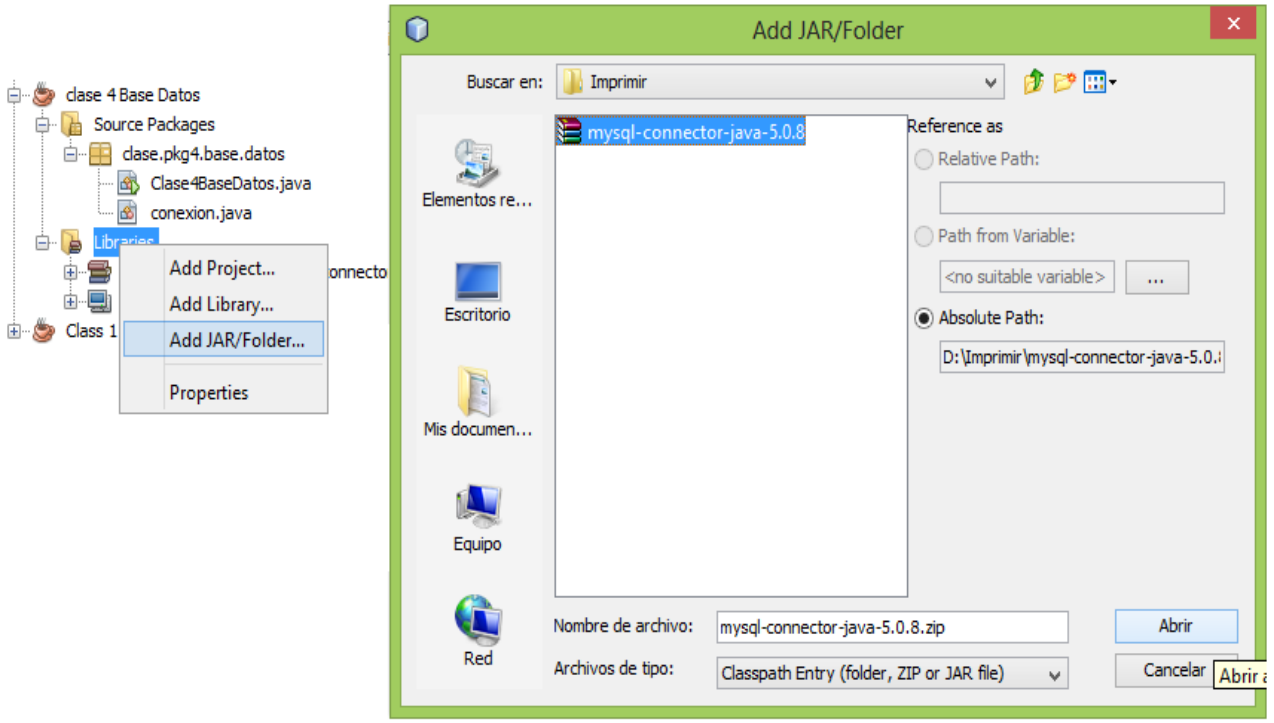
2. Agregar las librerías necesarias





Base de datos y Java (cont.)

3. Añadir jar



DB. Ejemplo práctico

- Casi todos los métodos relativos a BD pueden lanzar la excepción **SQLException**
  - La mayoría de las excepciones pertenecerán al paquete java.sql.*
  - Estas excepciones obligan a utilizar try – catch

Pasos para usar JDBS

1. Crear una instancia del JDBC driver.

```
28          try
29          {
30              //inicializar y registrar el driver
                Class.forName("com.mysql.jdbc.Driver");
32          }
33          catch(ClassNotFoundException ex)
34          {
35
36          }
37
```

Pasos para usar JDBS (cont.)

2. Especificar la url de la base de datos.
3. Establecer una conexión usando el driver que crea el objeto Connection.

```
try
{
    //inicializar y registrar el driver
    Class.forName("com.mysql.jdbc.Driver");

    //Establecer la conexion
    String servidor = "jdbc:mysql://localhost/DBVentas"; //url de la BD
    String usuarioDB = "curso";
    String passwordDB = "curso";
    conexion= DriverManager.getConnection(servidor,usuarioDB,passwordDB);
}
catch(ClassNotFoundException ex)
{
    JOptionPane.showMessageDialog(null, ex, "Error 1 en la Conexión con la BD "
    conexion = null;
}
```

Ver ejemplo: clase conexión

Pasos para usar JDBS (cont.)

4. Crear un objeto Statement, usando Connection.
5. Armar el postulado SQL y enviarlo a ejecución usando el Statement.
6. Recibir los resultados en el objeto ResultSet.

```
con = conexion.GetConnection();
Statement s = con.createStatement();
//ResultSet: manipulación de registros en consultas de tipo Select
ResultSet r = s.executeQuery("select * from vendedores");

while (r.next()) {
    System.out.println(r.getString("nombre") + " " + r.getDate("fechaAlta"));
}

con.close();
```

Ver ejemplo: consultas

Pasos para usar JDBS (cont.)

- Ejercicio
  - Usando la conexión implementada en clases, realizar las siguientes consultas
    - Insertar un producto
    - Actualizar un producto
    - Eliminar un producto

Hemos terminado por hoy